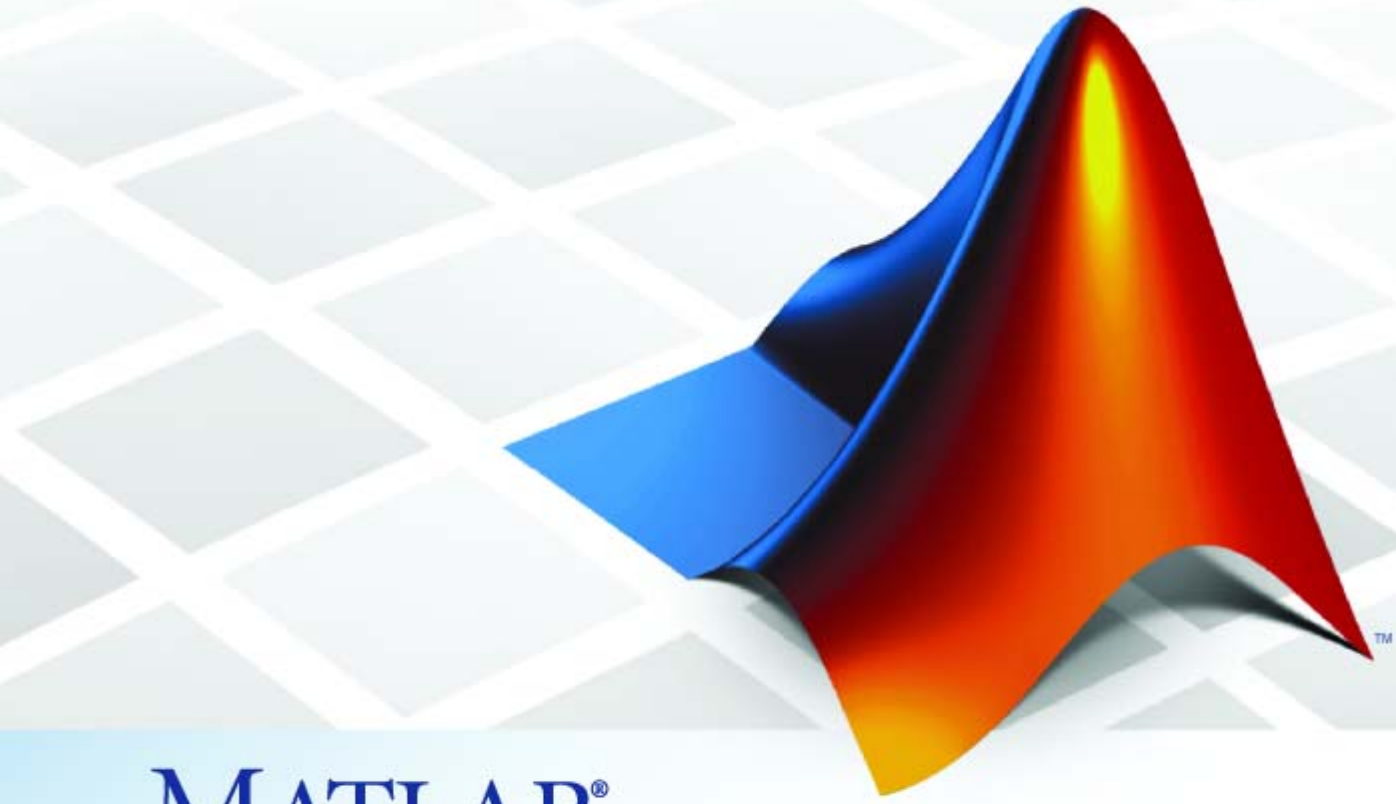
Signal Processing Blockset™ 6
Reference



MATLAB®
&SIMULINK®

The MathWorks™
Accelerating the pace of engineering and science

How to Contact The MathWorks

www.mathworks.com	Web
comp.soft-sys.matlab	Newsgroup
www.mathworks.com/contact_TS.html	Technical Support

suggest@mathworks.com	Product enhancement suggestions
bugs@mathworks.com	Bug reports
doc@mathworks.com	Documentation error reports
service@mathworks.com	Order status, license renewals, passcodes
info@mathworks.com	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

Signal Processing Blockset™ Reference

© COPYRIGHT 1995–2009 The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

The MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

April 1995	First printing	New for Version 1.0
May 1997	Second printing	Revised for Version 2.0
January 1998	Third printing	Revised for Version 2.2 (Release 10)
January 1999	Fourth printing	Revised for Version 3.0 (Release 11)
November 2000	Fifth printing	Revised for Version 4.0 (Release 12)
June 2001	Online only	Revised for Version 4.1 (Release 12.1)
July 2002	Sixth printing	Revised for Version 5.0 (Release 13)
April 2003	Seventh printing	Revised for Version 5.1 (Release 13SP1)
June 2004	Online only	Revised for Version 6.0 (Release 14) (Renamed from DSP Blockset User's Guide)
October 2004	Online only	Revised for Version 6.0.1 (Release 14SP1)
March 2005	Online only	Revised for Version 6.1 (Release 14SP2)
September 2005	Online only	Revised for Version 6.2 (Release 14SP3)
March 2006	Online only	Revised for Version 6.3 (Release 2006a)
September 2006	Online only	Revised for Version 6.4 (Release 2006b)
March 2007	Online only	Revised for Version 6.5 (Release 2007a)
September 2007	Online only	Revised for Version 6.6 (Release 2007b)
March 2008	Online only	Revised for Version 6.7 (Release 2008a)
October 2008	Online only	Revised for Version 6.8 (Release 2008b)
March 2009	Online only	Revised for Version 6.9 (Release 2009a)

Contents

1 Block Reference

Estimation 1-2
Linear Prediction 1-2
Parametric Estimation 1-3
Power Spectrum Estimation 1-3

Filtering 1-4
Adaptive Filters 1-4
Filter Design Toolbox 1-4
Filter Designs 1-5
Multirate Filters 1-6

Math Functions 1-6
Math Operations 1-7
Matrices and Linear Algebra 1-7
Polynomial Functions 1-10

Quantizers 1-11

Signal Management 1-11
Buffers 1-12
Indexing 1-12
Signal Attributes 1-13
Switches and Counters 1-13

Signal Operations 1-14

Signal Processing Sinks 1-15

Signal Processing Sources 1-16

Statistics 1-17

Transforms 1-17

2 Blocks — Alphabetical List

3 Function Reference

Glossary

Index

Block Reference

Estimation (p. 1-2)	Perform spectrum estimates and autoregressive modeling
Filtering (p. 1-4)	Design, create, and work with filters
Math Functions (p. 1-6)	Perform linear algebra and basic math calculations
Quantizers (p. 1-11)	Design and implement quantization schemes
Signal Management (p. 1-11)	Perform basic signal processing operations
Signal Operations (p. 1-14)	Control signal attributes, buffer signals, and index signals
Signal Processing Sinks (p. 1-15)	View or log signals
Signal Processing Sources (p. 1-16)	Generate discrete-time signals
Statistics (p. 1-17)	Perform statistical computations on signals
Transforms (p. 1-17)	Compute transforms

Estimation

Linear Prediction (p. 1-2)	Compute or work with linear predictive representations
Parametric Estimation (p. 1-3)	Compute estimates of autoregressive model parameters
Power Spectrum Estimation (p. 1-3)	Compute parametric and nonparametric spectral estimates

Linear Prediction

Autocorrelation LPC	Determine coefficients of Nth-order forward linear predictors
Levinson-Durbin	Solve linear system of equations using Levinson-Durbin recursion
LPC to LSF/LSP Conversion	Convert linear prediction coefficients to line spectral pairs or line spectral frequencies
LPC to/from Cepstral Coefficients	Convert linear prediction coefficients to cepstral coefficients or cepstral coefficients to linear prediction coefficients
LPC to/from RC	Convert linear prediction coefficients to reflection coefficients or reflection coefficients to linear prediction coefficients
LPC/RC to Autocorrelation	Convert linear prediction coefficients or reflection coefficients to autocorrelation coefficients
LSF/LSP to LPC Conversion	Convert line spectral frequencies or line spectral pairs to linear prediction coefficients

Parametric Estimation

Burg AR Estimator	Compute estimate of autoregressive (AR) model parameters using Burg method
Covariance AR Estimator	Compute estimate of autoregressive (AR) model parameters using covariance method
Modified Covariance AR Estimator	Compute estimate of autoregressive (AR) model parameters using modified covariance method
Yule-Walker AR Estimator	Compute estimate of autoregressive (AR) model parameters using Yule-Walker method

Power Spectrum Estimation

Burg Method	Compute parametric spectral estimate using Burg method
Covariance Method	Compute parametric spectral estimate using covariance method
Magnitude FFT	Compute nonparametric estimate of spectrum using periodogram method
Modified Covariance Method	Compute parametric spectral estimate using modified covariance method
Periodogram	Compute nonparametric estimate of spectrum
Yule-Walker Method	Compute parametric estimate of tspectrum using Yule-Walker autoregressive (AR) method

Filtering

Adaptive Filters (p. 1-4)	Use adaptive filter algorithms
Filter Design Toolbox (p. 1-4)	Design and implement single- and multirate FIR and IIR filters
Filter Designs (p. 1-5)	Design and implement filters
Multirate Filters (p. 1-6)	Implement multirate filters

Adaptive Filters

Block LMS Filter	Compute output, error, and weights using LMS adaptive algorithm
Fast Block LMS Filter	Compute output, error, and weights using LMS adaptive algorithm
Kalman Filter	Predict or estimate states of dynamic systems
LMS Filter	Compute output, error, and weights using LMS adaptive algorithm
RLS Filter	Compute filtered output, filter error, and filter weights for given input and desired signal using RLS adaptive filter algorithm

Filter Design Toolbox

Arbitrary Magnitude Filter	Design arbitrary response filter
Bandpass Filter	Design bandpass filter
Bandstop Filter	Design bandstop filter
CIC Compensator	Design CIC compensator
CIC Filter	Design Cascaded Integrator-Comb (CIC) Filter

Differentiator Filter	Design differentiator Þlter
Halfband Filter	Design halfband Þlter
Highpass Filter	Design highpass Þlter
Hilbert Filter	Design Hilbert Þlter
Inverse Sinc Filter	Design inverse sinc Þlter
Lowpass Filter	Design lowpass Filter
Nyquist Filter	Design Nyquist Þlter
Octave Filter	Design octave Þlter
Parametric Equalizer	Design parametric equalizer
Peak-Notch Filter	Design peak or notch Þlter

Filter Designs

Analog Filter Design	Design and implement analog Þlters
Biquad Filter	Model biquadratic IIR (SOS) Þlters
Digital Filter	Filter each channel of input over time using static or time-varying digital Þlter implementations
Digital Filter Design	Design and implement digital FIR and IIR Þlters
Filter Realization Wizard	Construct Þlter realizations using Digital Filter block or Sum, Gain, and Delay blocks
Overlap-Add FFT Filter	Impl ement overlap-add method of frequency-domain Þltering
Overlap-Save FFT Filter	Implement overlap-save method of frequency-domain Þltering

Multirate Filters

CIC Decimation	Decimate signal using Cascaded Integrator-Comb filter
CIC Interpolation	Interpolate signal using Cascaded Integrator-Comb filter
Dyadic Analysis Filter Bank	Decompose signals into subbands with smaller bandwidths and slower sample rates
Dyadic Synthesis Filter Bank	Reconstruct signals from subbands with smaller bandwidths and slower sample rates
FIR Decimation	Filter and downsample input signals
FIR Interpolation	Upsample and filter input signals
FIR Rate Conversion	Upsample, filter, and downsample input signals
Two-Channel Analysis Subband Filter	Decompose signal into high-frequency subband and low-frequency subband
Two-Channel Synthesis Subband Filter	Reconstruct signal from high-frequency subband and low-frequency subband

Math Functions

Math Operations (p. 1-7)	Use specialized math operations for signal processing applications
Matrices and Linear Algebra (p. 1-7)	Work with matrices
Polynomial Functions (p. 1-10)	Work with polynomials

Math Operations

Complex Exponential	Compute complex exponential function
Cumulative Product	Compute cumulative product of channel, column, or row elements
Cumulative Sum	Compute cumulative sum of channel, column, or row elements
dB Conversion	Convert magnitude data to decibels (dB or dBm)
dB Gain	Apply decibel gain
Difference	Compute element-to-element difference along specified dimension of input
Normalization	Perform vector normalization along rows, columns, or specified dimension

Matrices and Linear Algebra

Linear System Solvers (p. 1-7)	Solve matrix equation AX = B for X
Matrix Factorizations (p. 1-8)	Factor matrices
Matrix Inverses (p. 1-9)	Invert matrices
Matrix Operations (p. 1-9)	Perform basic matrix operations

Linear System Solvers

Backward Substitution	Solve UX=B for X when U is upper triangular matrix
Cholesky Solver	Solve SX=B for X when S is square Hermitian positive definite matrix

Forward Substitution	Solve LX=B for X when L is lower triangular matrix
LDL Solver	Solve SX=B for X when S is square Hermitian positive de Þnite matrix
Levinson-Durbin	Solve linear system of equations using Levinson-Durbin recursion
LU Solver	Solve AX=B for X when A is square matrix
QR Solver	Find minimum-norm-residual solution to A X=B
SVD Solver	Solve AX=B using singular value decomposition

Matrix Factorizations

Cholesky Factorization	Factor square Hermitian positive deÞnite matrix into triangular components
LDL Factorization	Factor square Hermitian positive deÞnite matrices into lower, upper, and diagonal components
LU Factorization	Factor square matrix into lower and upper triangular components
QR Factorization	Factor arbitrary matrix into unitary and upper triangular components
Singular Value Decomposition	Factor matrix using singular value decomposition

Matrix Inverses

Cholesky Inverse	Compute inverse of Hermitian positive definite matrix using Cholesky factorization
LDL Inverse	Compute inverse of Hermitian positive definite matrix using LDL factorization
LU Inverse	Compute inverse of square matrix using LU factorization
Pseudoinverse	Compute Moore-Penrose pseudoinverse of matrix

Matrix Operations

Array-Vector Add	Add vector to array along specified dimension
Array-Vector Divide	Divide array by vector along specified dimension
Array-Vector Multiply	Multiply array by vector along specified dimension
Array-Vector Subtract	Subtract vector from array along specified dimension
Constant Diagonal Matrix	Generate square, diagonal matrix
Create Diagonal Matrix	Create square diagonal matrix from diagonal elements
Extract Diagonal	Extract main diagonal of input matrix
Extract Triangular Matrix	Extract lower or upper triangle from input matrices
Identity Matrix	Generate matrix with ones on main diagonal and zeros elsewhere
Matrix 1-Norm	Compute 1-norm of matrix

Matrix Concatenate	Concatenate input signals of same data type to create contiguous output signal
Matrix Exponential	Compute matrix exponential
Matrix Multiply	Multiply or divide inputs
Matrix Product	Multiply matrix elements along rows, columns, or entire input
Matrix Square	Compute square of input matrix
Matrix Sum	Sum matrix elements along rows, columns, or entire input
Matrix Sum (Obsolete)	Sum matrix elements along rows, columns, or entire input
Overwrite Values	Overwrite submatrix or subdiagonal of input
Permute Matrix	Reorder matrix rows or columns
Reciprocal Condition	Compute reciprocal condition of square matrix in 1-norm
Submatrix	Select subset of elements (submatrix) from matrix input
Toeplitz	Generate matrix with Toeplitz symmetry
Transpose	Compute matrix transpose

Polynomial Functions

Least Squares Polynomial Fit	Compute polynomial coef Þcients that best Þt input data in least-squares sense
Polynomial Evaluation	Eval uate polynomial expression
Polynomial Stability Test	Use Schur-Cohn algorithm to determine whether all roots of input polynomial are inside unit circle

Quantizers

G711 Codec	Quantize narrowband speech input signals
Quantizer	Discretize input at specified interval
Scalar Quantizer Decoder	Convert each index value into quantized output value
Scalar Quantizer Design	Start Scalar Quantizer Design Tool (SQDTool) to design scalar quantizer using Lloyd algorithm
Scalar Quantizer Encoder	Encode each input value by associating it with index value of quantization region
Uniform Decoder	Decode integer input into floating-point output
Uniform Encoder	Quantize and encode floating-point input into integer output
Vector Quantizer Decoder	Find vector quantizer codeword that corresponds to given, zero-based index value
Vector Quantizer Design	Design vector quantizer using Vector Quantizer Design Tool (VQDTool)
Vector Quantizer Encoder	For given input, find index of nearest codeword based on Euclidean or weighted Euclidean distance measure

Signal Management

Buffers (p. 1-12)	Change sample rate or frame rate of signals by buffering or unbuffering
Indexing (p. 1-12)	Manipulate ordering of signals

Signal Attributes (p. 1-13)	Inspect or modify signal attributes
Switches and Counters (p. 1-13)	Perform actions when events occur

Buffers

Buffer	Buffer input sequence to smaller or larger frame size
Delay Line	Rebuffer sequence of inputs with one-sample shift
Queue	Store inputs in FIFO register
Stack	Store inputs into LIFO register
Unbuffer	Unbuffer input frame into sequence of scalar outputs

Indexing

Flip	Flip input vertically or horizontally
Multiport Selector	Distribute arbitrary subsets of input rows or columns to multiple output ports
Overwrite Values	Overwrite submatrix or subdiagonal of input
Selector	Select input elements from vector, matrix, or multidimensional signal
Submatrix	Select subset of elements (submatrix) from matrix input
Variable Selector	Select subset of rows or columns from input

Signal Attributes

Check Signal Attributes	Generate error when input signal does or does not match selected attributes exactly
Convert 1-D to 2-D	Reshape 1-D or 2-D input to 2-D matrix with specified dimensions
Convert 2-D to 1-D	Convert 2-D matrix input to 1-D vector
Data Type Conversion	Convert input signal to specified data type
Frame Conversion	Specify sampling mode of output signal
Inherit Complexity	Change complexity of input to match reference signal

Switches and Counters

Counter	Count up or down through specified range of numbers
Edge Detector	Detect transition from zero to nonzero value
Event-Count Comparator	Detect threshold crossing of accumulated nonzero inputs
Multiphase Clock	Generate multiple binary clock signals
N-Sample Enable	Output ones or zeros for specified number of sample times
N-Sample Switch	Switch between two inputs after specified number of sample periods

Signal Operations

Constant Ramp	Generate ramp signal with length based on input dimensions
Convolution	Compute convolution of two inputs
Delay	Delay discrete-time input by specified number of samples or frames
Downsample	Resample input at lower rate by deleting samples
Interpolation	Interpolate values of real input samples
NCO	Generate real or complex sinusoidal signals
Offset	Truncate vectors by removing or keeping beginning or ending values
Pad	Pad or truncate specified dimension(s)
Peak Finder	Determine whether each value of input signal is local minimum or maximum
Repeat	Resample input at higher rate by repeating values
Sample and Hold	Sample and hold input signal
Triggered Signal From Workspace	Import signal samples from MATLAB ® workspace when triggered
Unwrap	Unwrap signal phase
Upsample	Resample input at higher rate by inserting zeros
Variable Fractional Delay	Delay input by time-varying fractional number of sample periods

Variable Integer Delay	Delay input by time-varying integer number of sample periods
Window Function	Compute and/or apply window to input signal
Zero Crossing	Count number of times signal crosses zero in single time step

Signal Processing Sinks

Display	Show value of input
Matrix Viewer	Display matrices as color images
Signal To Workspace	Write simulation data to array in MATLAB workspace
Spectrum Scope	Compute and display periodogram of each input signal
Time Scope	Display signals generated during simulation
To Audio Device	Write audio data to computer's audio device
To Multimedia File	Write video frames and/or audio samples to multimedia Þle
To Wave File	Write audio data to Þle in Microsoft ® Wave (.wav) format
Triggered To Workspace	Write input sample to MATLAB workspace when triggered
Vector Scope	Display vector or matrix of time-domain, frequency-domain, or user-deÞned data
Waterfall	View vectors of data over time

Signal Processing Sources

Chirp	Generate swept-frequency cosine (chirp) signal
Constant	Generate constant value
Constant Diagonal Matrix	Gene rate square, diagonal matrix
Discrete Impulse	Generate discrete impulse
DSP Constant (Obsolete)	Generate discrete- or continuous-time constant signal
From Audio Device	Read audio data from computer's audio device
From Multimedia File	Read video frames and/or audio samples from multimedia Þle
From Wave File	Read audio data from Microsoft Wave (.wav) Þle
Identity Matrix	Generate matrix with ones on main diagonal and zeros elsewhere
Multiphase Clock	Generate multiple binary clock signals
N-Sample Enable	Output ones or zeros for speci Þed number of sample times
Random Source	Generate randomly distributed values
Signal From Workspace	Import signal from MATLAB workspace
Sine Wave	Generate continuous or discrete sine wave

Statistics

Autocorrelation	Compute autocorrelation of vector or matrix input
Correlation	Compute cross-correlation of two inputs
Detrend	Remove linear trend from vectors
Histogram	Generate histogram of input or sequence of inputs
Maximum	Find maximum values in input or sequence of inputs
Mean	Find mean value of input or sequence of inputs
Median	Find median value of input
Minimum	Find minimum values in input or sequence of inputs
RMS	Compute root-mean-square value of input or sequence of inputs
Sort	Sort input elements by value
Standard Deviation	Find standard deviation of input or sequence of inputs
Variance	Compute variance of input or sequence of inputs

Transforms

Analytic Signal	Compute analytic signals of discrete-time inputs
Complex Cepstrum	Compute complex cepstrum of input
DCT	Compute discrete cosine transform (DCT) of input

DWT	Compute discrete wavelet transform (DWT) of input
FFT	Compute fast Fourier transform (FFT) of input
IDCT	Compute inverse discrete cosine transform (IDCT) of input
IDWT	Compute inverse discrete wavelet transform (IDWT) of input
IFFT	Compute inverse fast Fourier transform (IFFT) of input
Inverse Short-Time FFT	Recover time-domain signals by performing inverse short-time, fast Fourier transform (FFT)
Magnitude FFT	Compute nonparametric estimate of spectrum using periodogram method
Real Cepstrum	Compute real cepstrum of input
Short-Time FFT	Compute nonparametric estimate of spectrum using short-time, fast Fourier transform (FFT) method

Blocks — Alphabetical List

Purpose

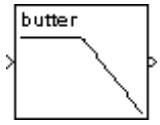
Design and implement analog filters

Library

Filtering / Filter Designs

dsparch4

Description



The Analog Filter Design block designs and implements a Butterworth, Chebyshev type I, Chebyshev type II, or elliptic filter in a highpass, lowpass, bandpass, or bandstop configuration.

The input must be a sample-based, continuous-time, real-valued, scalar signal.

The design and band configuration of the filter are selected from the Design method and Filter type pop-up menus in the dialog box. For each combination of design method and band configuration, an appropriate set of secondary parameters is displayed.

Filter Design	Description
Butterworth	The magnitude response of a Butterworth filter is maximally flat in the passband and monotonic overall.
Chebyshev type I	The magnitude response of a Chebyshev type I filter is equiripple in the passband and monotonic in the stopband.
Chebyshev type II	The magnitude response of a Chebyshev type II filter is monotonic in the passband and equiripple in the stopband.
Elliptic	The magnitude response of an elliptic filter is equiripple in both the passband and the stopband.

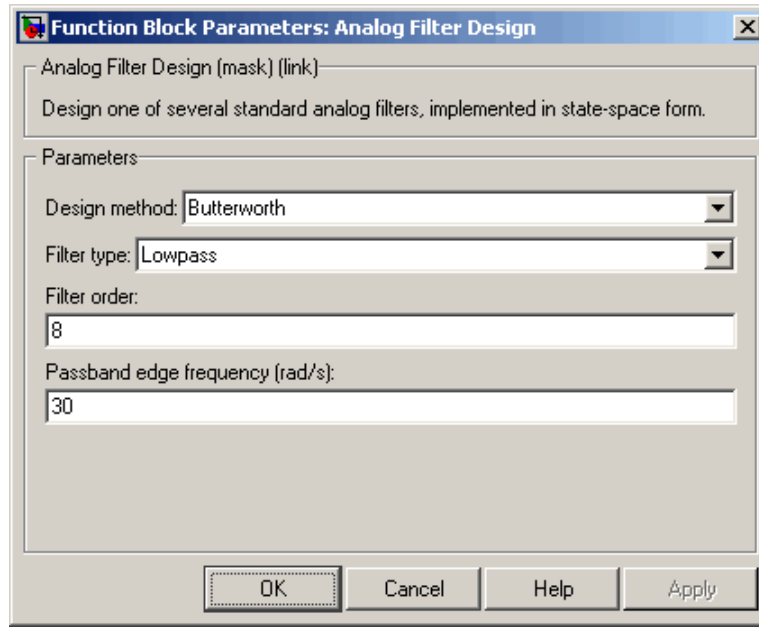
The following table lists the available parameters for each design/band combination. For lowpass and highpass band configurations, these parameters include the passband edge frequency Ω_p, the stopband edge frequency Ω_s, the passband ripple R_p, and the stopband attenuation R_s. For bandpass and bandstop configurations, the parameters include the

lower and upper passband edge frequencies, Ω_{p1} and Ω_{p2}, the lower and upper stopband edge frequencies, Ω_{s1} and Ω_{s2}, the passband ripple R_p, and the stopband attenuation R_s. Frequency values are in rad/s, and ripple and attenuation values are in dB.

	Lowpass	Highpass	Bandpass	Bandstop
Butterworth	Order, Ω_p	Order, Ω_p	Order, Ω_{p1}, Ω_{p2}	Order, Ω_{p1}, Ω_{p2}
Chebyshev Type I	Order, Ω_p, R_p	Order, Ω_p, R_p	Order, Ω_{p1}, Ω_{p2}, R_p	Order, Ω_{p1}, Ω_{p2}, R_p
Chebyshev Type II	Order, Ω_s, R_s	Order, Ω_s, R_s	Order, Ω_{s1}, Ω_{s2}, R_s	Order, Ω_{s1}, Ω_{s2}, R_s
Elliptic	Order, Ω_p, R_p, R_s	Order, Ω_p, R_p, R_s	Order, Ω_{p1}, Ω_{p2}, R_p, R_s	Order, Ω_{p1}, Ω_{p2}, R_p, R_s

The analog filters are designed using the filter design commands in Signal Processing Toolbox™ software's `buttap`, `cheb1ap`, `cheb2ap`, and `ellipap` functions, and are implemented in state-space form. Filters of order 8 or less are implemented in controller canonical form for improved efficiency.

Dialog Box



The parameters displayed in the dialog box vary for different design/band combinations. Only some of the parameters listed below are visible in the dialog box at any one time.

Design method

The filter design method: Butterworth, Chebyshev type I, Chebyshev type II, or Elliptic. Tunable.

Filter type

The type of filter to design: Lowpass, Highpass, Bandpass, or Bandstop. Tunable.

Filter order

The order of the filter, for lowpass and highpass configurations. For bandpass and bandstop configurations, the order of the final filter is twice this value.

Passband edge frequency
: The passband edge frequency, in rad/s, for the highpass and lowpass conÞgurations of the Butterworth, Chebyshev type I, and elliptic designs. Tunable.

Lower passband edge frequency
: The lower passband frequency, in rad/s, for the bandpass and bandstop conÞgurations of the Butterworth, Chebyshev type I, and elliptic designs. Tunable.

Upper passband edge frequency
: The upper passband frequency, in rad/s, for the bandpass and bandstop conÞgurations of the Butterworth, Chebyshev type I, or elliptic designs. Tunable.

Stopband edge frequency
: The stopband edge frequency, in rad/s, for the highpass and lowpass band conÞgurations of the Chebyshev type II design. Tunable.

Lower stopband edge frequency
: The lower stopband edge frequency, in rad/s, for the bandpass and bandstop conÞgurations of the Chebyshev type II design. Tunable.

Upper stopband edge frequency
: The upper stopband edge frequency, in rad/s, for the bandpass and bandstop Þlter conÞgurations of the Chebyshev type II design. Tunable.

Passband ripple in dB
: The passband ripple, in dB, for the Chebyshev Type I and elliptic designs. Tunable.

Stopband attenuation in dB
: The stopband attenuation, in dB, for the Chebyshev Type II and elliptic designs. Tunable.

References

Antoniou, A. Digital Filters: Analysis, Design, and Applications . 2nd ed. New York, NY: McGraw-Hill, 1993.

Supported Data Types

- Double-precision ßoating point

See Also

Digital Filter Design	Signal Processing Blockset
buttap	Signal Processing Toolbox
cheb1ap	Signal Processing Toolbox
cheb2ap	Signal Processing Toolbox
ellipap	Signal Processing Toolbox

See the following sections for related information:

- “Filters”
- “Analog Filter Design Block”

Purpose

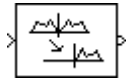
Compute analytic signals of discrete-time inputs

Library

Transforms

dspxfrm3

Description



The Analytic Signal block computes the complex analytic signal corresponding to each channel of the real M-by-N input, u

$$y = u + jH\{u\}$$

where $j = \sqrt{-1}$ and H{ } denotes the Hilbert transform. The real part of the output in each channel is a replica of the real input in that channel; the imaginary part is the Hilbert transform of the input. In the frequency domain, the analytic signal retains the positive frequency content of the original signal while zeroing-out negative frequencies and doubling the DC component.

The block computes the Hilbert transform using an equiripple FIR filter with the order specified by the Filter order parameter, n. The linear phase filter is designed using the Remez exchange algorithm, and imposes a delay of n/2 on the input samples.

The output has the same dimension and frame status as the input.

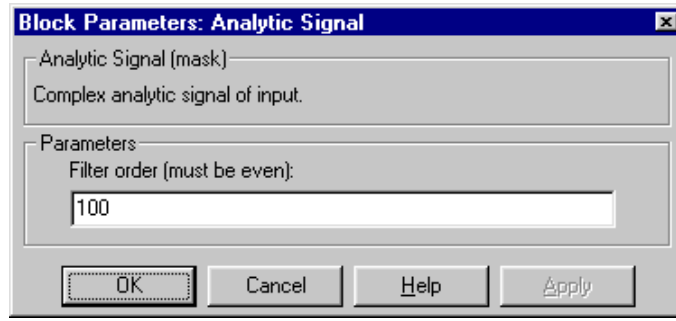
Sample-Based Operation

When the input is sample based, each of the M*N matrix elements represents an independent channel. Thus, the block computes the analytic signal for each channel (matrix element) over time.

Frame-Based Operation

When the input is frame based, each of the N columns in the matrix contains M sequential time samples from an independent channel, and the block computes the analytic signal for each channel over time.

Dialog Box



Filter order

The length of t he FIR Þlter used to compute the Hilbert transform.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

Purpose

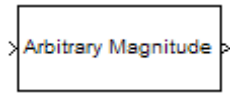
Design arbitrary response filter

Library

Filtering / Filter Design Toolbox

dspfdesign

Description



This block brings the functionality of the Fixed-Point Toolbox™ filterbuilder function to the Simulink® environment. You must have a Filter Design Toolbox™ product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See “Arbitrary Response Design Dialog Box — Main Pane” in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset™ product.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point

Port	Supported Data Types
	• Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Purpose

Add vector to array along specified dimension

Library

Math Functions / Matrices and Linear Algebra / Matrix Operations

dspmtrx3

Description

The Array-Vector Add block adds the values in the specified dimension of the N-dimensional input array A to the values in the input vector V.

The length of the input V must be the same as the length of the specified dimension of A. The Array-Vector Add block adds each element of V to the corresponding element along that dimension of A.

Consider a 3-dimensional M-by-N-by-P input array A(i,j,k) and a P-by-1 input vector V. When the Add along dimension parameter is set to 2, the output of the block Y(i,j,k) is

$$Y(i,j,k) = A(i,j,k) + V(j)$$

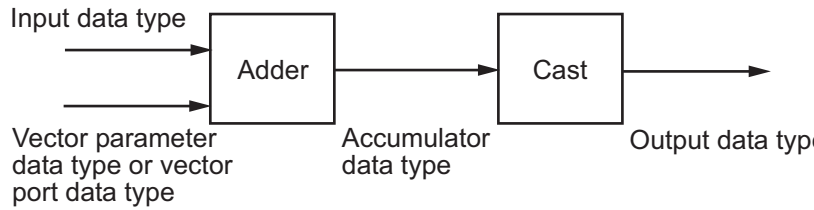
where

$$1 \le i \le M$$
$$1 \le j \le N$$
$$1 \le k \le P$$

The output of the Array-Vector Add block is the same size as the input array, A. When both inputs are sample based, the output is sample based; otherwise, the output is frame based. This block accepts real and complex floating-point and fixed-point inputs.

Fixed-Point Data Types

The following diagram shows the data types used within the Array-Vector Add block for fixed-point signals.

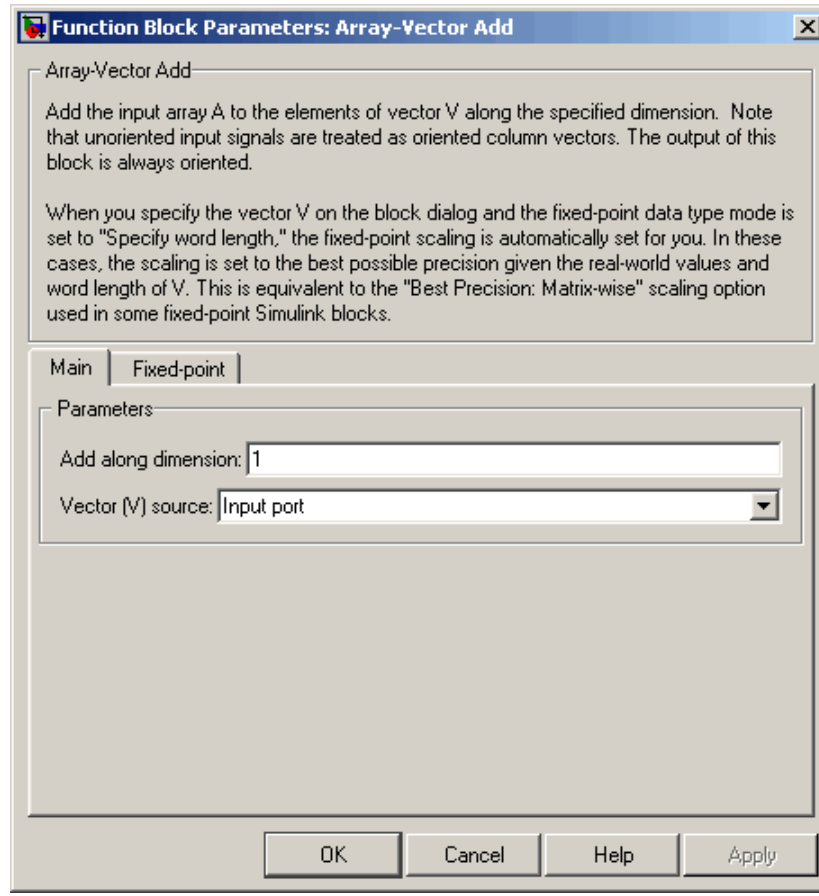


When the vector V is designated in the block mask, its elements have the data type and scaling th at you specify in the Vector (V) parameters on the Fixed-point tab. When the vector comes in through the block port, its elements inherit their data type and scaling from the driving block.

You can set the vector, accumulator, and output data types in the block dialog as discussed below.

Dialog Box

The Main pane of the Array-Vector Add block dialog appears as follows.



Add along dimension

Specify the dimension along which to add the input array A to the elements of vector V.

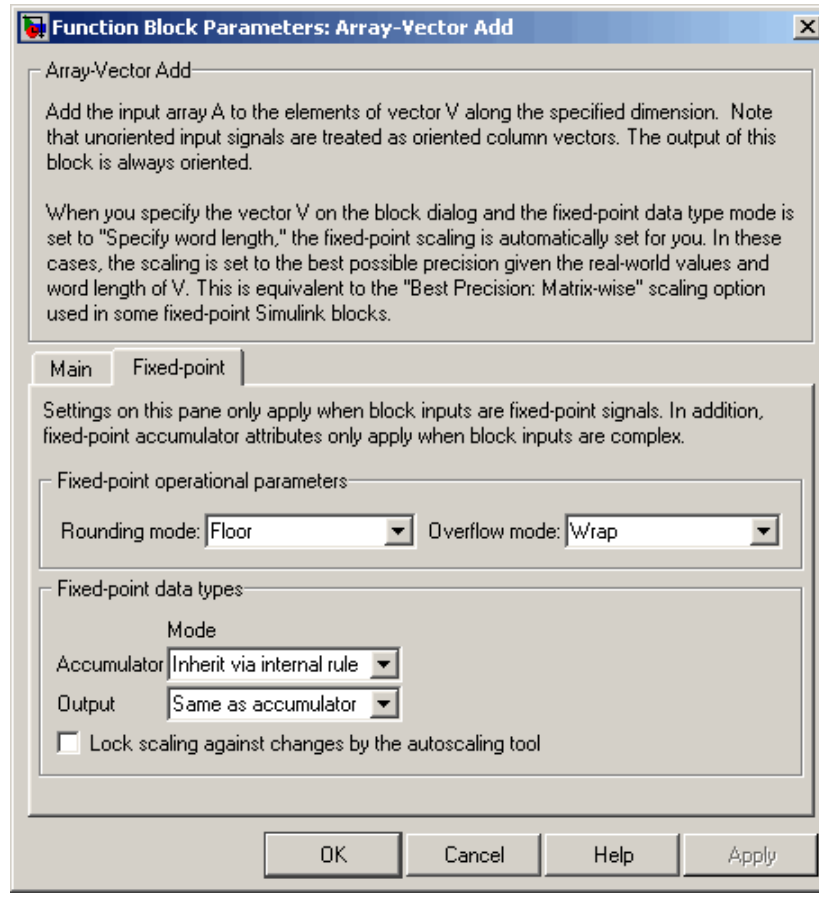
Vector (V) source

Specify the source of the vector, V. The vector can come from the Input port or from a Dialog parameter.

Vector (V)

Specify the vector, V. This parameter is visible only when you select Dialog parameter for the Vector (V) source parameter.

The Fixed-point pane of the Array-Vector Add block dialog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Vector (V)

Use this parameter to specify how you would like to designate the word and fraction lengths of the elements of the vector, V:

- When you select Same word length as input , the word length of the vector values match that of the input to the block.
- When you select Specify word length , you can enter the word length of the vector values, in bits. In this mode, the fraction length of the vector values is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the values.
- When you select Binary point scaling , you can enter the word length and the fraction length of the vector elements, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the sl ope of the vector element. This block requires power-of-two slope and a bias of zero.

Note The Vector (V) parameters on the Fixed-point pane are only applicable when you spe cify the vector through the Vector (V) parameter on the Main pane of the block mask. When the vector comes in through the block port, the data type and scaling of its elements are inherited from the driving block.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths. See “Fixed-Point Data Types” on page 2-11 for an illustration depicting the use of the accumulator data type in this block.

- When you select Inherit via internal rule , the accumulator word length and fraction length are calculated automatically. This block uses the same internal rule as the Simulink Sum block.
- When you select Same as first input , these characteristics match those of the Þrst input to the block.

- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as accumulator , these characteristics match those of the accumulator.
- When you select Same as first input , these characteristics match those of the Þrst input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any Þxed-point scaling you specify in this block mask f rom being overridden by the autoscaling tool in the Fixed-Point Tool.

Supported Data Types

Port	Supported Data Types
A	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

Port	Supported Data Types
V	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

See Also

Array-Vector Divide	Signal Processing Blockset
Array-Vector Multiply	Signal Processing Blockset
Array-Vector Subtract	Signal Processing Blockset

Purpose

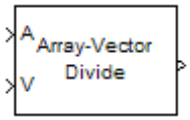
Divide array by vector along speci Þed dimension

Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

dspmtrx3

Description



The Array-Vector Divide block divides the values in the speci Þed dimension of the N-dimensional input array A by the values in the input vector V.

The length of the input V must be the same as the length of the speci Þed dimension of A. The Array-Vector Divide block divides each element of V by the corresponding element along that dimension of A.

Consider a 3-dimensional M-by-N-by-P input array A(i,j,k) and a P-by-1 input vector V. When the Divide along dimension parameter is set to 2, the output of the block Y(i,j,k) is

$$Y(i,j,k) = \frac{A(i,j,k)}{V(j)}$$

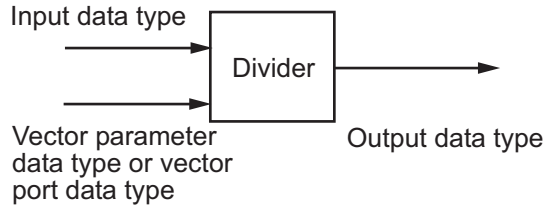
where

$$1 \le i \le M$$
$$1 \le j \le N$$
$$1 \le k \le P$$

The output of the Array-Vector Divide block is the same size as the input array, A. When both inputs are sample based, the output is sample based; otherwise, the output is frame based. This block accepts real and complex ßoating-point and Þxed-point input arrays, and real ßoating-point and Þxed-point input vectors.

Fixed-Point Data Types

The following diagram shows the data types used within the Array-Vector Divide block for Þxed-point signals.

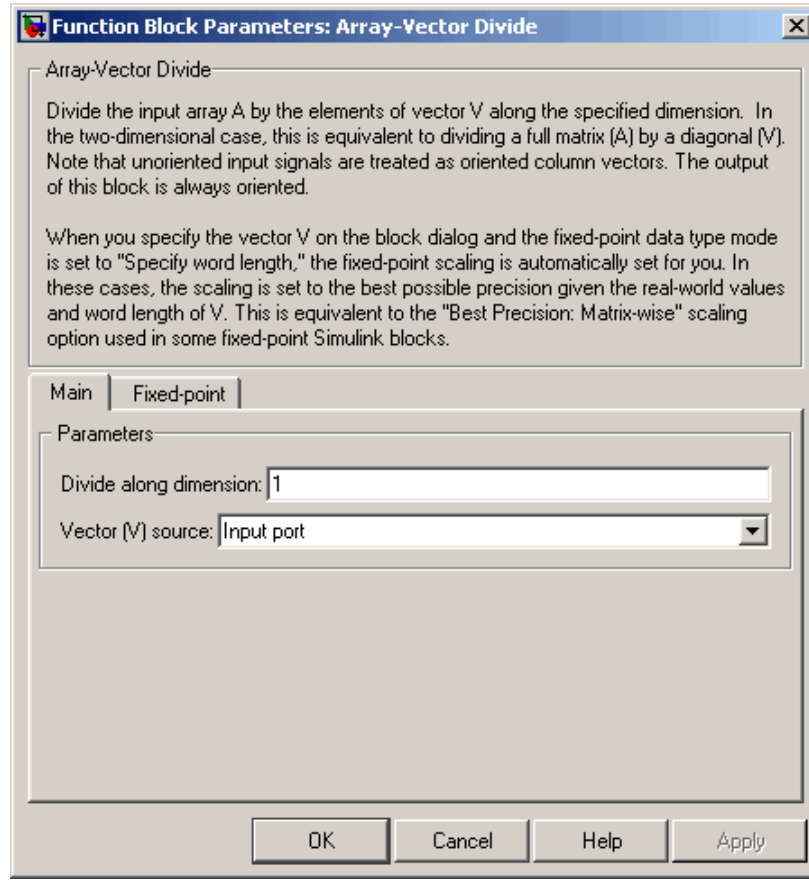


When the vector V is designated in the block mask, its elements have the data type and scaling th at you specify in the Vector (V) parameters on the Fixed-point tab. When the vector comes in through the block port, its elements inherit their data type and scaling from the driving block.

You can set the vector and output data types in the block dialog as discussed below.

Dialog Box

The Main pane of the Array-Vector Divide block dialog appears as follows.



Divide along dimension

Specify the dimension along which to divide the input array A by the elements of vector V.

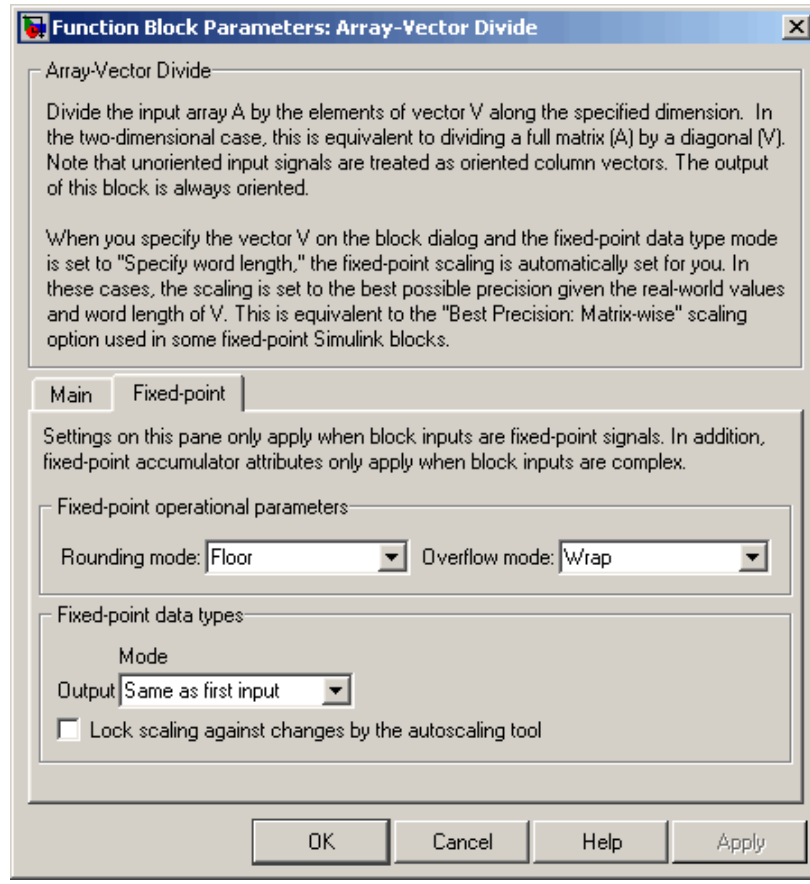
Vector (V) source

Specify the source of the vector, V. The vector can come from the Input port or from a Dialog parameter.

Vector (V)

Specify the vector, V. This parameter is visible only when you select Dialog parameter for the Vector (V) source parameter.

The Fixed-point pane of the Array-Vector Divide block dialog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Vector (V)

Use this parameter to specify how you would like to designate the word and fraction lengths of the elements of the vector, V:

- When you select Same word length as input , the word length of the vector values match that of the input to the block.
- When you select Specify word length , you can enter the word length of the vector values, in bits. In this mode, the fraction length of the vector values is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the values.
- When you select Binary point scaling , you can enter the word length and the fraction length of the vector elements, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the sl ope of the vector element. This block requires power-of-two slope and a bias of zero.

Note The Vector (V) parameters on the Fixed-point pane are only applicable when you spe cify the vector through the Vector (V) parameter on the Main pane of the block mask. When the vector comes in through the block port, the data type and scaling of its elements are inherited from the driving block.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as first input , these characteristics match those of the Þrst input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool.

Supported Data Types

Port	Supported Data Types
A	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
V	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

See Also

Array-Vector Add	Signal Processing Blockset
Array-Vector Multiply	Signal Processing Blockset
Array-Vector Subtract	Signal Processing Blockset

Array-Vector Multiply

Purpose

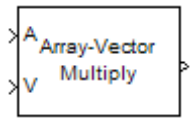
Multiply array by vector along speci Þed dimension

Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

dspmtrx3

Description



The Array-Vector Multiply block m ultiplies the values in the speci Þed dimension of the N-dimensional input array A by the values in the input vector V.

The length of the input V must be the same as the length of the speci Þed dimension of A. The Array-Vector Multiply block multiplies each element of V by the corresponding element along that dimension of A.

Consider a 3-dimensional M-by-N-by-P input array A(i,j,k) and a P-by-1 input vector V. When the Multiply along dimension parameter is set to 2, the output of the block Y(i,j,k) is

$$Y(i,j,k) = A(i,j,k) * V(j)$$

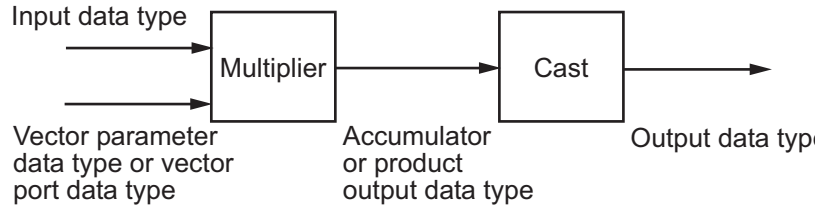
where

$$1 \le i \le M$$
$$1 \le j \le N$$
$$1 \le k \le P$$

The output of the Array-Vector Multiply block is the same size as the input array, A. When both inputs are sample based, the output is sample based; otherwise, the output is frame based. This block accepts real and complex ßoating-point and Þxed-point inputs.

Fixed-Point Data Types

The following diagram shows the data types used within the Array-Vector Multiply block for Þxed-point signals.



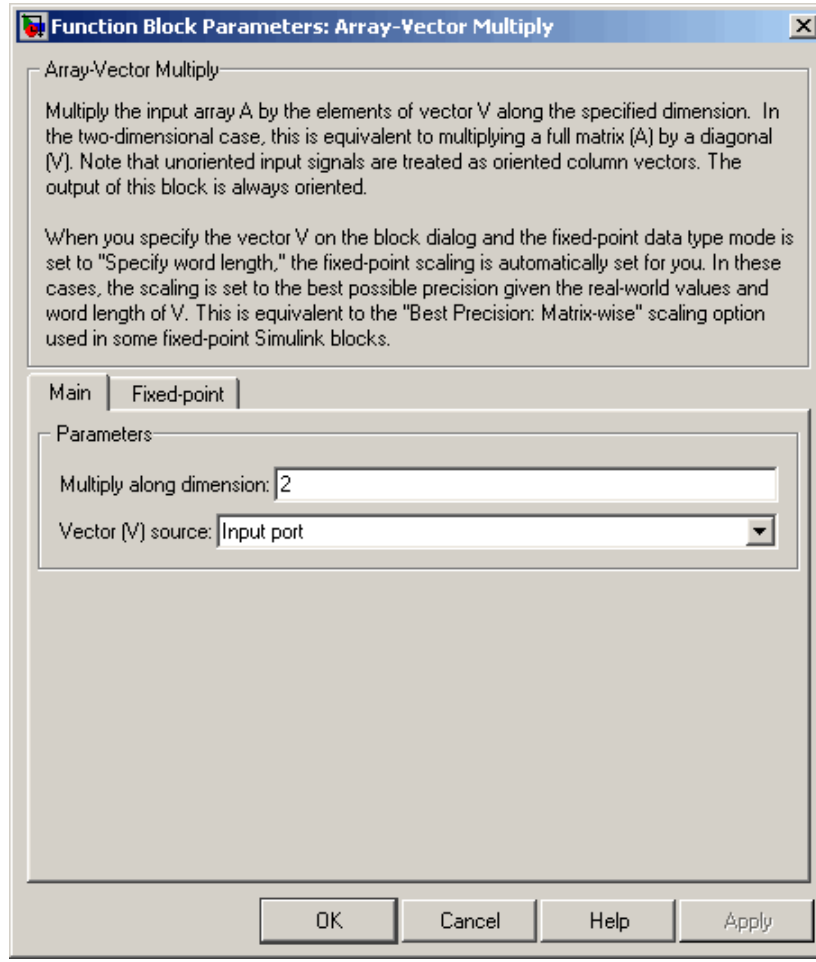
When the vector V is designated in the block mask, its elements have the data type and scaling th at you specify in the Vector (V) parameters on the Fixed-point tab. When the vector comes in through the block port, its elements inherit their data type and scaling from the driving block.

The output of the multiplier is in the product output data type when at least one of the inputs to the multiplier is real. When both of the inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For det ails on the complex multiplication performed, see “Multiplication Data Types”.

You can set the vector, accumulator, product output, and output data types in the block dialog as discussed below.

Dialog Box

The Main pane of the Array-Vector Multiply block dialog appears as follows.



Multiply along dimension

Specify the dimension along which to multiply the input array A by the elements of vector V.

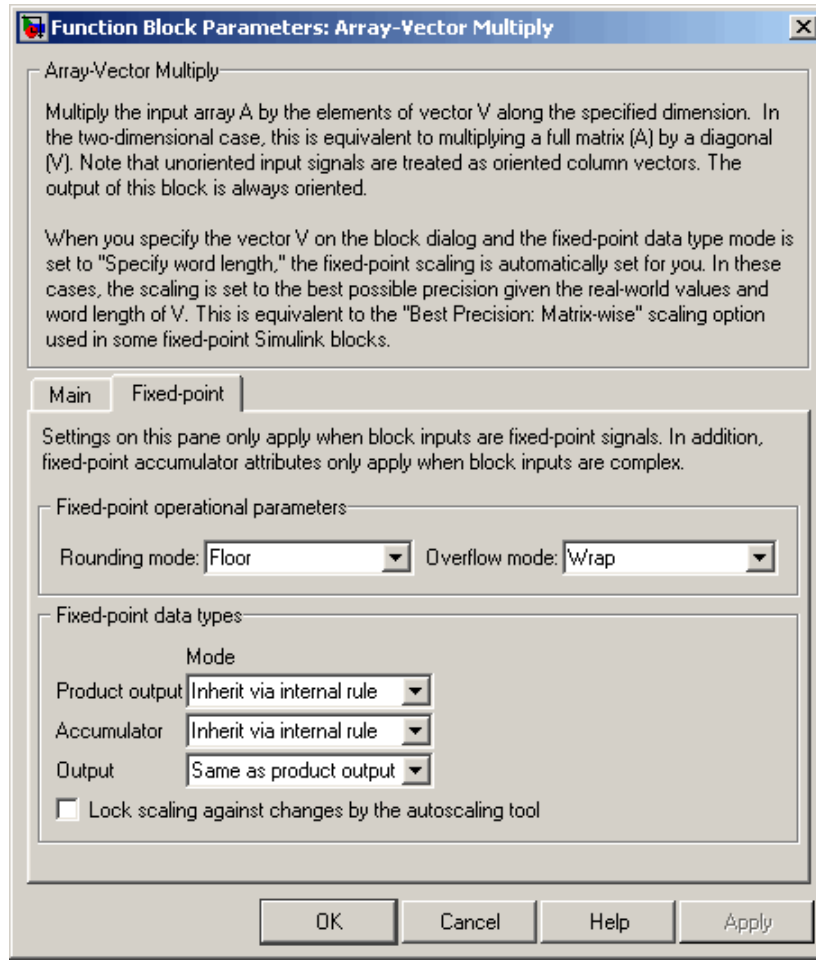
Vector (V) source

Specify the source of the vector, V. The vector can come from the Input port or from a Dialog parameter.

Vector (V)

Specify the vector, V. This parameter is visible only when you select Dialog parameter for the Vector (V) source parameter.

The Fixed-point pane of the Array-Vector Multiply block dialog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Vector (V)

Use this parameter to specify how you would like to designate the word and fraction lengths of the elements of the vector, V:

- When you select Same word length as input, the word length of the vector values match that of the input to the block.
- When you select Specify word length, you can enter the word length of the vector values, in bits. In this mode, the fraction length of the vector values is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the values.
- When you select Binary point scaling, you can enter the word length and the fraction length of the vector elements, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the vector element. This block requires power-of-two slope and a bias of zero.

Note The Vector (V) parameters on the Fixed-point pane are only applicable when you specify the vector through the Vector (V) parameter on the Main pane of the block mask. When the vector comes in through the block port, the data type and scaling of its elements are inherited from the driving block.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-26 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Inherit via internal rule, the product output word length and fraction length are calculated automatically. For information about how the product output

word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.

- When you select Same as first input, these characteristics match those of the first input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths. See “Fixed-Point Data Types” on page 2-26 for an illustration depicting the use of the accumulator data type in this block. Note that the accumulator data type is only used when both inputs to the multiplier are complex:

- When you select Inherit via internal rule, the accumulator word length and fraction length are calculated automatically. For information about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as first input, these characteristics match those of the first input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as product output , these characteristics match those of the product.
- When you select Same as first input , these characteristics match those of the Þrst input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any Þxed-point scaling you specify in this block mask f rom being overridden by the autoscaling tool in the Fixed-Point Tool.

Supported Data Types

Port	Supported Data Types
A	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
V	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

See Also

Array-Vector Add	Signal Processing Blockset
Array-Vector Divide	Signal Processing Blockset
Array-Vector Subtract	Signal Processing Blockset

Purpose

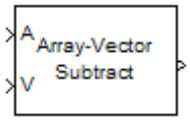
Subtract vector from array along specified dimension

Library

Math Functions / Matrices and Linear Algebra / Matrix Operations

dspmtrx3

Description



The Array-Vector Subtract block subtracts the values in the input vector V from the values in the specified dimension of the N-dimensional input array A.

The length of the input V must be the same as the length of the specified dimension of A. The Array-Vector Subtract block subtracts each element of V from the corresponding element along that dimension of A.

Consider a 3-dimensional M-by-N-by-P input array A(i,j,k) and a P-by-1 input vector V. When the Subtract along dimension parameter is set to 2, the output of the block Y(i,j,k) is

$$Y(i,j,k) = A(i,j,k) - V(j)$$

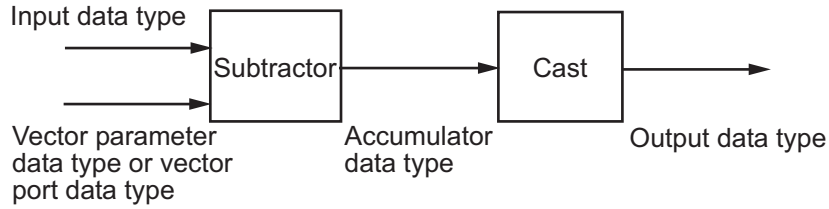
where

$$1 \le i \le M$$
$$1 \le j \le N$$
$$1 \le k \le P$$

The output of the Array-Vector Subtract block is the same size as the input array, A. When both inputs are sample based, the output is sample based; otherwise, the output is frame based. This block accepts real and complex floating-point and fixed-point inputs.

Fixed-Point Data Types

The following diagram shows the data types used within the Array-Vector Subtract block for fixed-point signals.



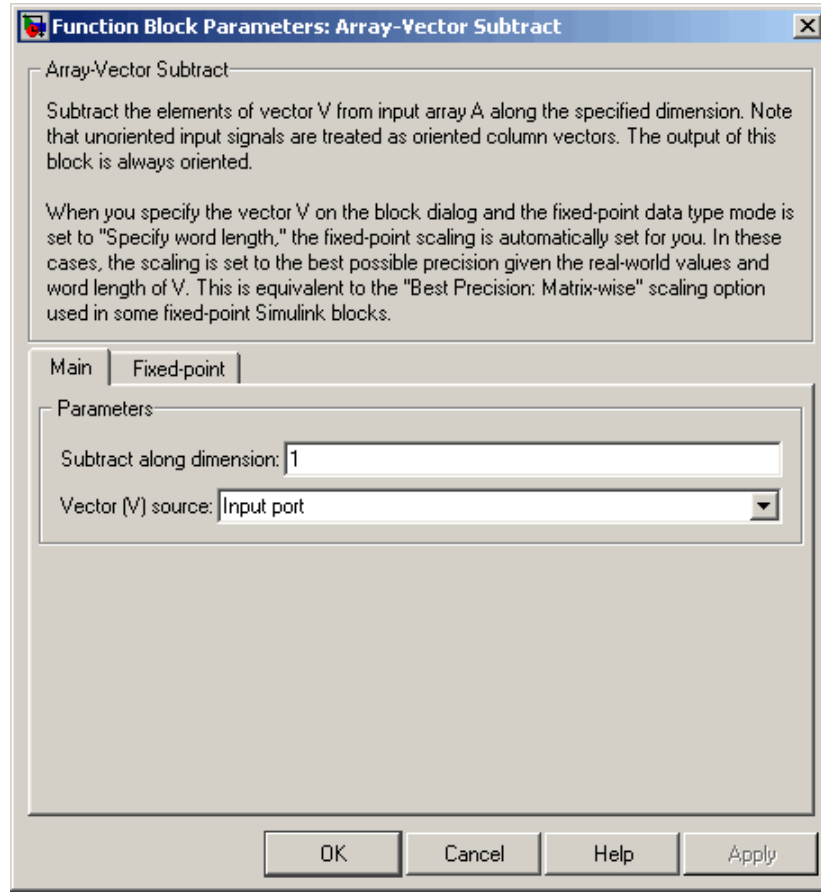
When the vector V is designated in the block mask, its elements have the data type and scaling th at you specify in the Vector (V) parameters on the Fixed-point tab. When the vector comes in through the block port, its elements inherit their data type and scaling from the driving block.

The output of the subtractor is in the accumulator data type.

You can set the vector, accumulator, and output data types in the block dialog as discussed below.

Dialog Box

The Main pane of the Array-Vector Subtract block dialog appears as follows.



Subtract along dimension

Specify the dimension along which to subtract the elements of vector V from the input array A.

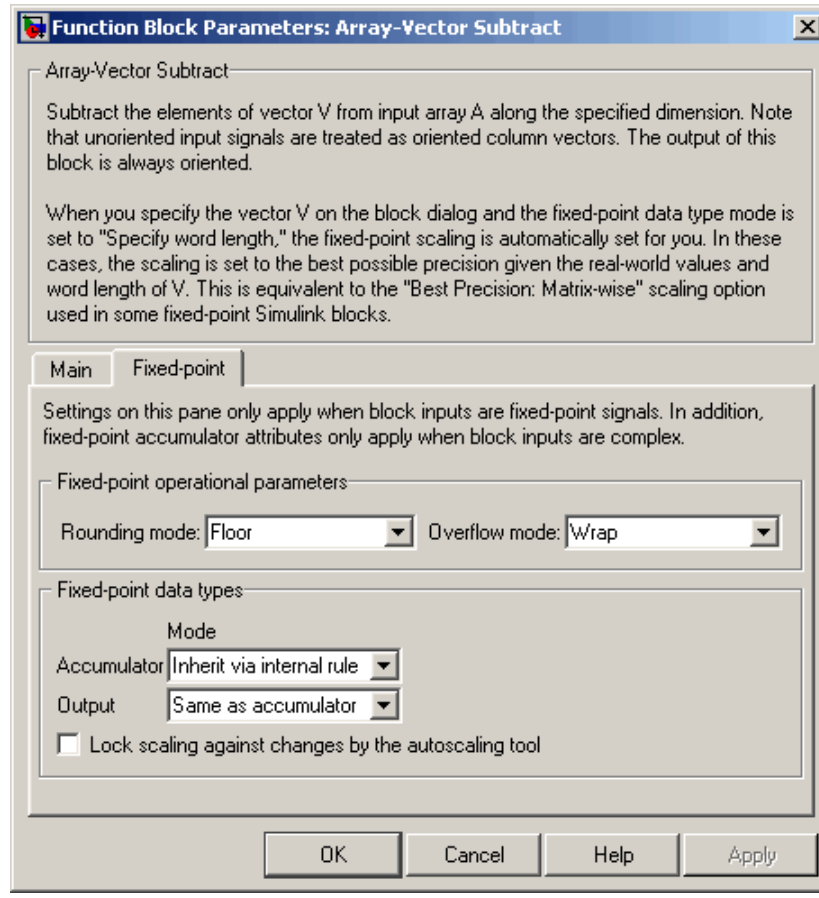
Vector (V) source

Specify the source of the vector, V. The vector can come from the Input port or from a Dialog parameter.

Vector (V)

Specify the vector, V. This parameter is visible only when you select Dialog parameter for the Vector (V) source parameter.

The Fixed-point pane of the Array-Vector Subtract block dialog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Vector (V)

Use this parameter to specify how you would like to designate the word and fraction lengths of the elements of the vector, V:

- When you select Same word length as input , the word length of the vector values match that of the input to the block.
- When you select Specify word length , you can enter the word length of the vector values, in bits. In this mode, the fraction length of the vector values is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the values.
- When you select Binary point scaling , you can enter the word length and the fraction length of the vector elements, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the sl ope of the vector element. This block requires power-of-two slope and a bias of zero.

Note The Vector (V) parameters on the Fixed-point pane are only applicable when you spe cify the vector through the Vector (V) parameter on the Main pane of the block mask. When the vector comes in through the block port, the data type and scaling of its elements are inherited from the driving block.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths. See “Fixed-Point Data Types” on page 2-35 for an illustration depicting the use of the accumulator data type in this block.

- When you select Inherit via internal rule , the accumulator word length and fraction length are calculated automatically. This block uses the same internal rule as the Simulink Sum block.
- When you select Same as first input , these characteristics match those of the Þrst input to the block.

- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as accumulator , these characteristics match those of the accumulator.
- When you select Same as first input , these characteristics match those of the Þrst input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any Þxed-point scaling you specify in this block mask f rom being overridden by the autoscaling tool in the Fixed-Point Tool.

Supported Data Types

Port	Supported Data Types
A	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

Port	Supported Data Types
V	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

See Also

Array-Vector Add	Signal Processing Blockset
Array-Vector Divide	Signal Processing Blockset
Array-Vector Multiply	Signal Processing Blockset

Purpose

Compute autocorrelation of vector or matrix input

Library

Statistics

dspstat3

Description

The Autocorrelation block computes the autocorrelation along each column of a frame-based input, and computes along the first dimension of an N-D sample-based input. The output of the block is always sample-based.

The Autocorrelation block accepts real and complex floating-point and fixed-point inputs except for complex unsigned fixed-point inputs. Fixed-point signals are not supported for the frequency domain.

Autocorrelation of Frame-Based Inputs

When the input to the Autocorrelation block is a frame-based M-by-N matrix u, the output, y, is a sample-based (l+1)-by-N matrix whose jth column has elements

$$y_{i,j} = \sum_{k=0}^{M-1} u_{k,j}^{*} u_{(k+i),j} \qquad 0 \le i \le l$$

where * denotes the complex conjugate, and l represents the maximum lag. $y_{0,j}$ is the zero-lag element in the jth column. When you select Compute all non-negative lags, l=M–1. Otherwise, l is specified as a nonnegative integer by the Maximum non-negative lag (less than input length) parameter.

Input u is zero when indexed outside of its valid range. When the input is real, the output is real; otherwise, the output is complex.

Autocorrelation of Sample-Based Inputs

When the input is a sample-based N-D array, the block computes the autocorrelation along the first dimension of the input. The output is a sample based N-D array, where the size of the first dimension is l+1, and the sizes of all other dimensions match those of the input array. For

example, when the input is an M-by-N-by-P array, the Autocorrelation block outputs an (l+1)-by-N-by-P sample-based array.

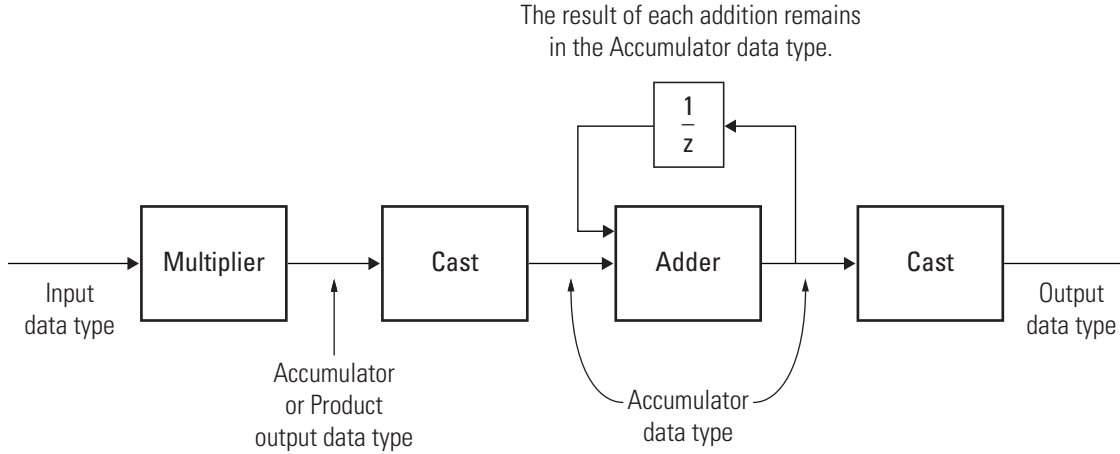
When you select Compute all non-negative lags, l=M–1. Otherwise, l is specified as a nonnegative integer by the Maximum non-negative lag (less than input length) parameter.

Input u is zero when indexed outside of its valid range. When the input is real, the output is real; otherwise, the output is complex.

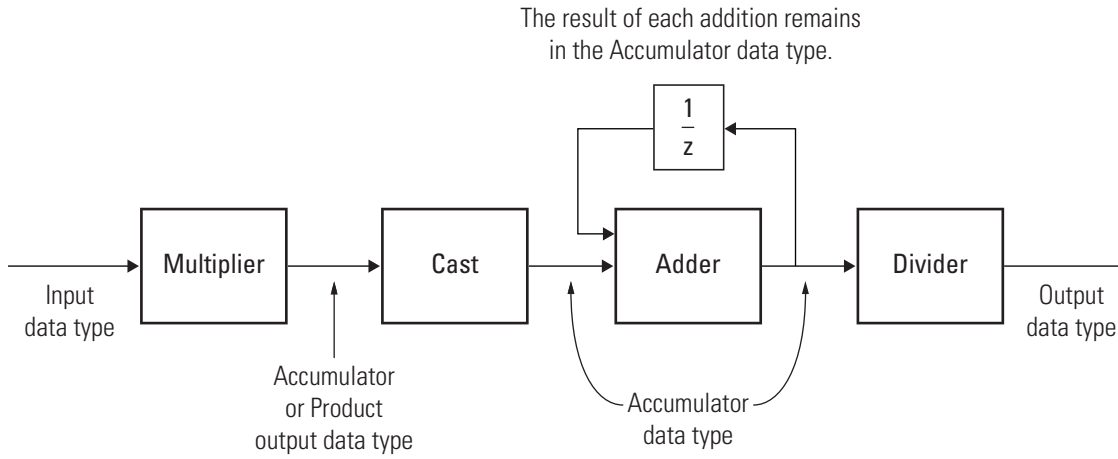
Fixed-Point Data Types

The following diagrams show the data types used within the Autocorrelation block for fixed-point signals (time domain only).

Signal flow when Scaling is "None"



Signal flow when Scaling is other than "None"

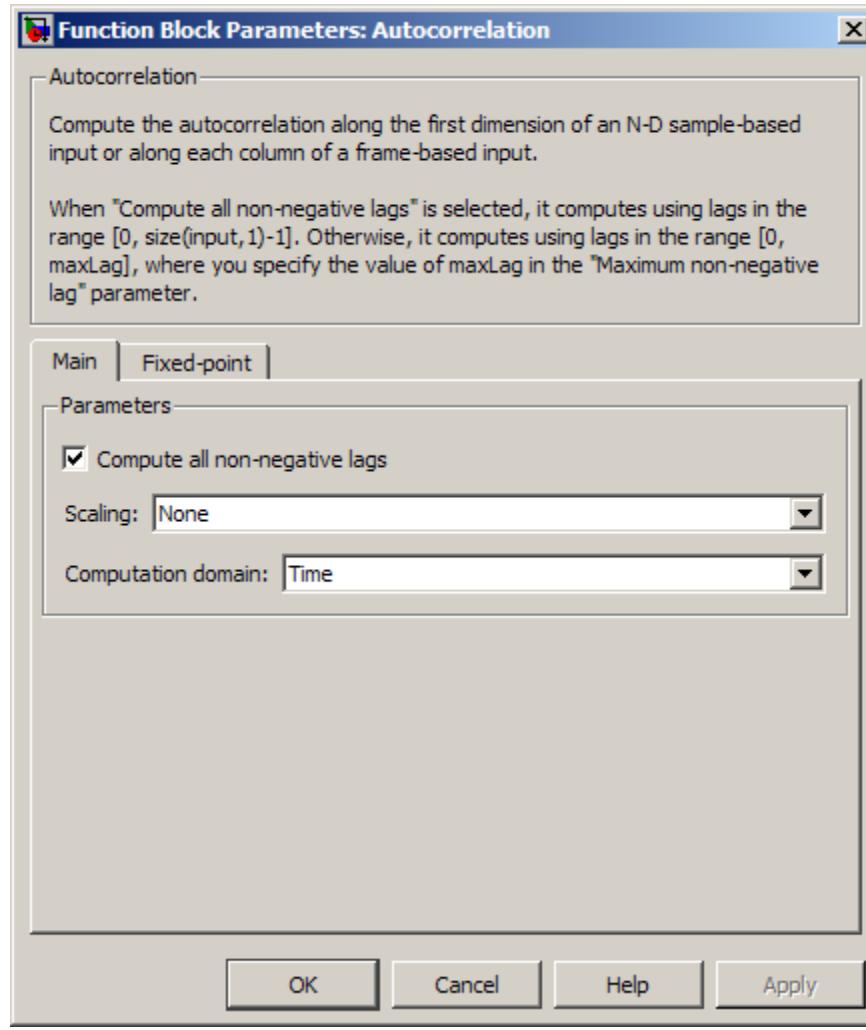


You can set the product output, accumulator, and output data types in the Fixed-Point pane of the block dialog as discussed in the next section.

The output of the multiplier is in the product output data type when the input is real. When the input is complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see "Multiplication Data Types".

Dialog Box

The Main pane of the Autocorrelation block dialog appears as follows.



Compute all non-negative lags

Select to compute the autocorrela tion over all nonnegative lags in the range [0, length(input)-1].

Maximum non-negative lag (less than input length)

Specify the maximum positive lag, l, for the autocorrelation. This parameter is enabled when you do not select the Compute all non-negative lags check box.

Scaling

This parameter controls the scaling that is applied to the output. The following options are available:

- None — Generates the raw autocorrelation $y_{i,j}$ without normalization.
- Biased — Generates the biased estimate of the autocorrelation.

$$y_{i,j}^{biased} = \frac{y_{i,j}}{M}$$

- Unbiased — Generates the unbiased estimate of the autocorrelation.

$$y_{i,j}^{unbiased} = \frac{y_{i,j}}{M - i}$$

- Unity at zero-lag — Normalizes the estimate of the autocorrelation for each channel so that the zero-lag sum is identically 1.

$$y_{0,j} = 1$$

This parameter is tunable, except in the Simulink external mode.

Computation domain

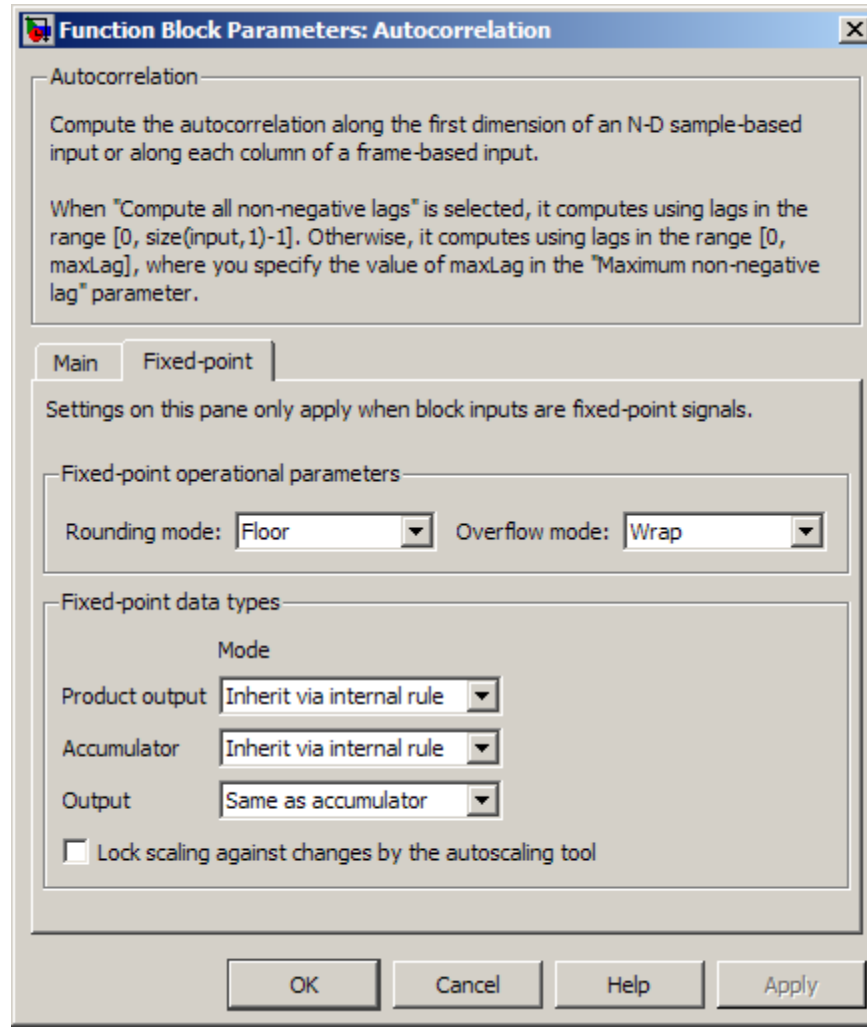
This parameter sets the domain in which the block computes convolutions to one of the following settings:

- Time — Computes in the time domain, which minimizes memory use
- Frequency — Computes in the frequency domain, which might require fewer computations than computing in the time domain, depending on the input length

---

Note This parameter must be set to Time for fixed-point signals.

The Fixed-point pane of the Autocorrelation block dialog appears as follows.


Function Block Parameters: Autocorrelation
Autocorrelation
Compute the autocorrelation along the first dimension of an N-D sample-based input or along each column of a frame-based input.
When "Compute all non-negative lags" is selected, it computes using lags in the range [0, size(input,1)-1]. Otherwise, it computes using lags in the range [0, maxLag], where you specify the value of maxLag in the "Maximum non-negative lag" parameter.
Main
Fixed-point
Settings on this pane only apply when block inputs are fixed-point signals.
Fixed-point operational parameters
Rounding mode:
Floor
Overflow mode:
Wrap
Fixed-point data types
Mode
Product output
Inherit via internal rule
Accumulator
Inherit via internal rule
Output
Same as accumulator
Lock scaling against changes by the autoscaling tool
OK
Cancel
Help
Apply

Note Fixed-point signals are only supported for the time domain. To use the parameters on this pane, make sure Time is selected for the Computation domain parameter on the Main pane.

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

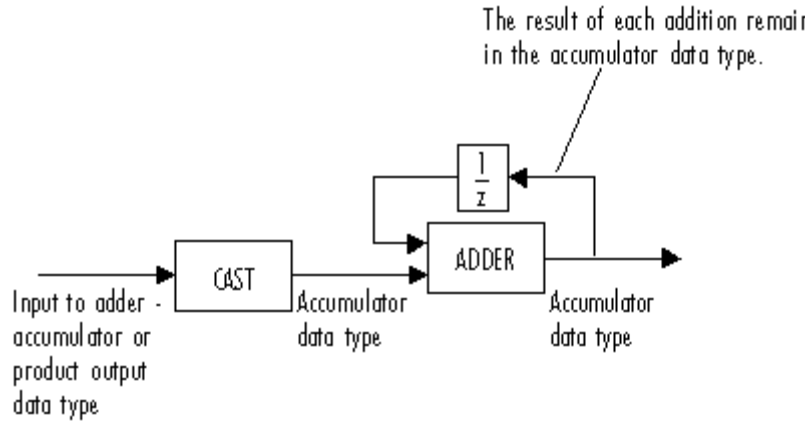
Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how you want to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-44 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Inherit via internal rule, the product output word length and fraction length are calculated automatically. For information about how the product output word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as input, these characteristics match those of the block input.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator



As depicted in this figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how you want to designate this accumulator word and fraction lengths.

You also use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block. See “Multiplication Data Types” for more information.

- When you select Inherit via internal rule, the accumulator word length and fraction length are calculated automatically. For information on about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as product output, these characteristics match those of the product output.
- When you select Same as input, these characteristics match those of the block input.

- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as accumulator , these characteristics match those of the accumulator.

  A special case occurs when Inherit via internal rule is specified for Accumulator , and the block input is complex. In that case, the output word length is one less than the accumulator word length.
- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Correlation	Signal Processing Blockset
xcorr	Signal Processing Toolbox

Autocorrelation LPC

Purpose

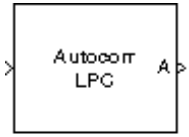
Determine coefficients of Nth-order forward linear predictors

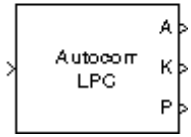
Library

Estimation / Linear Prediction

dsplp

Description





The Autocorrelation LPC block determines the coefficients of an N-step forward linear predictor for the time-series in each length-M input channel, u, by minimizing the prediction error in the least squares sense. A linear predictor is an FIR filter that predicts the next value in a sequence from the present and past inputs. This technique has applications in filter design, speech coding, spectral analysis, and system identification.

The Autocorrelation LPC block can output the prediction error for each channel as polynomial coefficients, reflection coefficients, or both. It can also output the prediction error power for each channel. The input u can be a scalar, 1-D vector, frame- or sample-based column vector, a sample-based row vector, or a channel-based matrix. Frame-based row vectors are not valid inputs. Frame-based matrices of size M-by-N are treated as N channels of length M.

When you select Inherit prediction order from input dimensions, the prediction order, N, is inherited from the input dimensions. Otherwise, you can use the Prediction order parameter to specify the value of N. Note that N must be a scalar with a value less than the length of the input channels or the block produces an error.

When Output(s) is set to A, port A is enabled. For each channel, port A outputs an (N+1)-by-1 column vector, $a = [1\ a_2\ a_3\ \ldots\ a_{N+1}]^T$, containing the coefficients of an Nth-order moving average (MA) linear process that predicts the next value, $\hat{u}_{M+1}$, in the input time-series.

$$\hat{u}_{M+1} = -(a_2 u_M) - (a_3 u_{M-1}) - \ldots - (a_{N+1} u_{M-N+1})$$

When Output(s) is set to K, port K is enabled. For each channel, port K outputs a length-N column vector whose elements are the prediction error reflection coefficients. When Output(s) is set to A and K, both

port A and K are enabled, and each p ort outputs its respective set of prediction coef Þcients for each channel.

When you select Output prediction error power (P) , port P is enabled. The prediction error pow er is output at port P as a vector whose length is the number of input channels.

Algorithm

The Autocorrelation LPC block compu tes the least squares solution to

$$\min_{\tilde{a}\in\Re^n} \|U\tilde{a} - b\|$$

where $\|\cdot\|$ indicates the 2-norm and

$$U = \begin{bmatrix} u_1 & 0 & \cdots & 0 \\ u_2 & u_1 & \ddots & \vdots \\ \vdots & u_2 & \ddots & 0 \\ \vdots & \vdots & \ddots & u_1 \\ \vdots & \vdots & \vdots & u_2 \\ \vdots & \vdots & \vdots & \vdots \\ u_M & \vdots & \vdots & \vdots \\ 0 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & u_M \end{bmatrix}, \tilde{a} = \begin{bmatrix} a_2 \\ \vdots \\ a_n + 1 \end{bmatrix}, b = \begin{bmatrix} u_2 \\ u_3 \\ \vdots \\ u_M \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Solving the least squares problem via the normal equations

$$U^*U\tilde{a} = U^*b$$

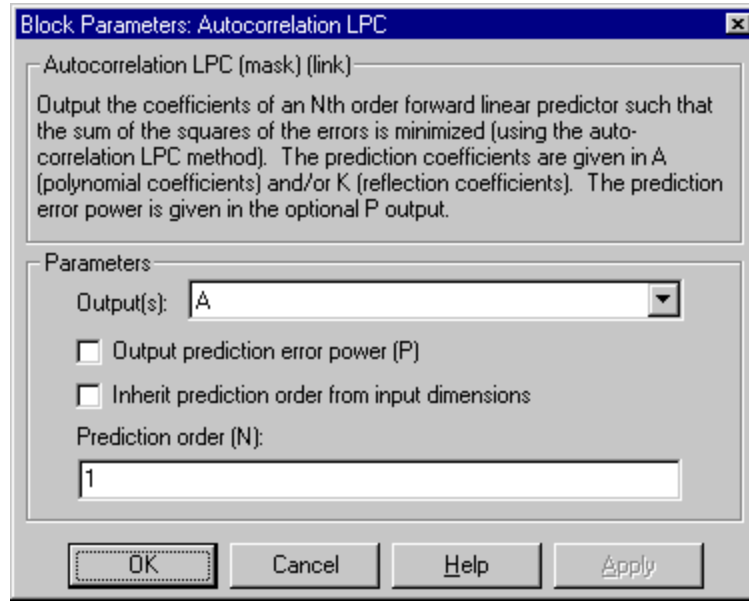
leads to the system of equations

$$\begin{bmatrix} r_1 & r_2^* & \cdots & r_n^* \\ r_2 & r_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_2^* \\ r_n & \cdots & r_2 & r_1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_3 \\ \vdots \\ a_{n+1} \end{bmatrix} = \begin{bmatrix} -r_2 \\ -r_3 \\ \vdots \\ -r_{n+1} \end{bmatrix}$$

where $r = [r_1\ r_2\ r_3\ \ldots\ r_{n+1}]^T$ is an autocorrelation estimate for u computed using the Autocorrelation block, an d * indicates the complex conjugate transpose. The normal equations are solved in $O(n^2)$ operations by the Levinson-Durbin block.

Note that the solution to the LPC problem is very closely related to the Yule-Walker AR method of spectr al estimation. In that context, the normal equations above are referred to as the Yule-Walker AR equations.

Dialog Box



Output(s)
: The type of prediction coefficients output by the block. The block can output polynomial coefficients (A), reflection coefficients (K), or both (A and K).

Output prediction error power (P)
: When selected, enables port P, which outputs the output prediction error power.

Inherit prediction order from input dimensions
: When selected, the block inherits the prediction order from the input dimensions.

Prediction order (N)
: Specify the prediction order, N, which must be a scalar. This parameter is disabled when you select the Inherit prediction order from input dimensions parameter.

References

Haykin, S. Adaptive Filter Theory. 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1996.

Ljung, L. System Identification: Theory for the User. Englewood Cliffs, NJ: Prentice Hall, 1987. Pgs. 278-280.

Proakis, J. and D. Manolakis. Digital Signal Processing. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

Autocorrelation	Signal Processing Blockset
Levinson-Durbin	Signal Processing Blockset
Yule-Walker Method	Signal Processing Blockset
lpc	Signal Processing Toolbox

Backward Substitution

Purpose

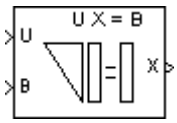
Solve UX=B for X when U is upper triangular matrix

Library

Math Functions / Matrices and Linear Algebra / Linear System Solvers

dspsolvers

Description



The Backward Substitution block solves the linear system UX=B by simple backward substitution of variables, where:

- U is the upper triangular M-by-M matrix input to the U port.
- B is the M-by-N matrix input to the B port.

The sample-based output is the M-by-N matrix X that is the solution of the equations. The block does not check the rank of the inputs.

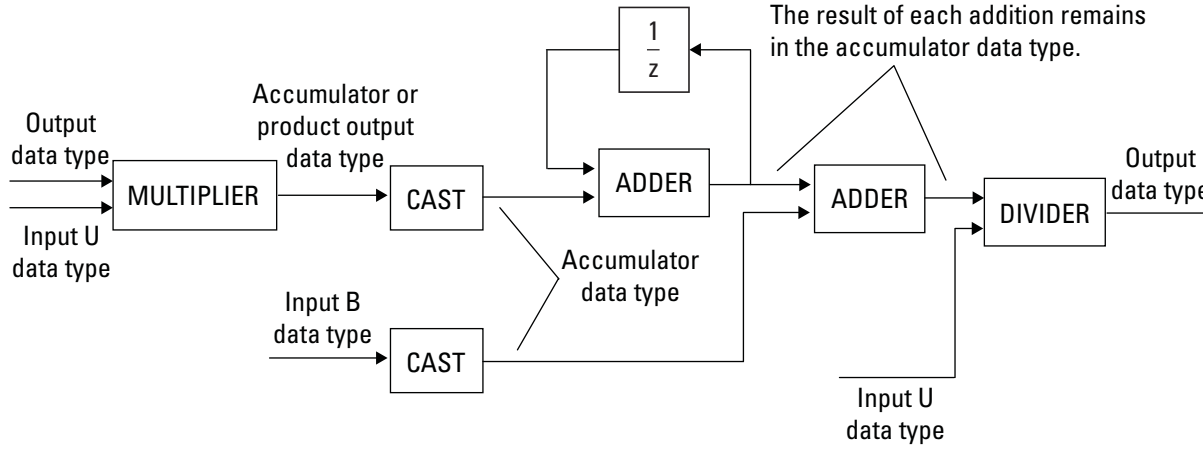
The block uses only the elements in the upper triangle of input U and ignores the lower elements. When you select the Input U is unit-upper triangular check box, the block replaces the elements on the diagonal of U with 1s. This is useful when matrix U is the result of another operation, such as an LDL decomposition, that uses the diagonal elements to represent the D matrix.

The block treats a length- M vector input at port B as an M-by-1 matrix.

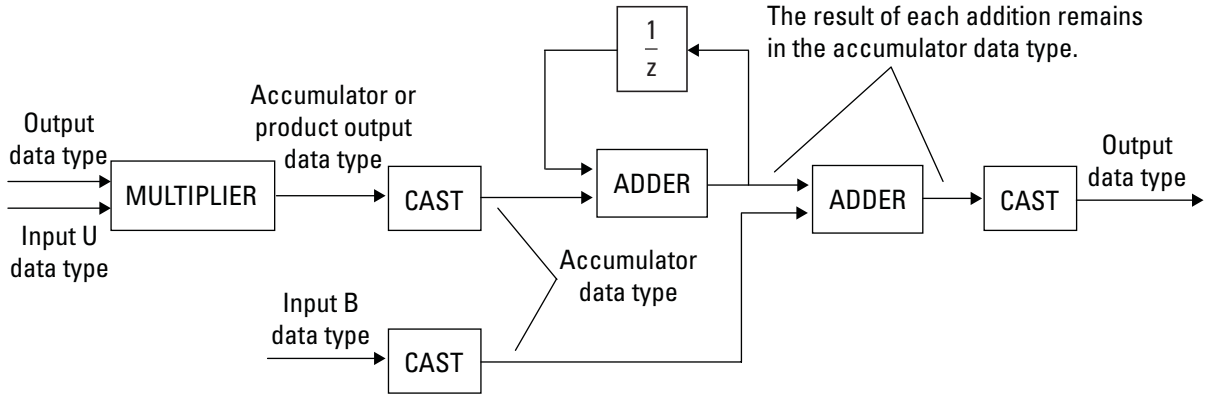
Fixed-Point Data Types

The following diagram shows the data types used within the Backward Substitution block for fixed-point signals.

When input U is not unit-upper triangular:



When input U is unit-upper triangular:

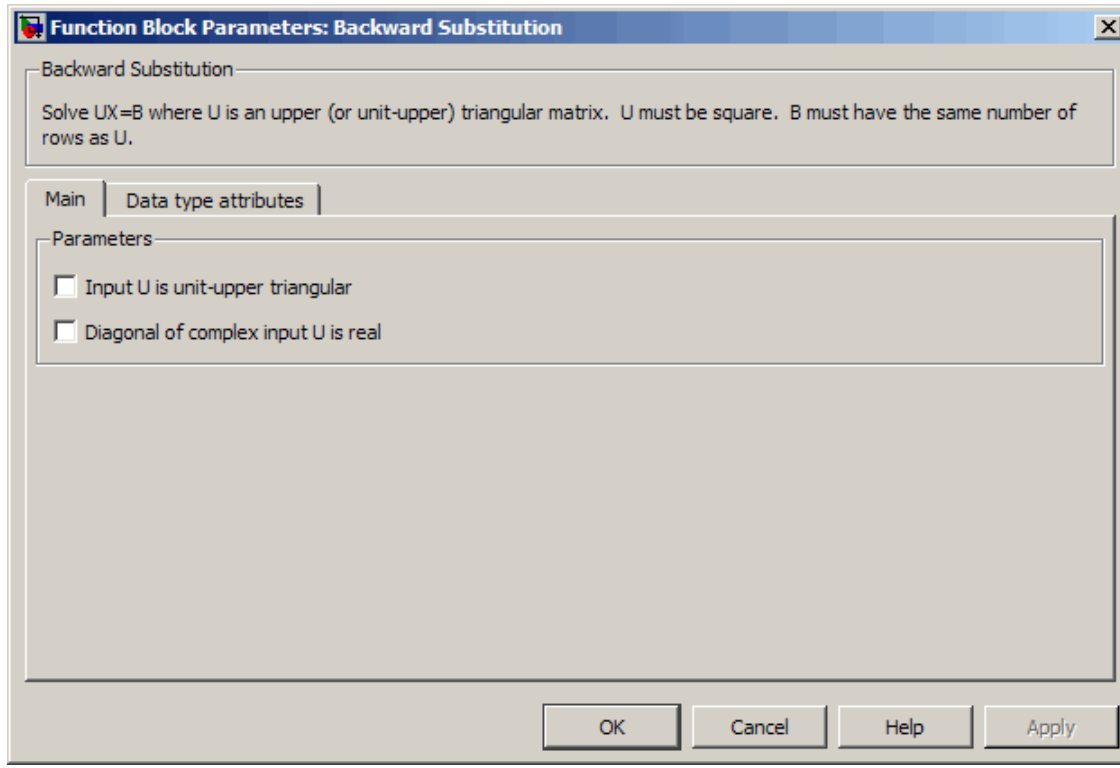


You can set the product output, accumulator, and output data types in the block dialog as discussed in the following section.

The output of the multiplier is in the product output data type when the input is real. When the input is complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see “Multiplication Data Types”.

Dialog Box

The Main pane of the Backward Substitution block dialog box appears as follows.



Input U is unit-upper triangular

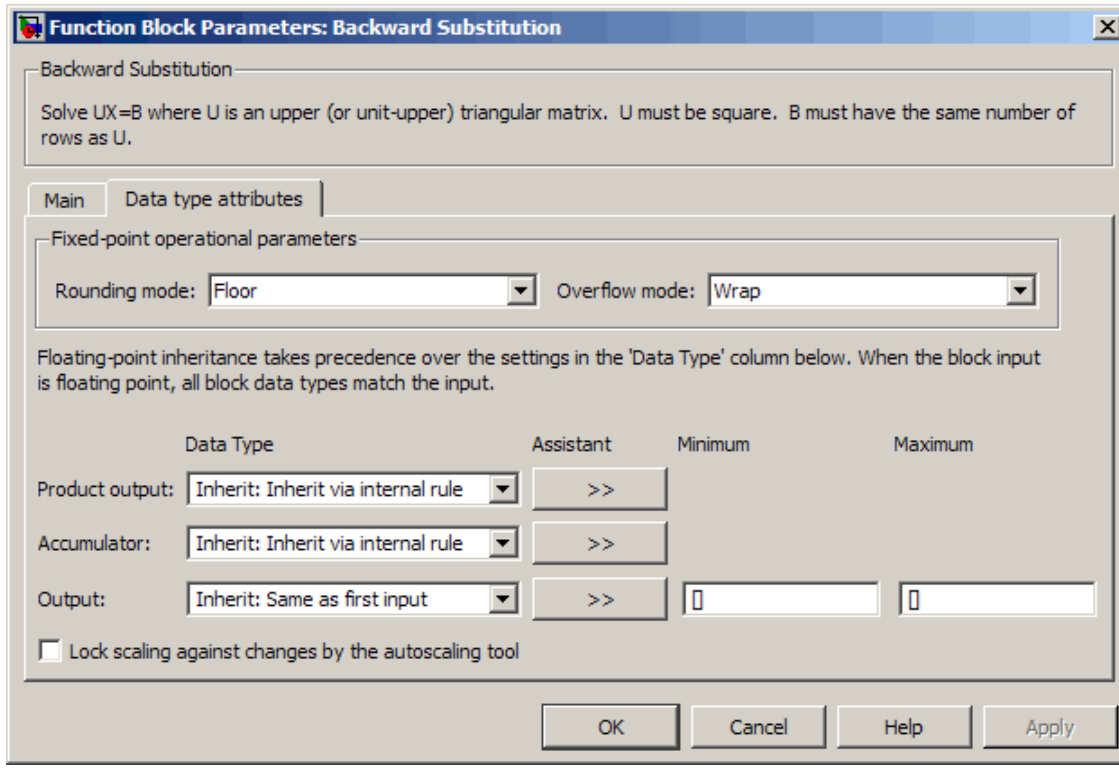
Select to replace the elements on the diagonal of U with 1s.

Diagonal of complex input U is real

Select to optimize simulation speed when the diagonal elements of complex input U are real. This parameter is only visible when Input U is unit-upper triangular is not selected.

Note When U is a complex Þxed-point signal, you must select either Input U is unit-upper triangular or Diagonal of complex input U is real . In such a case, any imaginary part of the diagonal of U is ignored.

The Data type attributes pane of the Backward Substitution block dialog appears as follows.



Note Floating-point inheritance takes precedence over the data type settings de Þned on this pane. When inputs are ßoating point, the block ignores these settings, and a ll internal data types are ßoating point.

Rounding mode

Select the rounding mode for Þxed-point operations.

Over flow mode

Select the over ßow mode for Þxed-point operations.

Product output data type

Specify the product output data type. See “Fixed-Point Data Types” on page 2-58 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block. You can set it to:

- A rule that inherits a data type, for example, Inherit: Inherit via internal rule
- An expression that evaluates to a data type, for example, fixdt(1,16,0)

Click the Show data type assistant button

to display the Data Type Assistant , which helps you set the Product output data type parameter.

See “Using the Data Type Assistant” in Simulink User’s Guide for more information.

Accumulator data type

Specify the accumulator data typ e. See “Fixed-Point Data Types” on page 2-58 for illustrations depicting the use of the accumulator data type in this block. You can set this parameter to:

- A rule that inherits a data type, for example, Inherit: Inherit via internal rule

- An expression that evaluates to a data type, for example, fixdt(1,16,0)

Click the Show data type assistant button to display the Data Type Assistant, which helps you set the Accumulator data type parameter.

See "Using the Data Type Assistant" in Simulink User's Guide for more information.

Output data type

Specify the output data type. See "Fixed-Point Data Types" on page 2-58 for illustrations depicting the use of the output data type in this block. You can set it to:

- A rule that inherits a data type, for example, Inherit: Same as first input
- An expression that evaluates to a data type, for example, fixdt(1,16,0)

Click the Show data type assistant button to display the Data Type Assistant, which helps you set the Output data type parameter.

See "Specifying Block Output Data Types" in Simulink User's Guide for more information.

Minimum

Specify the minimum value that the block should output. The default value, [], is equivalent to -Inf. Simulink software uses this value to perform:

- Simulation range checking (see "Checking Signal Ranges")
- Automatic scaling of fixed-point data types

Maximum

Specify the maximum value that the block should output. The default value, [] , is equivalent to Inf . Simulink software uses this value to perform:

- Simulation range checking (see “Checking Signal Ranges”)
- Automatic scaling of Þxed-point data types

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any Þxed-point scaling you specify in this block mask f rom being overridden by the autoscaling tool in the Fixed-Point Tool.

Supported Data Types

Port	Supported Data Types
U	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
B	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
X	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

See Also

Cholesky Solver Signal Processing Blockset

Forward Substitution Signal Processing Blockset

LDL Solver	Signal Processing Blockset
Levinson-Durbin	Signal Processing Blockset
LU Solver	Signal Processing Blockset
QR Solver	Signal Processing Blockset

See “Linear System Solvers” for related information.

Purpose

Design bandpass filter

Library

Filtering / Filter Design Toolbox

dspfdesign

Description

This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See “Bandpass Filter Design Dialog Box — Main Pane” in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset product.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point

Port	Supported Data Types
	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Purpose

Design bandstop filter

Library

Filtering / Filter Design Toolbox

dspfdesign

Description

This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See “Bandstop Filter Design Dialog Box — Main Pane” in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset product.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point

Port	Supported Data Types
	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Purpose

Model biquadratic IIR (SOS) filters

Library

Filtering / Filter Designs

dsparch4

Description

The Biquad Filter block independently filters each channel of the input signal with the specified biquadratic IIR filter. The block implements static filters with fixed coefficients that you can tune during simulation.

This block filters each channel of the input signal independently over time, treating each element of the input as an individual channel. The output dimensions always equal those of the input signal. The outputs of this block numerically match the outputs of the Signal Processing Toolbox dfilt function.

The Biquad Filter block supports the Simulink state logging feature. See "States" in the Simulink User's Guide for more information. The Digital Filter block also has direct feedthrough. Therefore, if you connect the output of this block back to its input, you get an algebraic loop. For more information on direct feedthrough and algebraic loops, see "Algebraic Loops" in the Simulink documentation.

Coefficient Source and Filter Structures

The Biquad Filter block can operate in three different modes. Select the mode in the Coefficient source group box.

- If you select Dialog parameters, you enter information about the filter such as structure and coefficients in the block mask. In this mode, you can choose the following filter structures in the Filter structure parameter:
  - Direct form I
  - Direct form I transposed
  - Direct form II
  - Direct form II transposed

- If you select Input port(s) , you enter information about the filter structure in the block mask using the Filter structure parameter, but the filter coefficients come into the block via input ports. The following additional ports appear on the block icon:
  - Num — numerator coefficients
  - Den — denominator coefficients
  - g — scale values
- If you select Discrete-time filter object (DFILT) , you specify the filter using a dfilt object from the Signal Processing Toolbox product or the Filter Design Toolbox product. This block supports the following dfilt structures:
  - dfilt.df1sos
  - dfilt.df1tsos
  - dfilt.df2sos
  - dfilt.df2tsos

Specifying the SOS Matrix and Scale Values

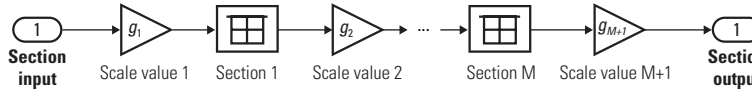
The SOS matrix (Mx6) is an M-by-6 matrix, where M is the number of sections in the second-order section filter. Each row of the SOS matrix contains the numerator and denominator coefficients (b_{ik} and a_{ik}) of the corresponding section in the filter.

$$\begin{bmatrix} b_{01} & b_{11} & b_{21} & a_{01} & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & a_{02} & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0M} & b_{1M} & b_{2M} & a_{0M} & a_{1M} & a_{2M} \end{bmatrix}$$

You can use the ss2sos and tf2sos functions from Signal Processing Toolbox software to convert a state-space or transfer function description of your filter into the second-order section description used by this block.

The Scale values parameter specifies the scalar or vector of M+1 scale values to be used between SOS sections.

- If you enter a scalar, the value specifies the gain value before the first section of the second-order filter. The rest of the gain values default to 1.
- If you enter a vector of M+1 values, each value specifies a separate section of the filter. For example, the first element is the first gain value, the second element is the second gain value, and so on.



Select the Optimize unity scale values check box to optimize your simulation when one or more scale values equal 1. Selecting this option removes the unity gains so that the values are treated like Simulink lines or wires. In some fixed-point cases when there are unity scale values, selecting this parameter also omits certain casts. Refer to "Filter Structure Diagrams" on page 2-91 for more information.

Specifying Initial Conditions

The Biquad Filter block initializes the internal filter states to zero by default. You can optionally use the Initial conditions or Initial conditions on zeros side and Initial conditions on poles side parameters to specify nonzero initial states for the filter delays.

To determine the number of initial conditions you must specify and how to specify them, see the following table on valid initial conditions.

Valid Initial Conditions

Initial Condition	Description
Scalar	The block initializes all delay elements in the filter to the scalar value.
Vector or matrix (for applying different delay elements to each channel)	Each vector or matrix element specifies a unique initial condition for a corresponding delay element in a corresponding channel. Where M is the number of sections and N is the number of input channels: • The vector length must equal the number of delay elements in the filter, max(#_of_zeros, #_of_poles)-1, or M*2. • The matrix must have the same number of rows as the number of delay elements in the filter, max(#_of_zeros, #_of_poles)-1, or (M*2)*N. The matrix must also have one column for each channel of the input signal.

Fixed-Point Data Types

See “Filter Structure Diagrams” on page 2-91. The following constraints apply when you process a fixed-point signal with any of the filter structures supported by this block:

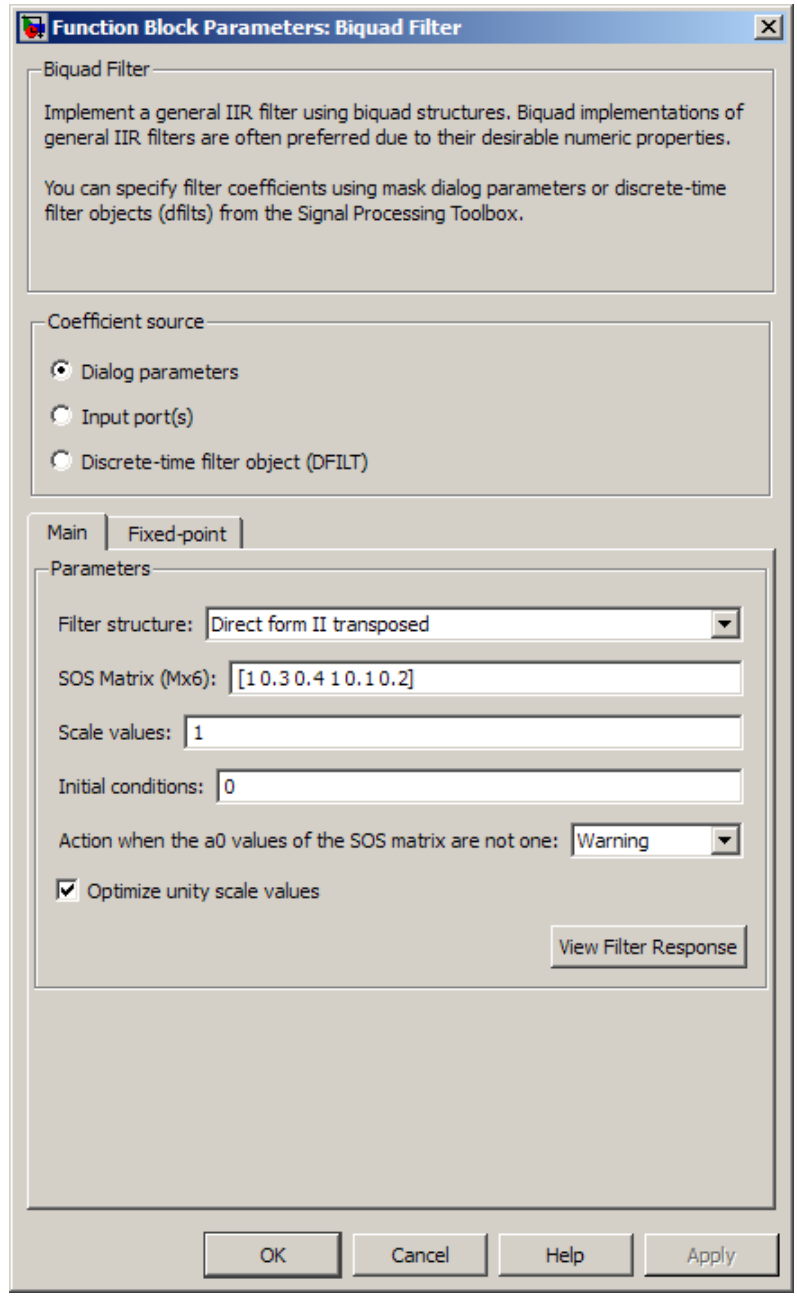
- Inputs and coefficients can be real or complex.
- Numerator and denominator coefficients can be real or complex.
- Specify the coefficients by a M-by-6 matrix in the block mask.
- States are complex when either the inputs or the coefficients are complex.
- Scale values must be real.
- The scale value parameter must be a scalar or a vector of length M+1, where M is the number of sections.
- The Section input and Section output parameters determine the data type for the section input and output data types.

Examples

Open an example model by typing `doc_biquad_filter_ref` at the MATLAB command line.

Dialog Box

The Main pane of the Biquad Filter block dialog appears as follows when Dialog parameters is selected in the Coefficient source group box.


Function Block Parameters: Biquad Filter
Biquad Filter
Implement a general IIR filter using biquad structures. Biquad implementations of general IIR filters are often preferred due to their desirable numeric properties.
You can specify filter coefficients using mask dialog parameters or discrete-time filter objects (dfilts) from the Signal Processing Toolbox.
Coefficient source
Dialog parameters
Input port(s)
Discrete-time filter object (DFILT)
Main
Fixed-point
Parameters
Filter structure: Direct form II transposed
SOS Matrix (Mx6): [1 0.3 0.4 1 0.1 0.2]
Scale values: 1
Initial conditions: 0
Action when the a0 values of the SOS matrix are not one: Warning
Optimize unity scale values
View Filter Response
OK
Cancel
Help
Apply

Filter structure

Select the filter structure.

This parameter is only visible when Dialog parameters or Input port(s) is selected.

SOS Matrix

Specify an M-by-6 matrix, where M is the number of sections in the second-order section filter. Each row of the SOS matrix contains the numerator and denominator coefficients (b_{ik} and a_{ik}) of the corresponding section in the filter.

$$\begin{bmatrix} b_{01} & b_{11} & b_{21} & a_{01} & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & a_{02} & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0M} & b_{1M} & b_{2M} & a_{0M} & a_{1M} & a_{2M} \end{bmatrix}$$

You can use the ss2sos and tf2sos functions from Signal Processing Toolbox software to check whether your SOS matrix is valid.

This parameter is only visible when Dialog parameters is selected.

Scale values

The Scale values parameter specifies the scalar or vector of M+1 scale values to be used between SOS sections.

- When you enter a scalar, the value specifies the gain value before the first section of the second-order filter. The rest of the gain values default to 1.
- When you enter a vector of M+1 values, each value specifies a separate section of the filter. For example, the first element is the first gain value, the second element is the second gain value, and so on.

This parameter is only visible when Dialog parameters is selected.

Initial conditions

Specify the initial conditions of the filter states. To learn how to specify initial conditions, see “Specifying Initial Conditions” on page 2-73.

This parameter is only visible when Dialog parameters or Input port(s) is selected and the filter structure is Direct form II or Direct form II transposed.

Initial conditions on zeros side

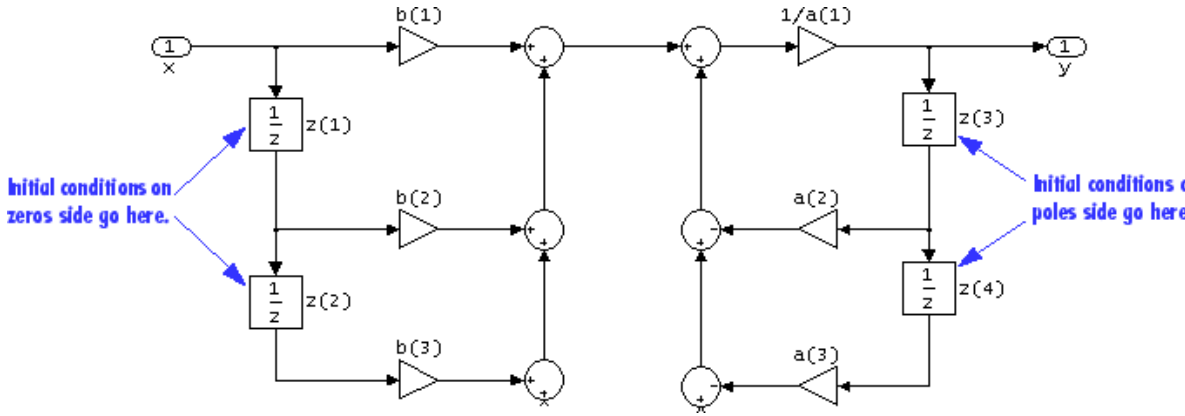
Specify the initial conditions for the filter states on the side of the filter structure with the zeros (b_0, b_1,b_2, ...); see the next diagram.

This parameter is only visible when Dialog parameters or Input port(s) is selected and the filter structure is Direct form I or Direct form I transposed. To learn how to specify initial conditions, see“Specifying Initial Conditions” on page 2-73.

Initial conditions on poles side

Specify the initial conditions for the filter states on the side of the filter structure with the poles (a_0, a_1,a_2, ...); see the next diagram.

This parameter is only visible when Dialog parameters or Input port(s) is selected and the filter structure is Direct form I or Direct form I transposed. To learn how to specify initial conditions, see “Specifying Initial Conditions” on page 2-73.



Action when the a0 values of the SOS matrix are not one

Specify the action the block should perform when the SOS matrix a_{0j} values do not equal one; None, Error , or Warn.

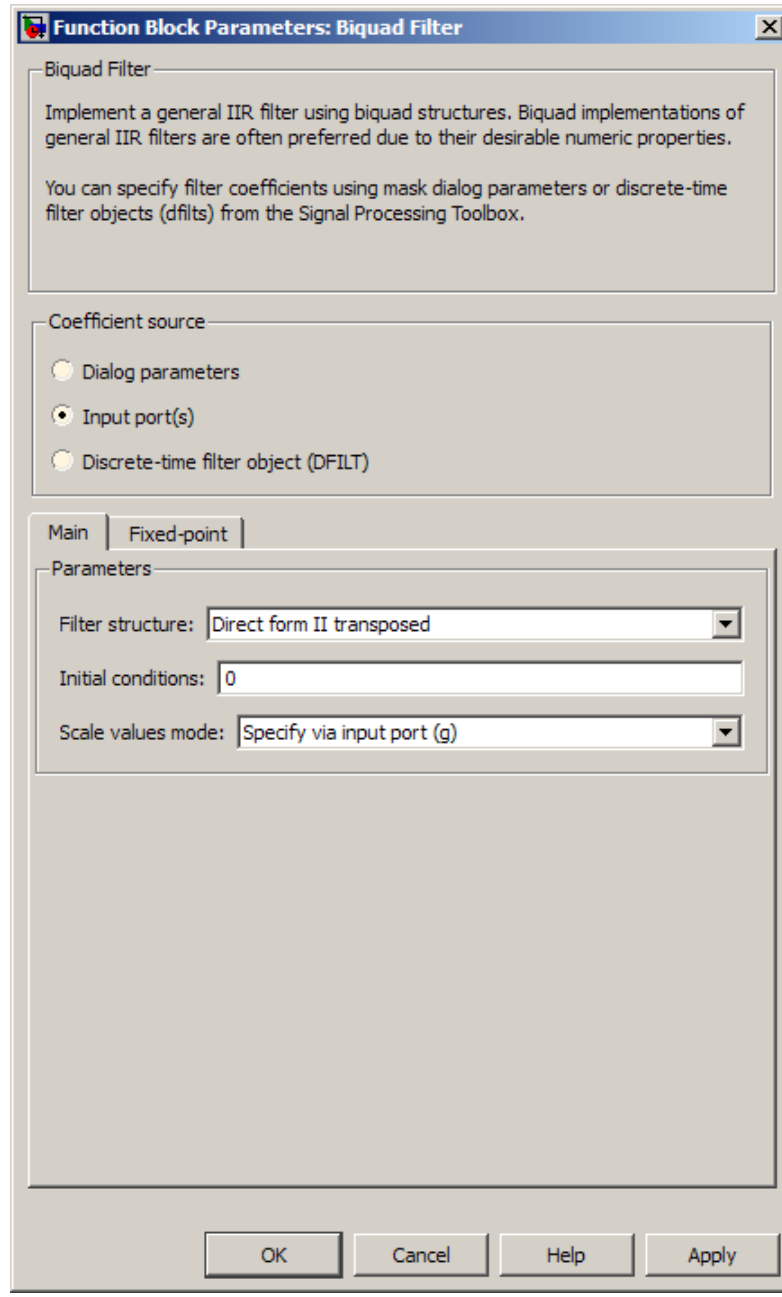
This parameter is only visible when Dialog parameters is selected.

Optimize unity scale values

Select this check box to optimi ze your simulation when one or more scale values equal 1. Selecting this option removes the unity gains so that the values are treated like Simulink lines or wires. In some Þxed-point cases when there are unity scale values, selecting this parameter also omits certain casts. Refer to “Filter Structure Diagrams” on page 2-91 for more information.

This parameter is only visible when Dialog parameters is selected.

The Main pane of the Biquad Filter block dialog appears as follows when Input port(s) is selected in the Coefficient source group box.


Function Block Parameters: Biquad Filter
Biquad Filter
Implement a general IIR filter using biquad structures. Biquad implementations of general IIR filters are often preferred due to their desirable numeric properties.
You can specify filter coefficients using mask dialog parameters or discrete-time filter objects (dfilts) from the Signal Processing Toolbox.
Coefficient source
Dialog parameters
Input port(s)
Discrete-time filter object (DFILT)
Main
Fixed-point
Parameters
Filter structure:
Direct form II transposed
Initial conditions:
0
Scale values mode:
Specify via input port (g)
OK
Cancel
Help
Apply

Filter structure

Select the filter structure.

This parameter is only visible when Dialog parameters or Input port(s) is selected.

Initial conditions

Specify the initial conditions of the filter states. To learn how to specify initial conditions, see “Specifying Initial Conditions” on page 2-73.

This parameter is only visible when Dialog parameters or Input port(s) is selected and the filter structure is Direct form II or Direct form II transposed.

Initial conditions on zeros side

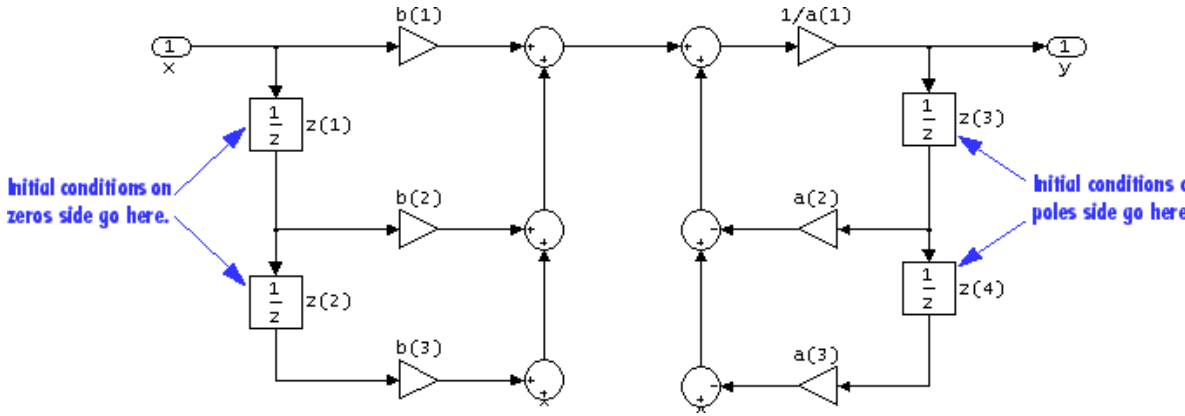
Specify the initial conditions for the filter states on the side of the filter structure with the zeros (b_0, b_1,b_2, ...); see the next diagram.

This parameter is only visible when Dialog parameters or Input port(s) is selected and the filter structure is Direct form I or Direct form I transposed. To learn how to specify initial conditions, see“Specifying Initial Conditions” on page 2-73.

Initial conditions on poles side

Specify the initial conditions for the filter states on the side of the filter structure with the poles (a_0, a_1,a_2, ...); see the next diagram.

This parameter is only visible when Dialog parameters or Input port(s) is selected and the filter structure is Direct form I or Direct form I transposed. To learn how to specify initial conditions, see “Specifying Initial Conditions” on page 2-73.

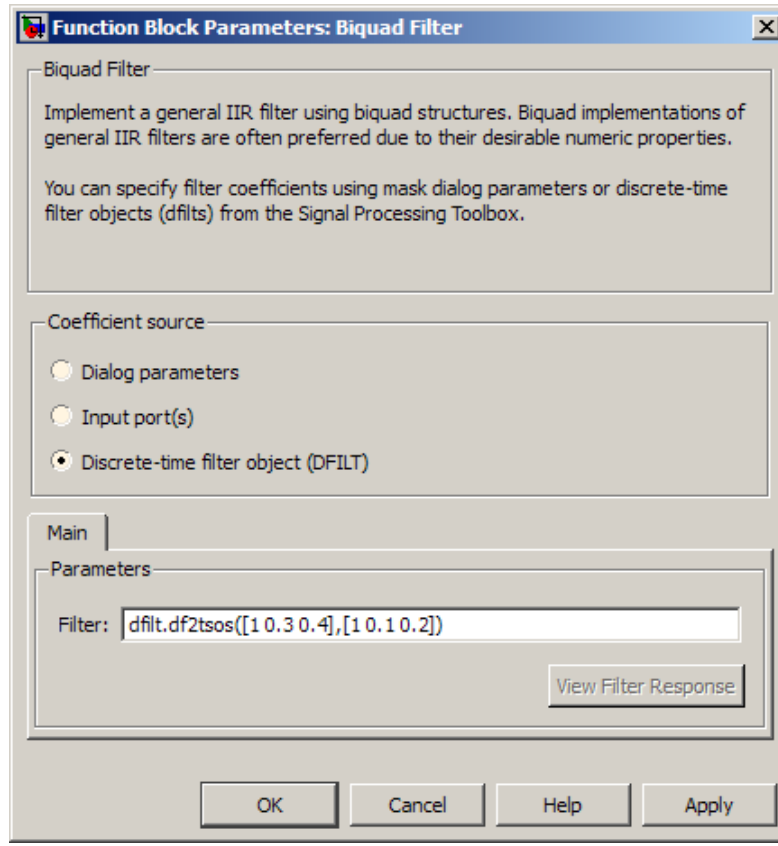


Scale values mode

Choose how to specify the scale values to use between filter sections. When you select Specify via input port (g) , you enter the scale values as a 2-D vector at port g. When you select Assume all are unity and optimize , all scale values are removed and treated like Simulink lines or wires.

This parameter is only visible when Input port(s) is selected.

The Main pane of the Biquad Filter block dialog appears as follows when Discrete-time filter object (DFILT) is selected in the Coefficient source group box.


Function Block Parameters: Biquad Filter
Biquad Filter
Implement a general IIR filter using biquad structures. Biquad implementations of general IIR filters are often preferred due to their desirable numeric properties.
You can specify filter coefficients using mask dialog parameters or discrete-time filter objects (dfilts) from the Signal Processing Toolbox.
Coefficient source
Dialog parameters
Input port(s)
Discrete-time filter object (DFILT)
Main
Parameters
Filter:
dfilt.df2tsos([1 0.3 0.4],[1 0.1 0.2])
View Filter Response
OK
Cancel
Help
Apply

Discrete-time filter object (DFILT)

Specify the discrete-time filter object (dfilt) that you would like the block to implement. You can do so in one of three ways:

- You can fully specify the dfilt object in the block mask.
- You can enter the variable name of a dfilt object that is defined in any workspace.
- You can enter a variable name for a dfilt object that is not yet defined.

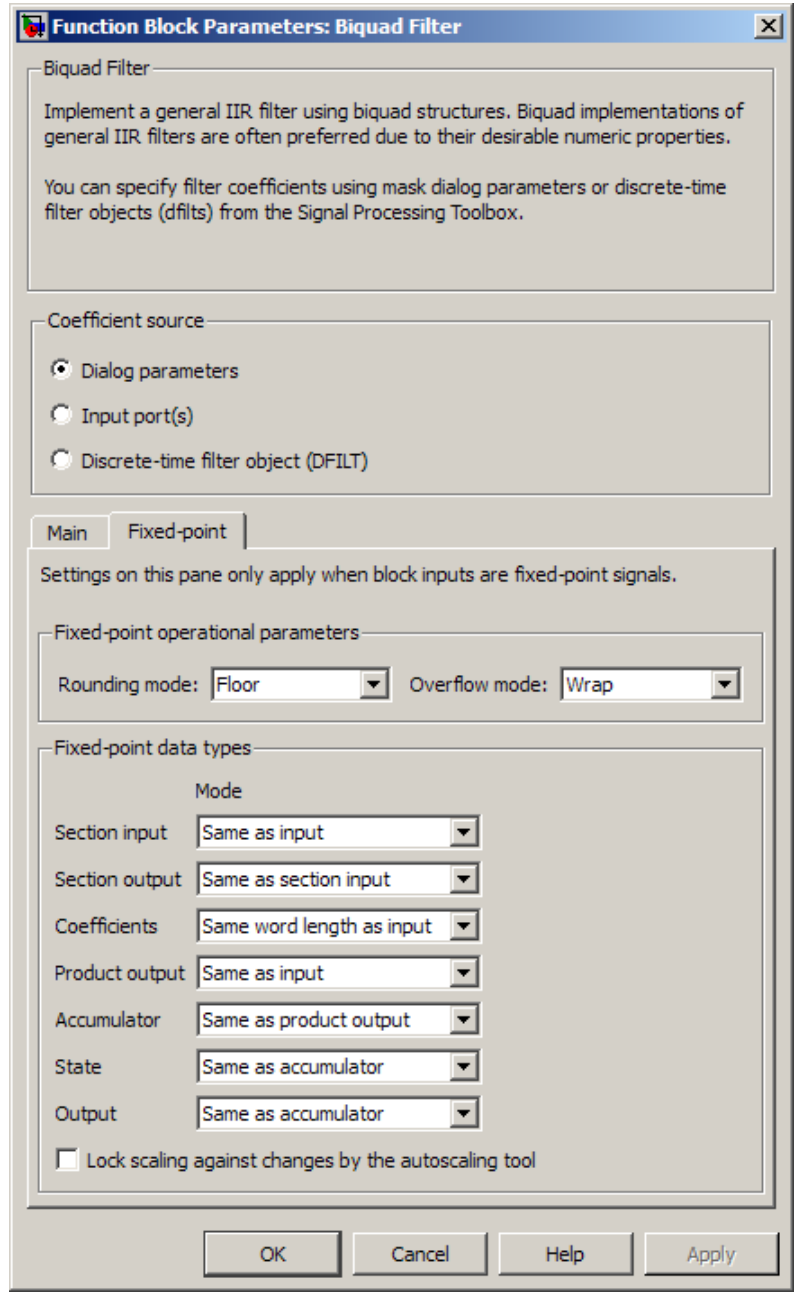
For more information on creating dfilt objects, see the dfilt function reference page in the Signal Processing Toolbox documentation.

View filter response

This button opens the Filter Visualization Tool (fvtool) from the Signal Processing Toolbox product and displays the filter response of the dfilt object specified in the Discrete-time filter object (DFILT) parameter. For more information on FVTool, see the Signal Processing Toolbox documentation.

Note This button is only available when the Filter Design Toolbox and Fixed-Point Toolbox products a re installed. If you specify a filter in the Discrete-time filter object (DFILT) parameter, you must apply the filter by clicking the Apply button before using the View filter response button.

The Fixed-point pane of the Biquad Filter block dialog appears as follows. This pane only appears when Dialog parameters or Input port(s) is selected in the Coef ficient source group box.


Function Block Parameters: Biquad Filter
Biquad Filter
Implement a general IIR filter using biquad structures. Biquad implementations of general IIR filters are often preferred due to their desirable numeric properties.
You can specify filter coefficients using mask dialog parameters or discrete-time filter objects (dfilts) from the Signal Processing Toolbox.
Coefficient source
Dialog parameters
Input port(s)
Discrete-time filter object (DFILT)
Main
Fixed-point
Settings on this pane only apply when block inputs are fixed-point signals.
Fixed-point operational parameters
Rounding mode: Floor
Overflow mode: Wrap
Fixed-point data types
Mode
Section input: Same as input
Section output: Same as section input
Coefficients: Same word length as input
Product output: Same as input
Accumulator: Same as product output
State: Same as accumulator
Output: Same as accumulator
Lock scaling against changes by the autoscaling tool
OK
Cancel
Help
Apply

Rounding mode

Select the rounding mode for fixed-point operations. The filter coefficients do not obey this parameter; instead, they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations. The filter coefficients do not obey this parameter; instead, they are always saturated.

Section input

Choose how you specify the word length and the fraction length of the fixed-point data type going into each section of a biquadratic filter. See "Fixed-Point Data Types" on page 2-74 for illustrations depicting the use of the section input data type in this block.

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word and fraction lengths of the section input and output, in bits.
- When you select Slope and bias scaling, you can enter the word lengths, in bits, and the slopes of the section input and output. This block requires power-of-two slope and a bias of zero.

Section output

Choose how you specify the word length and the fraction length of the fixed-point data type coming out of each section of a biquadratic filter. See "Fixed-Point Data Types" on page 2-74 for illustrations depicting the use of the section input data type in this block.

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word and fraction lengths of the section input and output, in bits.

- When you select Slope and bias scaling, you can enter the word lengths, in bits, and the slopes of the section input and output. This block requires power-of-two slope and a bias of zero.

Coefficients

Choose how you specify the word length and the fraction length of the filter coefficients (numerator, denominator, and scale value) when Dialog parameters is selected in the Coefficient source group box. See "Fixed-Point Data Types" on page 2-74 for illustrations depicting the use of the coefficient data types in this block:

- When you select Same word length as input, the word length of the filter coefficients match that of the input to the block. In this mode, the block automatically sets the fraction length of the coefficients to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select Specify word length, you can enter the word length of the coefficients, in bits. In this mode, the block automatically sets the fraction length of the coefficients to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select Binary point scaling, you can enter the word length and the fraction length of the coefficients, in bits. If applicable, you can enter separate fraction lengths for the numerator and denominator coefficients.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the coefficients. If applicable, you can enter separate slopes for the numerator and denominator coefficients. This block requires power-of-two slope and a bias of zero.

- The filter coefficients do not obey the Rounding mode and the Overflow mode parameters; instead, they are always saturated and rounded to Nearest.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See "Fixed-Point Data Types" on page 2-74 and "Multiplication Data Types" for illustrations depicting the use of the product output data type in this block:

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths. See "Fixed-Point Data Types" on page 2-74 and "Multiplication Data Types" for illustrations depicting the use of the accumulator data type in this block:

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Same as product output, these characteristics match those of the product output.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

State

Use this parameter to specify how you would like to designate the state word and fraction lengths when Dialog parameters is selected in the Coefficient source group box. See “Fixed-Point Data Types” on page 2-74 for illustrations depicting the use of the state data type in this block.

This parameter is not visible for Direct form I and Direct form I transposed filter structures.

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Same as accumulator, these characteristics match those of the accumulator.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length. See “Fixed-Point Data Types” on page 2-74 for illustrations depicting the use of the output data type in this block.

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Same as accumulator, these characteristics match those of the accumulator.
- When you select Binary point scaling, you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

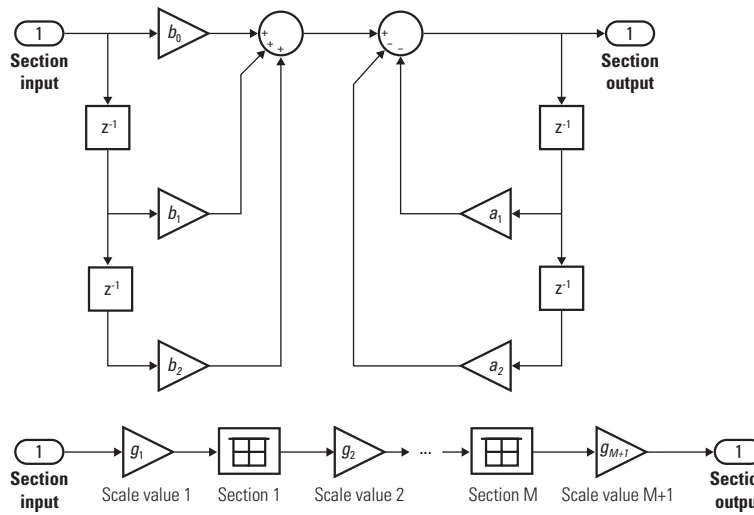
Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool.

Filter Structure Diagrams

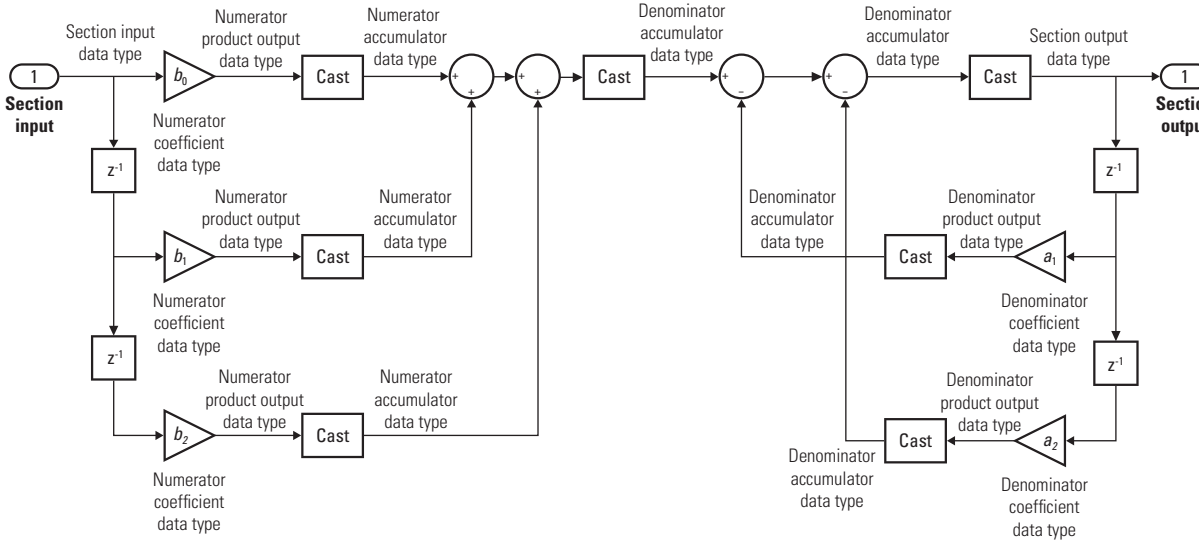
The diagrams in the following sections show the filter structures supported by the Biquad Filter block. They also show the data types used in the filter structures for fixed-point signals. You can set the data types shown in these diagrams in the block dialog box. This is discussed in "Dialog Box" on page 2-75.

- "Direct Form I" on page 2-91
- "Direct Form I Transposed" on page 2-93
- "Direct Form II" on page 2-96
- "Direct Form II Transposed" on page 2-99

Direct Form I

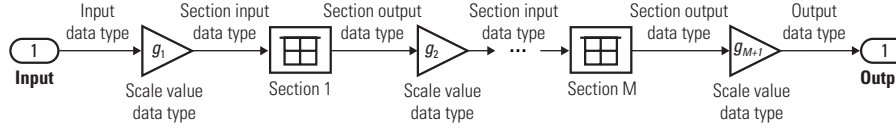


The following diagram shows the data types for one section of the Þlter for Þxed-point signals.

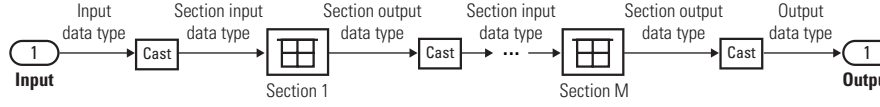


The following diagrams show the Þxed-point data types between Þlter sections.

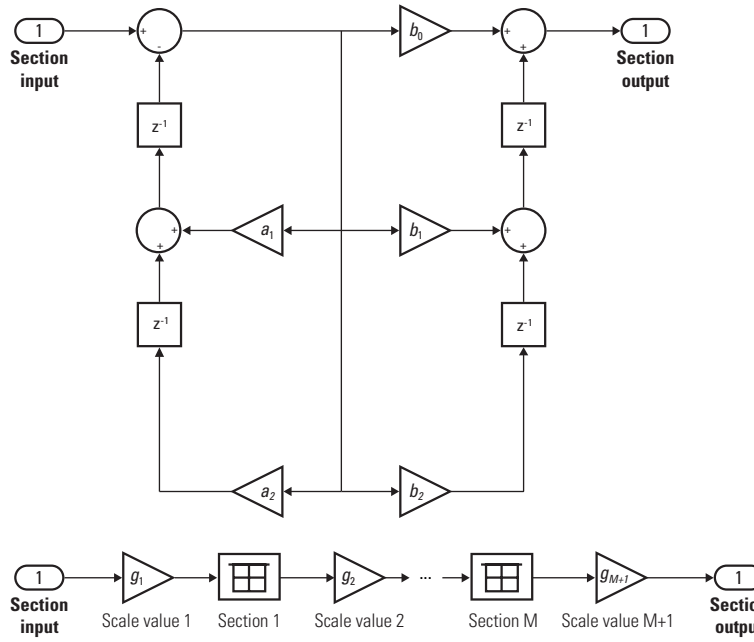
When the data is not optimized:



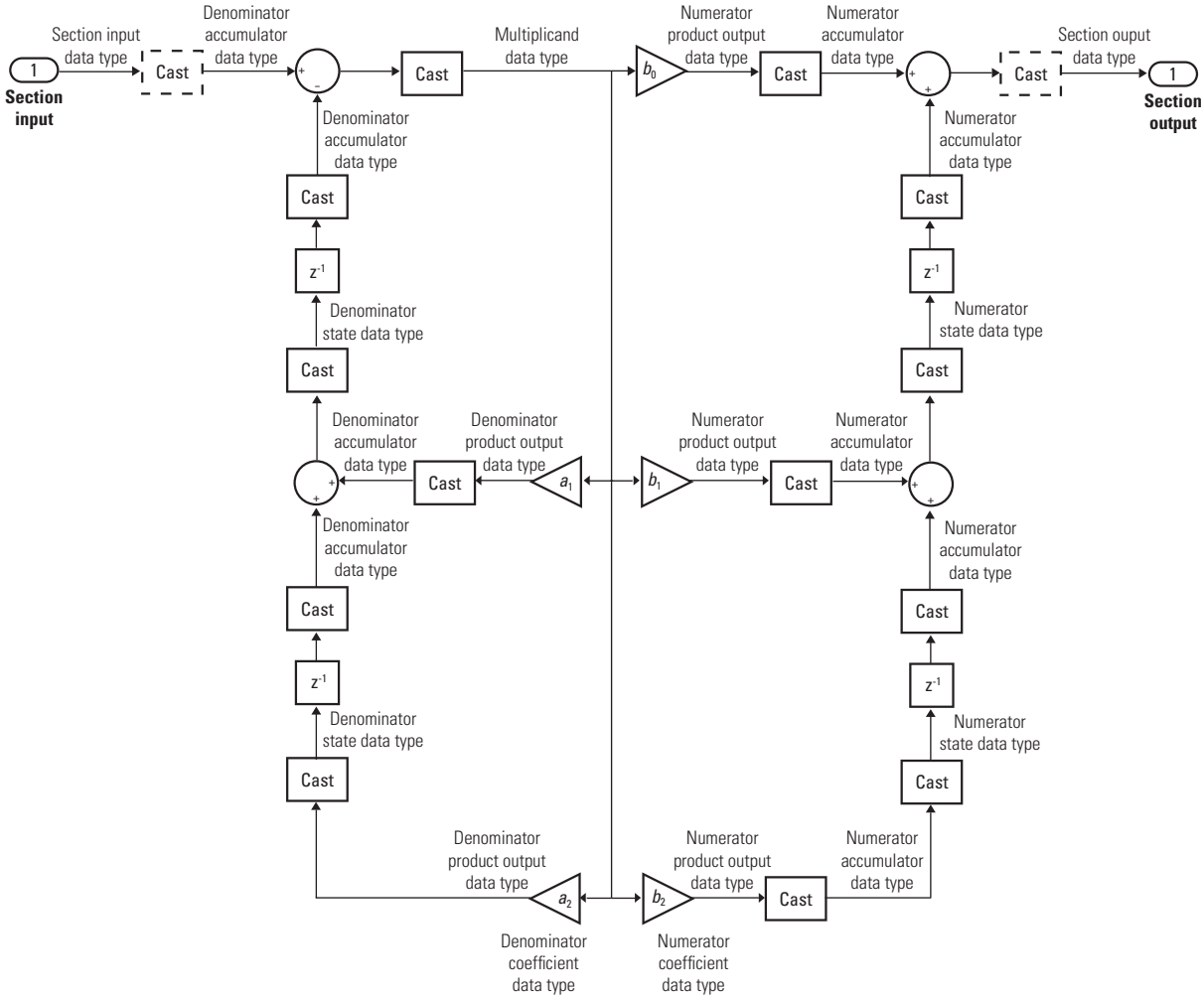
When you select Optimize unity scale values and scale values equal 1:



Direct Form I Transposed



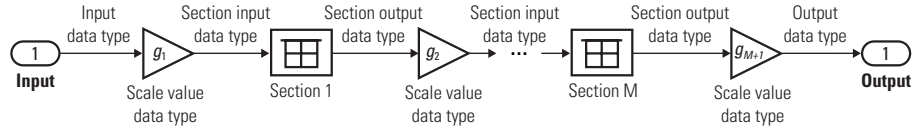
The following diagram shows the data types for one section of the Þlter for Þxed-point signals.



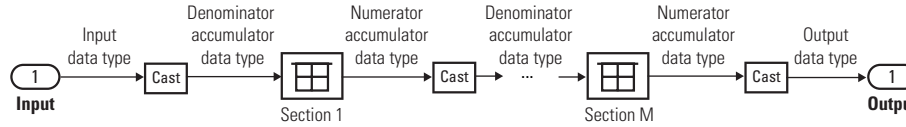
The dashed casts are omitted when Optimize unity scale values is selected and scale values equal one.

The following diagrams show the fixed-point data types between filter sections.

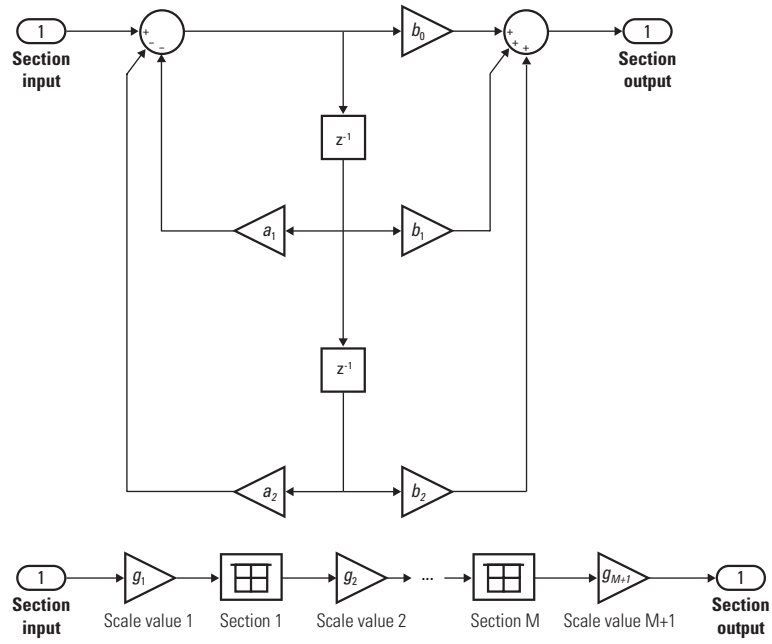
When the data is not optimized:



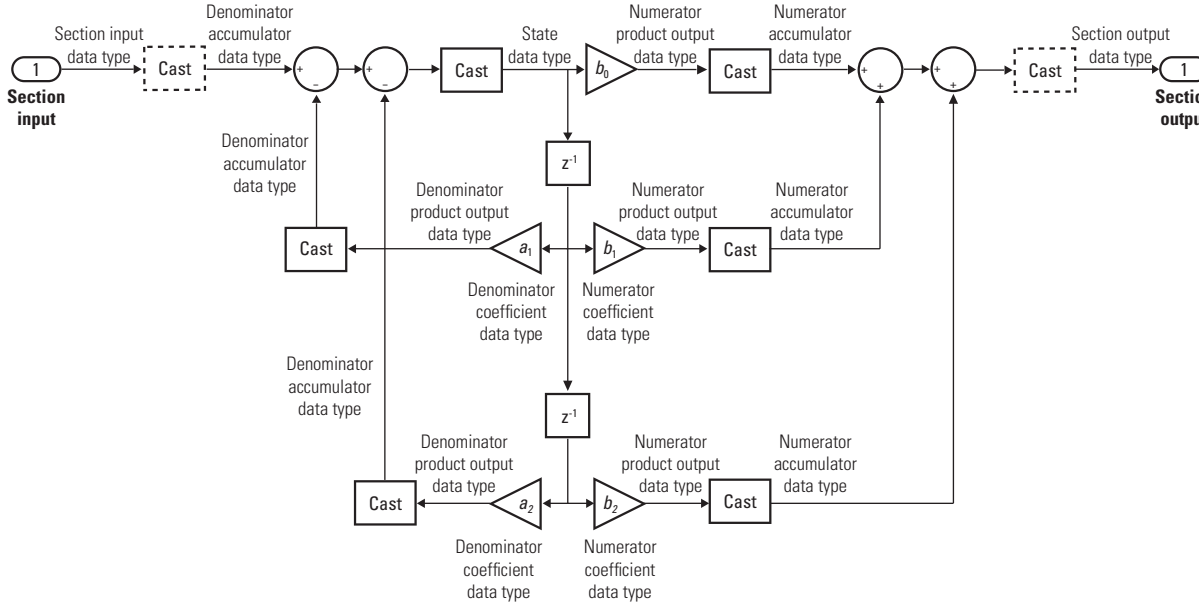
When you select Optimize unity scale values and scale values equal 1:



Direct Form II



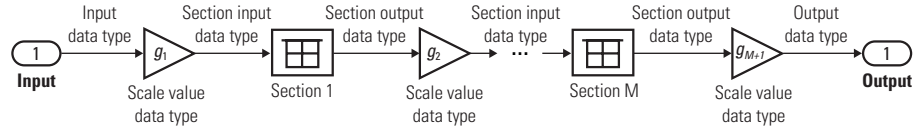
The following diagram shows the data types for one section of the Þlter for Þxed-point signals.



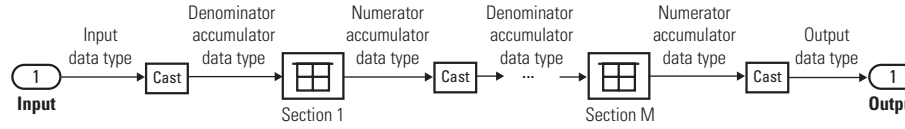
The dashed casts are omitted when Optimize unity scale values is selected and scale values equal one.

The following diagrams show the fixed-point data types between filter sections.

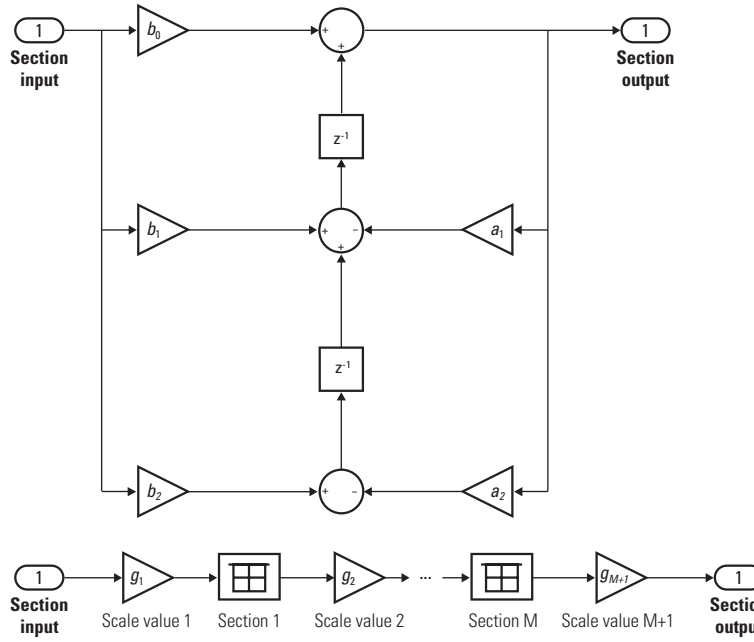
When the data is not optimized:



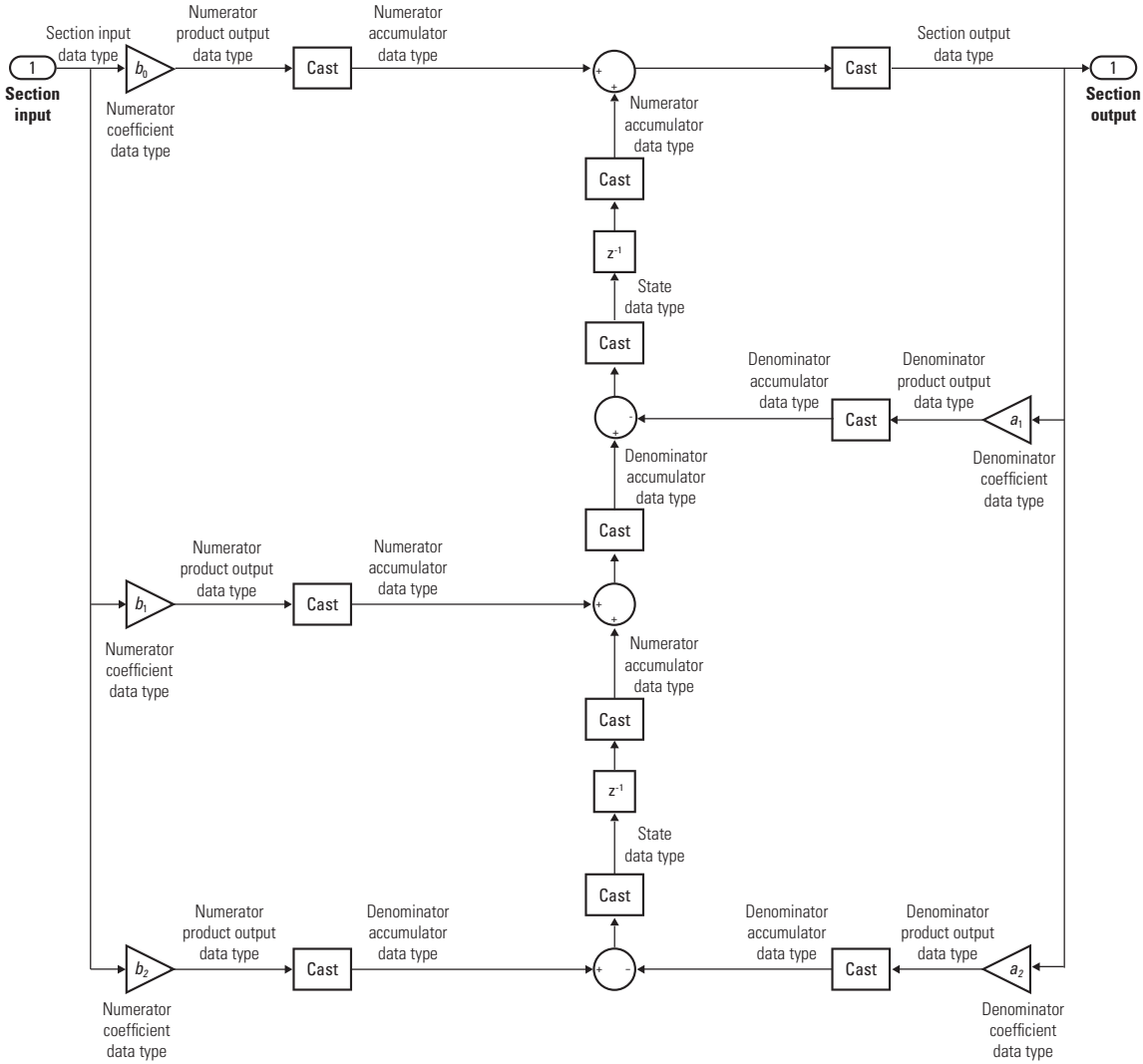
When you select Optimize unity scale values and scale values equal 1:



Direct Form II Transposed

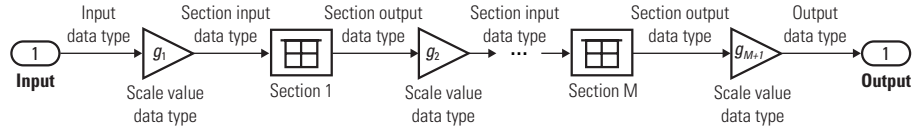


The following diagram shows the data types for one section of the Þlter for Þxed-point signals.

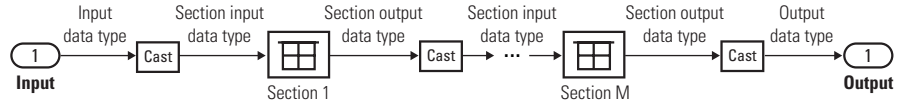


The following diagrams show the fixed-point data types between filter sections.

When the data is not optimized:



When you select Optimize unity scale values and scale values equal 1:



Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

See Also

dfilt.df1sos	Filter Design Toolbox
dfilt.df1tsos	Filter Design Toolbox

dfilt.df2sos	Filter Design Toolbox
dfilt.df2tsos	Filter Design Toolbox

Purpose

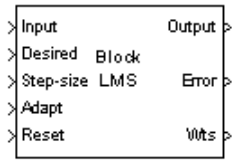
Compute output, error, and weigh ts using LMS adaptive algorithm

Library

Filtering / Adaptive Filters

dspadpt3

Description



Block LMS Filter

The Block LMS Filter block implements an adaptive least mean-square (LMS) Þlter, where the adaptation of Þlter weights occurs once for every block of samples. The block estimates the Þlter weights, or coef Þcients, needed to minimize the error, e(n), between the output signal, y(n), and the desired signal, d(n). Connect the signal you want to Þlter to the Input port. This input signal can be a sample-based scalar or a single-channel frame-based signal. Connect the signal you want to model to the Desired port. The desired signal must have the same data type, frame status, complexity, an d dimensions as the input signal. The Output port outputs the Þltered input signal, which can be sample or frame based. The Error port outputs the result of subtracting the output signal from the desired signal.

The block calculates the Þlter weights using the Block LMS adaptive Þlter algorithm. This algorithm is de Þned by the following equations.

$$n = kN + i$$

$$y(n) = \mathbf{w}^T(k-1)\mathbf{u}(n)$$

$$e(n) = d(n) - y(n)$$

$$\mathbf{w}(k) = \mathbf{w}(k-1) + f(\mathbf{u}(n), e(n), \mu)$$

The weight update function for the Block LMS adaptive Þlter algorithm is deÞned as

$$f(\mathbf{u}(n), e(n), \mu) = \mu \sum_{i=0}^{N-1} \mathbf{u}^*(kN+i)e(kN+i)$$

The variables are as follows.

Variable	Description
n	The current time index
i	The iteration variable in each block, $0 \le i \le N-1$
k	The block number
N	The block size
u(n)	The vector of buffered input samples at step n
w(n)	The vector of filter-tap estimates at step n
y(n)	The filtered output at step n
e(n)	The estimation error at time n
d(n)	The desired response at time n
μ	The adaptation step size

Use the Filter length parameter to specify the length of the filter weights vector.

The Block size parameter determines how many samples of the input signal are acquired before the filter weights are updated. The input frame length must be a multiple of the Block size parameter.

The adaptation Step-size (mu) parameter corresponds to μ in the equations. You can either specify a step-size using the input port, Step-size, or enter a value in the Block Parameters: Block LMS Filter dialog box.

Use the Leakage factor (0 to 1) parameter to specify the leakage factor, $0 < 1 - \mu\alpha \le 1$, in the leaky LMS algorithm shown below.

$$\mathbf{w}(k) = (1 - \mu\alpha)\mathbf{w}(k-1) + f(\mathbf{u}(n), e(n), \mu)$$

Enter the initial filter weights as a vector or a scalar in the Initial value of filter weights text box. When you enter a scalar, the block uses the scalar value to create a vector of filter weights. This vector

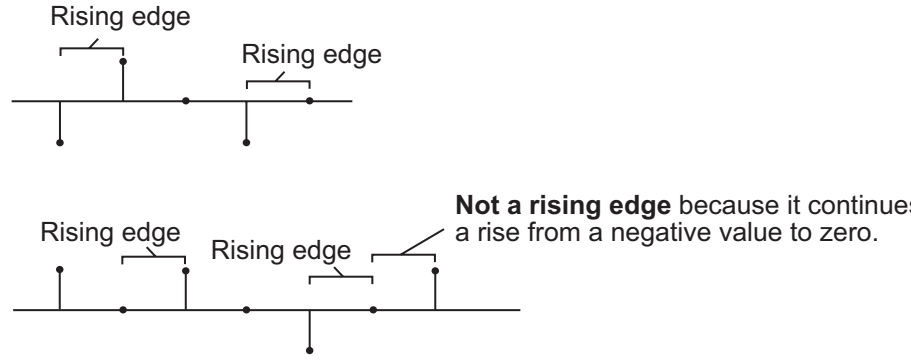
has length equal to the Þlter length and all of its values are equal to the scalar value

When you select the Adapt port check box, an Adapt port appears on the block. When the input to this port is greater than zero, the block continuously updates the Þlter weights. When the input to this port is zero, the Þlter weights remain at their current values.

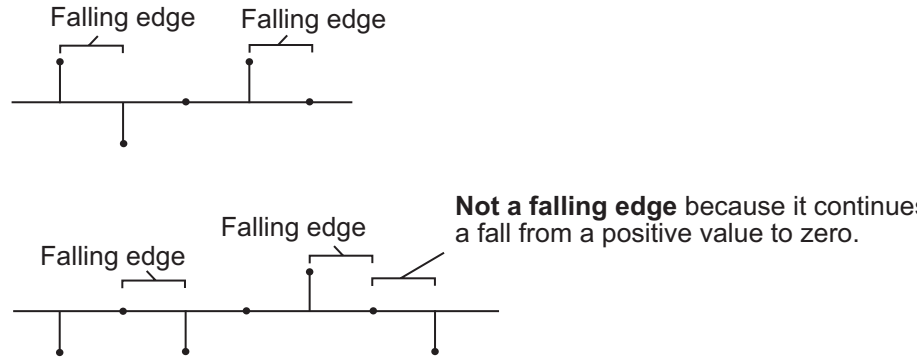
When you want to reset the value of the Þlter weights to their initial values, use the Reset input parameter. The block resets the Þlter weights whenever a reset event is de tected at the Reset port. The reset signal rate must be the same rate as the data signal input.

From the Reset input list, select None to disable the Reset port. To enable the Reset port, select one of the following from the Reset input list:

- Rising edge — Triggers a reset operation when the Reset input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following Þgure).



- Falling edge — Triggers a reset operation when the Reset input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)



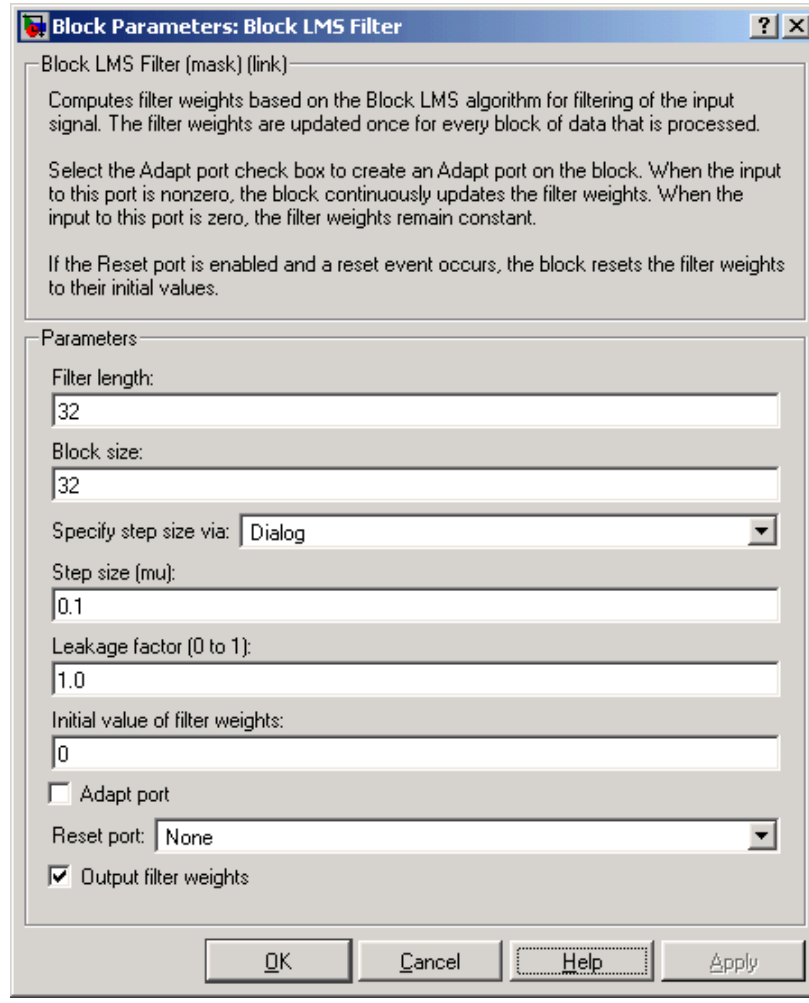
- Either edge — Triggers a reset operation when the Reset input is a Rising edge or Falling edge (as described above)
- Non-zero sample — Triggers a reset opera tion at each sample time that the Reset input is not zero

---

Note When running simulations in the Simulink MultiTasking mode, sample-based reset signals have a one-sample latency, and frame-based reset signals have one frame of latency. Thus, there is a one-sample or one-frame delay between the time the block detects a reset event, and when it applies the reset. For more information on latency and the Simulink tasking modes, see “Ex cess Algorithmic Delay (Tasking Latency)” and “Models with Multiple Sample Rates” in the Real-Time Workshop® User’s Guide.

Select the Output filter weights check box to create a Wts port on the block. For each iteration, the block outputs the current updated filter weights from this port.

Dialog Box



Filter length
: Enter the length of the FIR filter weights vector.

Block size
: Enter the number of samples to acquire before the filter weights are updated. The input frame length must be an integer multiple of the block size.

Specify step-size via
: Select Dialog to enter a value for mu in the Block parameters: LMS Filter dialog box. Select Input port to specify mu using the Step-size input port.

Step-size (mu)
: Enter the step-size. Tunable.

Leakage factor (0 to 1)
: Enter the leakage factor, $0 < 1 - \mu\alpha \leq 1$. Tunable.

Initial value of filter weights
: Specify the initial values of the FIR filter weights.

Adapt port
: Select this check box to enable the Adapt input port.

Reset input
: Select this check box to enable the Reset input port.

Output filter weights
: Select this check box to export the filter weights from the Wts port.

References

Hayes, M. H. Statistical Digital Signal Processing and Modeling. New York: John Wiley & Sons, 1996.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point

Port	Supported Data Types
Desired	• Must be the same as Input for ßoating-point signals • Must be any Þxed-point data type when Input is Þxed point
Step-size	• Must be the same as Input for ßoating-point signals • Must be any Þxed-point data type when Input is Þxed point
Adapt	• Double-precision ßoating point • Single-precision ßoating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Reset	• Double-precision ßoating point • Single-precision ßoating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Must be the same as Input for ßoating-point signals • Must be the same as Desired for Þxed-point signals

Port	Supported Data Types
Error	• Must be the same as Input for ßoating-point signals • Must be the same as Desired for Þxed-point signals
Wts	• Must be the same as Input for ßoating-point signals • Obeys the Weights parameter for Þxed-point signals

See Also

Fast Block LMS Filter — Signal Processing Blockset

Kalman Adaptive Filter (Obsolete) — Signal Processing Blockset

LMS Filter — Signal Processing Blockset

RLS Filter — Signal Processing Blockset

See “Adaptive Filters” for related information.

Purpose

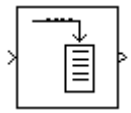
Buffer input sequence to smaller or larger frame size

Library

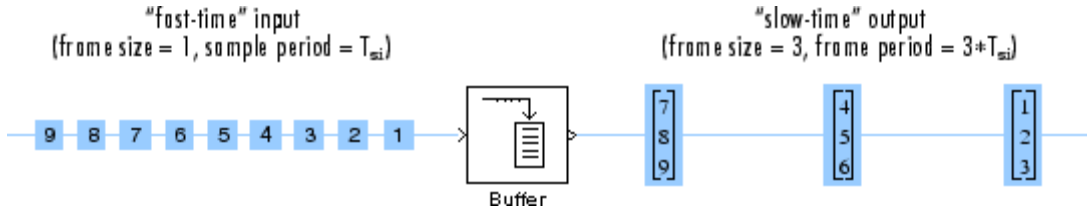
Signal Management / Buffers

dspbuff3

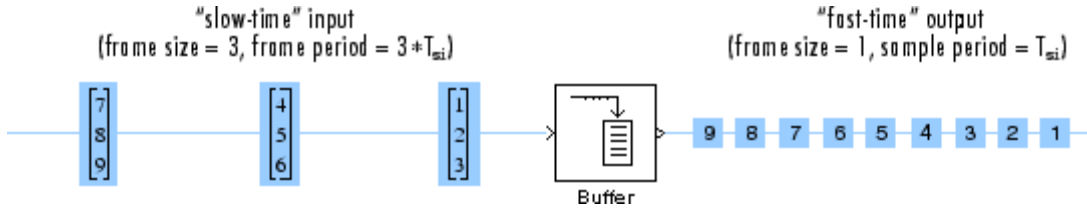
Description



The Buffer block redistributes the input samples to a new frame size. Buffering to a larger frame size yields an output with a slower frame rate than the input, as illustrated below for scalar input.



Buffering to a smaller frame size yields an output with a faster frame rate than the input, as illustrated below for scalar output.



The block coordinates the output frame size and frame rate of nonoverlapping buffers such that the sample period of the signal is the same at both the input and output: $T_{so} = T_{si}$.

This block supports triggered subsystems when the block's input and output rates are the same.

Sample-based full-dimension matrix inputs are not accepted.

Sample-Based Operation

Sample-based inputs are interpreted by the Buffer block as independent channels of data. Therefore, a sample-based length- N vector input is interpreted as N independent samples.

In sample-based operation, the Buffer block creates frame-based outputs from sample-based inputs. A sequence of sample-based length- N vector inputs is buffered into an M_o-by-N matrix, where M_o is specified by the Output buffer size parameter (M_o>1). That is, each input vector becomes a row in the N-channel frame-based output matrix. When M_o=1, the input is simply passed through to the output, and retains the same dimension.

The Buffer overlap parameter, L, specifies the number of samples (rows) from the current output to repeat in the next output,

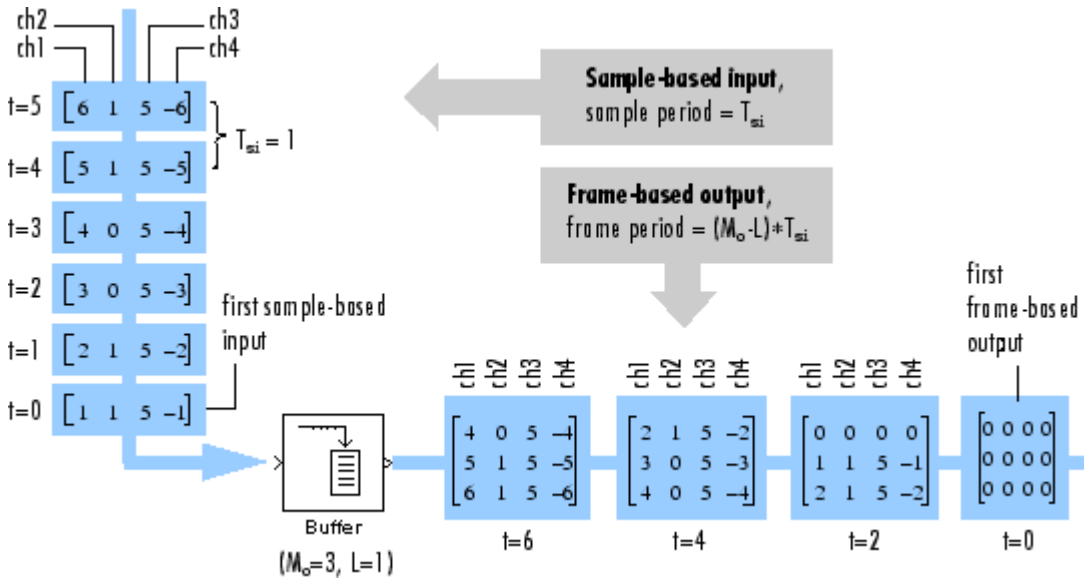
where $L < M_o$. For $0 \le L < M_o$, the number of new input samples that the block acquires before propagating the buffered data to the output is the difference between the Output buffer size and Buffer overlap ,

or $M_o - L$.

The output frame period is $(M_o - L)T_{si}$, which is equal to the input sequence sample period, T_{si}, when the Buffer overlap is $M_o - 1$. For $L < 0$, the block simply discards L input samples after the buffer fills,

and outputs the buffer with period $(M_o - L)T_{si}$, which is longer than the zero-overlap case.

In the model below, the block buffers a four-channel sample-based input using a Output buffer size of three and a Buffer overlap of one.



Notice tha t the input vectors do not begin appearing at the output until the second row of the second matrix. This is due to the block's latency. The Þrst o utput matrix (all zeros in this example) re ßects the block's Initial c onditions setting, while the Þrst row of zeros in the second output is a result of the one-sample overlap between consecutive output frames.

You can u se the rebuffer_delay function with a frame size of 1 to precisely compute the delay (in samples) for sample-based signals. For the prev ious example,

```
d = rebuffer_delay(1,3,1)
d =
     4
```

This agr ees with the four samples of delay (zeros) per channel shown in the prev ious Þgure.

Frame-Based Operation

In frame-based operation, the Buffer block redistributes the samples in the input frame to an output frame with a new size and rate. A sequence of M_i-by-N matrix inputs is buffered into a sequence of M_o-by-N frame-based matrix outputs, where M_o is the output frame size specified by the Output buffer size parameter. The output buffer size is the number of consecutive samples from the input frame that are buffered into the output frame. M_o can be greater or less than the input frame size, M_i. Each of the N input channels is buffered independently.

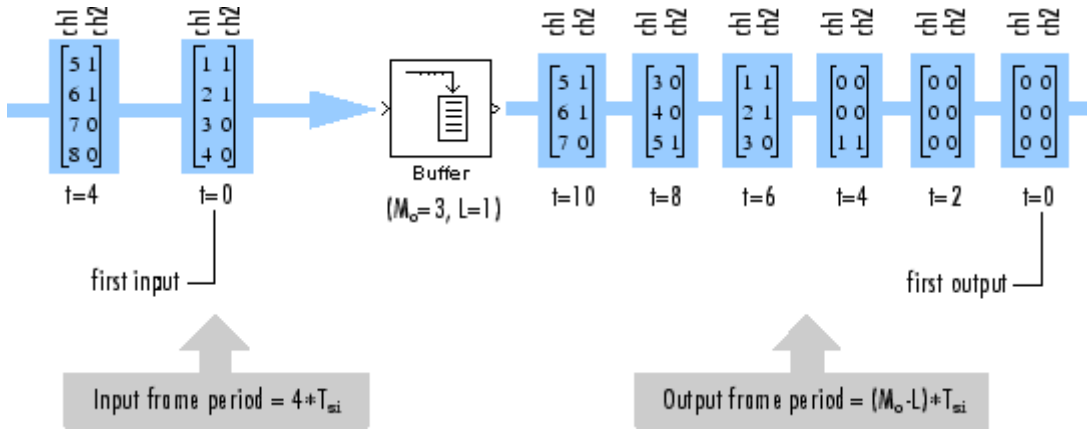
The Buffer overlap parameter, L, specifies the number of samples (rows) from the current output to repeat in the next output, where $L < M_o$. For $0 \le L < M_o$, the number of new input samples the block acquires before propagating the buffered data to the output is the difference between the Output buffer size and Buffer overlap, $M_o - L$.

The input frame period is $M_i T_{si}$, where T_{si} is the sample period. The output frame period is $(M_o - L)T_{si}$, which is equal to the sequence sample period when the Buffer overlap is $M_o - 1$. The output sample period is therefore related to the input sample period by

$$T_{so} = \frac{(M_o - L)T_{si}}{M_i}$$

Negative Buffer overlap values are not permitted.

In the model below, the block buffers a two-channel frame-based input using a Output buffer size of three and a Buffer overlap of one.



Notice that the sequence is delayed by eight samples, which is the latency of t he block in the Simulink multitasking mode for the parameter settings of this example. The Þrst eight output samples therefore adopt the value speci Þed for the Initial conditions , which is assumed here to be zero. Use the rebuffer_delay function to determine the block ’s latency for any combinati on of frame size and overlap.

Latency

Zero-tasking latency means that the Þrst input sample, received at t = 0, appears as the Þrst output sample. In the Simulink single-tasking mode, the Buffer block has zero-tasking latency for the following special cases:

- Scalar input and output ($M_o = M_i = 1$) with zero or negative Buffer overlap ($L \le 0$)
- Input frame size is an integer multiple of the output frame size

  $$M_i = kM_o$$

  where k is an integer with zero Buffer overlap ($L = 0$); notable cases of this include

- Any input frame size M_i with scalar output ($M_o = 1$) and zero Buffer overlap ($L = 0$)
- Equal input and output frame sizes ($M_o = M_i$) with zero Buffer overlap ($L = 0$)

For all cases of sample-based single-tasking operation other than those listed above, the Buffer block's buffer is initialized to the value(s) specified by the Initial conditions parameter, and the block reads from this buffer to generate the first D output samples, where

$$D = \begin{cases} M_o + L & (L \geq 0) \\ M_o & (L < 0) \end{cases}$$

The dimensions of the Initial conditions parameter depend on the Buffer overlap , L, and whether the input is single-channel or multichannel:

- When $L \neq 0$, the Initial conditions parameter must be a scalar.
- When $L = 0$, the Initial conditions parameter can be a scalar, or it can be a vector with the following constraints:
  - For single-channel inputs, the Initial conditions parameter can be a vector of length M_o if M_i is 1, or a vector of length M_i if M_o is 1.
  - For multichannel inputs, the Initial conditions parameter can be a vector of length $M_o * N$ if M_i is 1, or a vector of length $M_i * N$ if M_o is 1.

For frame-based single-tasking operation and all multitasking operation, use the rebuffer_delay function to compute the exact delay, in samples, that the Buffer block introduces for a given combination of buffer size and buffer overlap.

For general buffering between arbitrary frame sizes, the Initial conditions parameter must be a scalar value, which is then repeated

across all elements of the initial output(s). However, in the special case where the input is 1-by- N, and the block's output is therefore an M_o-by-N matrix, Initial conditions can be

- An M_o-by-N matrix
- A length- M_o vector to be repeated across all columns of the initial output(s)
- A scalar to be repeated across all elements of the initial output(s)

In the special case where the output is 1-by- N , which is the result of unbuffering an M_i-by-N frame-based matrix, Initial conditions can be

- A vector containing M_i samples to output sequentially for each channel during the Þrst M_i sample times
- A scalar to be repeated across all elements of the initial output(s)

Note For more information on latency and the Simulink tasking modes, see "Excess Algorithmic Delay (Tasking Latency)" and "Models with Multiple Sample Rates" in the Real-Time Workshop User's Guide .

Behavior in Enabled Subsystems

The Buffer block cannot be used in an enabled subsystem under the following conditions:

- In a multirate multitasking environment
- When the Buffer overlap parameter is set to a negative value

The Buffer block has an internal reservoir that temporarily stores data. When the Buffer block is used in an enabled subsystem, there is the possibility that the reservoir can overrun or underrun. The block implements safeguards against these occurrences.

Overrun occurs when more data enters into the buffer than it can hold. For example, consider buffering a sample-based input to a frame of size three with a buffer that accepts input every second and outputs every three seconds. If you place this buffer inside an enabled subsystem that is disabled every three seconds at t = 3s, t = 6s, and so on, the buffer accumulates data in its internal re servoir without being able to empty it. This condition results in overrun.

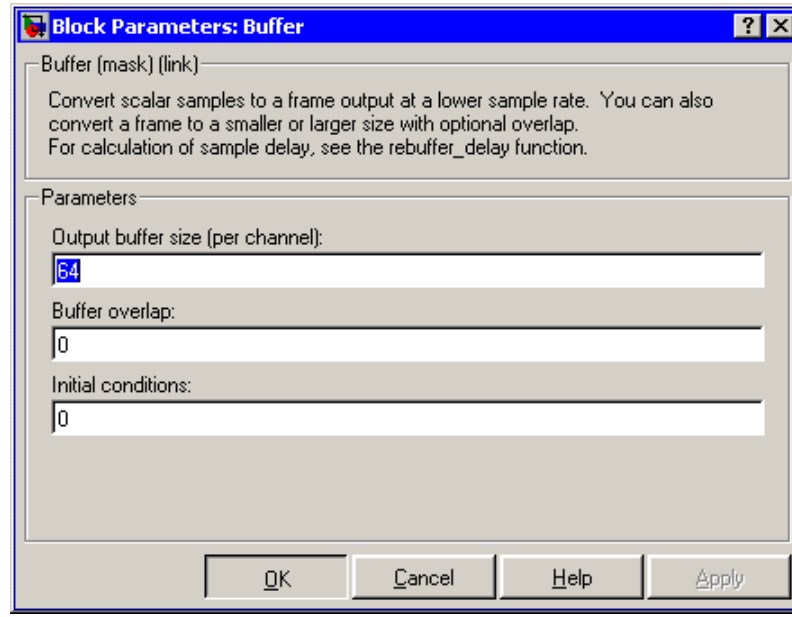
Underrun occurs when the buffer runs out of data to output. For example, again consider buffering a sample-based input to a frame size of three with a buffer that accepts i nput every second and outputs every three seconds. If you place this buffer inside an enabled subsystem that is disabled at t = 10s, t = 11s, t = 13s, t = 14s, t = 16s, and t = 17s, its internal reservoir becomes draine d, and there is no data to output at t = 18s. This condition results in underrun.

To protect from overrun and underr un, the Buffer block keeps a record of the amount of data in its internal reservoir. When the Buffer block reads data, the amount of data in its reservoir goes up. When data is output from the Buffer block, the amount of data in its reservoir goes down. To protect from overrun, the oldest samples in the reservoir are discarded whenever amount of data in the reservoir is larger than the actual buffer size. To protect from underrun, the most recent samples are repeated whenever an output is due and there is no data in the reservoir.

Examples

See “Frame Rebuffering Blocks” and “Converting Frame Status” in the Signal Processing Blockset User’s Guide .

Dialog Box



Output buffer size

Specify the number of consecutive samples, M_o, from each channel to buffer into the output frame.

Buffer overlap

Specify the number of samples, L, by which consecutive output frames overlap.

Initial conditions

Specify the value of the block's initial output for cases of nonzero latency; a scalar, vector, or matrix.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Delay Line	Signal Processing Blockset
Unbuffer	Signal Processing Blockset
rebuffer_delay	Signal Processing Blockset

See “Converting Sample and Frame Rates” and “Converting Frame Status” for more information.

Purpose

Compute estimate of autoregressive (AR) model parameters using Burg method

Library

Estimation / Parametric Estimation

dspparest3

Description

The Burg AR Estimator block uses the Burg method to Þt an autoregressive (AR) model to the input data by minimizing (least squares) the forward and backward pre diction errors while constraining the AR parameters to satisfy th e Levinson-Durb in recursion.

The input is a sample-based vector (row, column, or 1-D) or frame-based vector (column only) representing a frame of consecutive time samples from a single-channel signal, which is assumed to be the output of an AR system driven by white noise. T he block computes the normalized estimate of the AR system parameters, A(z), independently for each successive input frame.

$$H(z) = \frac{G}{A(z)} = \frac{G}{1 + a(2)z^{-1} + \ldots + a(p+1)z^{-p}}$$

When you select the Inherit estimation order from input dimensions parameter, the order, p, of the all-pole model is one less that the length of the input vector. Otherwise, the order is the value speciÞed by the Estimation order parameter.

The Output(s) parameter allows you to sele ct between two realizations of the AR process:

- A — The top output, A, is a column vector of length p+1 with the same frame status as the input, and con tains the normalized estimate of the AR model polynomial coef Þcients in descending powers of z.

  [1 a(2) ... a(p+1)]

- K — The top output, K, is a column vector of length p with the same frame status as the input, and contains the reflection coefficients (which are a secondary result of the Levinson recursion).
- A and K — The block outputs both realizations.

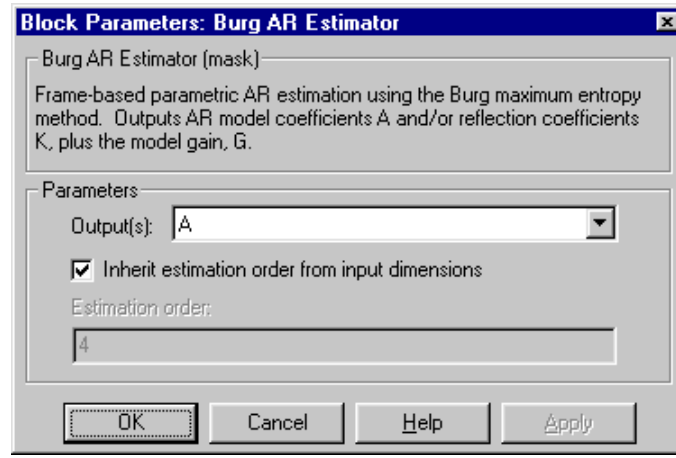
The scalar gain, G, is provided at the bottom output (G).

The following table compares the features of the Burg AR Estimator block to the Covariance AR Estimator, Modified Covariance AR Estimator, and Yule-Walker AR Estimator blocks.

	Burg AR Estimator	Covariance AR Estimator	Modified Covariance AR Estimator	Yule-Walker AR Estimator
Characteristics	Does not apply window to data	Does not apply window to data	Does not apply window to data	Applies window to data
	Minimizes the forward and backward prediction errors in the least squares sense, with the AR coefficients constrained to satisfy the L-D recursion	Minimizes the forward prediction error in the least squares sense	Minimizes the forward and backward prediction errors in the least squares sense	Minimizes the forward prediction error in the least squares sense (also called "autocorrelation method")
Advantages	Always produces a stable model			Always produces a stable model

	Burg AR Estimator	Covariance AR Estimator	Modified Covariance AR Estimator	Yule-Walker AR Estimator
Disadvantages		May produce unstable models	May produce unstable models	Performs relatively poorly for short data records
Conditions for Nonsingularity		Order must be less than or equal to half the input frame size	Order must be less than or equal to 2/3 the input frame size	Because of the biased estimate, the autocorrelation matrix is guaranteed to positive-de Þnite, hence nonsingular

Dialog Box



Output(s)
: The realization to output, model coefficients, reflection coefficients, or both.

Inherit estimation order from input dimensions
: When selected, sets the estimation order p to one less than the length of the input vector.

Estimation order
: The order of the AR model, p. This parameter is enabled when you do not select Inherit estimation order from input dimensions.

References

Kay, S. M. Modern Spectral Estimation: Theory and Application. Englewood Cliffs, NJ: Prentice-Hall, 1988.

Marple, S. L., Jr., Digital Spectral Analysis with Applications. Englewood Cliffs, NJ: Prentice-Hall, 1987.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point
A	• Double-precision floating point • Single-precision floating point
G	• Double-precision floating point • Single-precision floating point

See Also

Burg Method	Signal Processing Blockset
Covariance AR Estimator	Signal Processing Blockset
Modified Covariance AR Estimator	Signal Processing Blockset

Yule-Walker AR Estimator

arburg

Signal Processing Blockset

Signal Processing Toolbox

Burg Method

Purpose

Compute parametric spectral estimate using Burg method

Library

Estimation / Power Spectrum Estimation

dspspect3

Description

The Burg Method block estimates the power spectral density (PSD) of the input frame using the Burg method. This method fits an autoregressive (AR) model to the signal by minimizing (least squares) the forward and backward prediction errors while constraining the AR parameters to satisfy the Levinson-Durbin recursion.

The input is a sample-based vector (row, column, or 1-D) or frame-based vector (column only) representing a frame of consecutive time samples from a single-channel signal. The block's output (a column vector) is the estimate of the signal's power spectral density at N_{fft} equally spaced frequency points in the range [0, F_s), where F_s is the signal's sample frequency.

When you select the **Inherit estimation order from input dimensions** parameter, the order of the all-pole model is one less that the input frame size. Otherwise, the order is the value specified by the **Estimation order** parameter. The spectrum is computed from the FFT of the estimated AR model parameters.

When you select the **Inherit FFT length from estimation order** parameter, N_{fft} is one greater than the estimation order. When you do not select **Inherit FFT length from estimation order**, N_{fft} is specified as a power of 2 by the **FFT length** parameter, and the block zero pads or wraps the input to N_{fft} before computing the FFT. The output is always sample based.

The Burg Method and Yule-Walker Method blocks return similar results for large frame sizes. The following table compares the features of the Burg Method block to the Covariance Method, Modified Covariance Method, and Yule-Walker Method blocks.

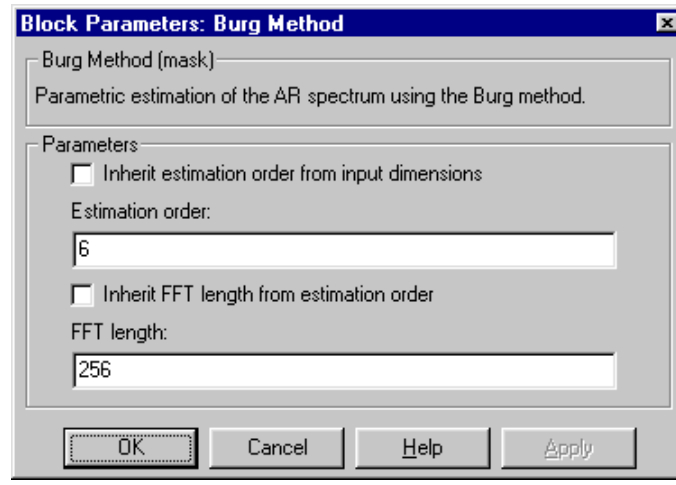
	Burg	Covariance	Modified Covariance	Yule-Walker
Characteristics	Does not apply window to data	Does not apply window to data	Does not apply window to data	Applies window to data
	Minimizes the forward and backward prediction errors in the least squares sense, with the AR coefficients constrained to satisfy the L-D recursion	Minimizes the forward prediction error in the least squares sense	Minimizes the forward and backward prediction errors in the least squares sense	Minimizes the forward prediction error in the least squares sense (also called "autocorrelation method")
Advantages	High resolution for short data records	Better resolution than Y-W for short data records (more accurate estimates)	High resolution for short data records	Performs as well as other methods for large data records
	Always produces a stable model	Able to extract frequencies from data consisting of p or more pure sinusoids	Able to extract frequencies from data consisting of p or more pure sinusoids	Always produces a stable model
			Does not suffer spectral line-splitting	

	Burg	Covariance	Modified Covariance	Yule-Walker
Disadvantages	Peak locations highly dependent on initial phase	May produce unstable models	May produce unstable models	Performs relatively poorly for short data records
	May suffer spectral line-splitting for sinusoids in noise, or when order is very large	Frequency bias for estimates of sinusoids in noise	Peak locations slightly dependent on initial phase	Frequency bias for estimates of sinusoids in noise
	Frequency bias for estimates of sinusoids in noise		Minor frequency bias for estimates of sinusoids in noise	
Conditions for Nonsingularity		Order must be less than or equal to half the input frame size	Order must be less than or equal to 2/3 the input frame size	Because of the biased estimate, the autocorrelation matrix is guaranteed to positive-de Þnite, hence nonsingular

Examples

The dspsacomp demo compares the Burg method with several other spectral estimation methods.

Dialog Box



Inherit estimation order from input dimensions

When selected, sets the estimation order to one less than the length of the input vector.

Estimation order

The order of the AR model. This parameter is enabled when you do not select Inherit estimation order from input dimensions.

Inherit FFT length from estimation order

When selected, the FFT length is one greater than the estimation order.

FFT length

Enter the number of data points on which to perform the FFT, N_{fft}. When N_{fft} is larger than the input frame size, each frame is zero-padded as needed. When N_{fft} is smaller than the input frame size, each frame is wrapped as needed. This parameter is enabled when you clear the Inherit FFT length from input dimensions check box.

References

Kay, S. M. Modern Spectral Estimation: Theory and Application. Englewood Cliffs, NJ: Prentice-Hall, 1988.

Orfanidis, S. J. Introduction to Signal Processing . Englewood Cliffs, NJ: Prentice-Hall, 1995.

Orfanidis, S. J. Optimum Signal Processing: An Introduction. 2nd ed. New York, NY: Macmillan, 1985.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point
Output	• Double-precision ßoating point • Single-precision ßoating point

See Also

Burg AR Estimator	Signal Processing Blockset
Covariance Method	Signal Processing Blockset
Modi Þed Covariance Method	Signal Processing Blockset
Short-Time FFT	Signal Processing Blockset
Yule-Walker Method	Signal Processing Blockset
pburg	Signal Processing Toolbox

See “Power Spectrum Estimatio n” for related information.

Purpose

Generate error when input signal does or does not match selected attributes exactly

Library

Signal Management / Signal Attributes

dspsigattribs

Description

The Check Signal Attributes block terminates the simulation with an error when the input characteristics differ from those specified by the block parameters.

When the Error when input parameter is set to Does not match attributes exactly, the block generates an error if the input fails to match any of the attributes specified by the other parameters. Only signals that possess all of the specified attributes are propagated to the output unaltered, and do not cause the block to generate an error.

When the Error when input parameter is set to Matches attributes exactly, the block generates an error only when the input possesses all attributes specified by the other parameters. Signals that do not possess all of the specified attributes are propagated to the output unaltered, and do not cause the block to generate an error.

Signal Attributes

The Check Signal Attributes block can test for up to five different signal attributes, as specified by the following parameters. When you select the Ignore in any parameter, the block does not check the signal for the corresponding attribute. For example, when Complexity is set to Ignore, neither real nor complex inputs cause the block to generate an error. The attributes are:

- Complexity

  Checks whether the signal is real or complex. (Note that this information can be displayed in a model by attaching a Probe block with Probe complex signal selected. Alternatively, in the model window, from the Format menu, point to Port/Signal Displays and select Port Data Types.)

- Frame status

  Checks whether the signal is frame based or sample based. (Note that the Simulink environment disp lays sample-based signals using a single line and displays frame-ba sed signals using a double line.)

- Dimensionality

  Checks the dimension of signal for compliance (Is...) or noncompliance (Is not...) with the attributes in the subordinate Dimension menu, which are shown in the following table. M and N are positive integers unless otherwise indicated.

Dimensions	Is...	Is not...
1-D	1-D vector, 1-D scalar	M-by-N matrix, 1-by-N matrix (row vector), M-by-1 matrix (column vector), 1-by-1 matrix (2-D scalar)
2-D	M-by-N matrix, 1-by-N matrix (row vector), M-by-1 matrix (column vector), 1-by-1 matrix (2-D scalar)	1-D vector, 1-D scalar
Scalar (1-D or 2-D)	1-D scalar, 1-by-1 matrix (2-D scalar)	1-D vector with length>1, M-by-N matrix with M>1 and/or N>1
Vector (1-D or 2-D)	1-D vector, 1-D scalar, 1-by-N matrix (row vector), M-by-1 matrix (column vector), 1-by-1 matrix (2-D scalar) Vector (1-D or 2-D) or scalar	M-by-N matrix with M>1 and N>1
Row Vector (2-D)	1-by-N matrix (row vector), 1-by-1 matrix (2-D scalar) Row vector (2-D) or scalar	1-D vector, 1-D scalar, M-by-N matrix with M>1

Dimensions	Is...	Is not...
Column Vector (2-D)	M-by-1 matrix (column vector), 1-by-1 matrix (2-D scalar) Column vector (2-D) or scalar	1-D vector, 1-D scalar, M-by-N matrix with N>1
Full matrix	M-by-N matrix with M>1 and N>1	1-D vector, 1-D scalar, 1-by-N matrix (row vector), M-by-1 matrix (column vector), 1-by-1 matrix (2-D scalar)
Square matrix	M-by-N matrix with M=N, 1-D scalar, 1-by-1 matrix (2-D scalar	M-by-N matrix with M≠N, 1-D vector, 1-by-N matrix (row vector), M-by-1 matrix (column vector)

If, in the model window, from the Format menu, you point to Port/Signal Displays and select Signal Dimensions, Simulink displays the size of a 1-D vector signal as an unbracketed integer, and displays the dimension of a 2-D signal as a pair of bracketed integers, [MxN]. Simulink does not display any size information for a 1-D or 2-D scalar signal. Dimension information for a signal can also be displayed in a model by attaching a Probe block with Probe signal dimensions selected.

- Data type

  Checks the signal data type for compliance (Is...) or noncompliance (Is not...) with the attributes in the subordinate General data type menu, which are listed in the following table. Any of the specific data types listed in the Is... column can be individually selected from the subordinate Specific data type menu.

General Data Type	Is...	Is not...
Boolean	boolean	single, double, uint8, int8, uint16, int16, uint32, int32, fixed-point

General Data Type	Is...	Is not...
Floating point	single, double	boolean, uint8, int8, uint16, int16, uint32, int32, fixed-point
Fixed point	Þxed point	boolean, uint8, int8, uint16, int16, uint32, int32, single, double
Integer	Signed integer int8 , int16 , int32 Unsigned integer uint8 , uint16 , uint32	boolean, single, double

To display data type information, in your model window, from the Format menu, point to Port/Signal Displays and select Port Data Types .

- Sample time

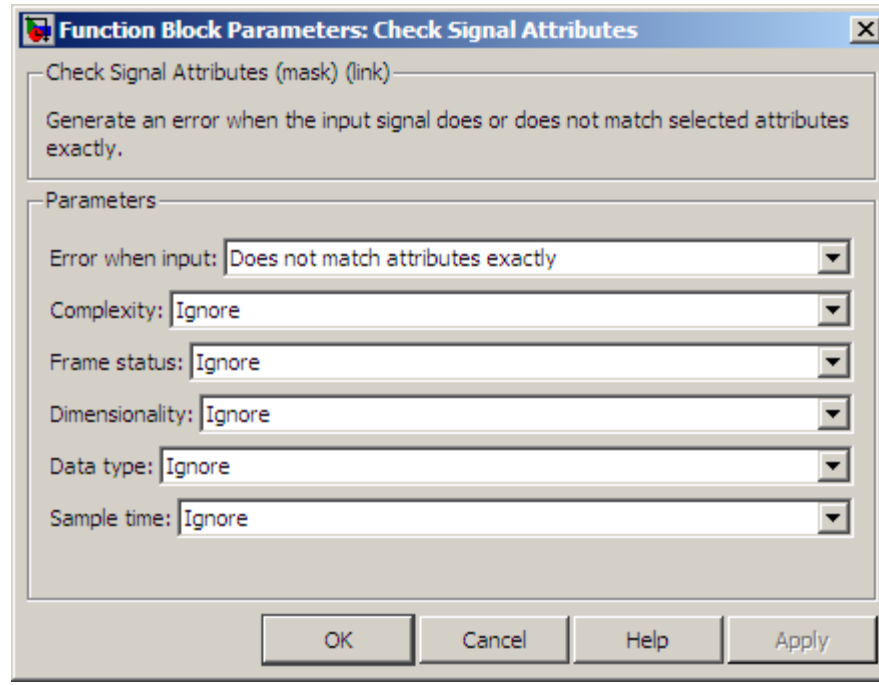
  Checks whether the signal is discrete time or continuous time. (If, from the Format menu, you point to Sample Time Display and select Colors , Simulink displays continuous-time signal lines in black or grey and discrete-time sig nal lines in colors corresponding to the relative rate.)

  When a Probe block with Probe sample time enabled is attached to a continuous-time signal, the block icon displays the string Ts:[0 To] , where To is the sample time offset. Valid values of To for continuous-time signals are 0 and 1. If To is 0, updates occur at every major and minor time step; if To is 1, updates occur only at major time steps, and the sample time is said to be *fi*xed in minor time step .

  When a Probe block with Probe sample time enabled is attached to a discrete-time signal, the block icon displays the string Ts:[Ts To]

for sample-based signals, and displays Tf:[Tf To] for frame-based signals. Ts and Tf are the positive sample period and frame period, respectively. To is the offset, such that 0 ≤ *offset* < *period*. Frame-based signals are almost always discrete time. For more information on the Simulink Probe block, see the Probe reference page.

Dialog Box



Error when input

Specifies whether the block generates an error when the input does not possess all of the required attributes (Does not match attributes exactly), or when the input does possess all of the required attributes (Matches attributes exactly).

Complexity

The complexity for which the input should be checked, Real or Complex. When you select Ignore from the list, the block does not check the input's complexity.

Frame status

The frame status for which the input should be checked, Sample-based or Frame-based. When you select Ignore from the list, the block does not check the input's frame status.

Dimensionality

Specifies whether the input should be checked for compliance (Is...) or noncompliance (Is not...) with the attributes in the subordinate Dimensions menu. When you select Ignore from the list, the block does not check the input's dimensionality.

Data type

Specifies whether the input should be checked for compliance (Is...) or noncompliance (Is not...) with the attributes in the subordinate General data type menu. When you select Ignore from the list, the block does not check the input's data type.

Sample time

The sample time for which the input should be checked, Discrete or Continuous. When you select Ignore from the list, the block does not check the input's sample time.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8, 16, and 32-bit signed integers

Port	Supported Data Types
	• 8, 16, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • Boolean • 8, 16, and 32-bit signed integers • 8, 16, and 32-bit unsigned integers

See Also

Buffer	Signal Processing Blockset
Convert 1-D to 2-D	Signal Processing Blockset
Convert 2-D to 1-D	Signal Processing Blockset
Data Type Conversion	Simulink
Frame Status Conversion (Obsolete)	Signal Processing Blockset
Inherit Complexity	Signal Processing Blockset
Probe	Simulink
Reshape	Simulink
Submatrix	Signal Processing Blockset

Chirp

Purpose

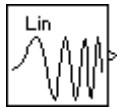
Generate swept-frequency cosine (chirp) signal

Library

Signal Processing Sources

dspsrcs4

Description



The Chirp block outputs a swept-frequency cosine (chirp) signal with unity amplitude and continuous phase. To specify the desired output chirp signal, you must define its instantaneous frequency function, also known as the output frequency sweep. The frequency sweep can be linear, quadratic, or logarithmic, and repeats once every Sweep time by default. See other sections of this reference page for more details about the block.

Sections of This Reference Page

- Variables Used in This Reference Page on page 2-139
- “Setting the Output Frame Status” on page 2-139
- “Shaping the Frequency Sweep by Setting Frequency Sweep and Sweep Mode” on page 2-140
- “Unidirectional and Bidirectional Sweep Modes” on page 2-141
- “Setting Instantaneous Frequency Sweep Values” on page 2-142
- “Block Computation Methods” on page 2-143
- “Cautions Regarding the Swept Cosine Sweep” on page 2-146
- “Dialog Box” on page 2-148
- “Examples” on page 2-150
- “Supported Data Types” on page 2-161
- “See Also” on page 2-161

Variables Used in This Reference Page

f_0	Initial frequency parameter (Hz)
$f_i(t_g)$	Target frequency parameter (Hz)
t_g	Target time parameter (seconds)
T_{sw}	Sweep time parameter (seconds)
ϕ_0	Initial phase parameter (radians)
$\psi(t)$	Phase of the chirp signal (radians)
$f_i(t)$	User-specified output instantaneous frequency function (Hz); user-specified sweep
$f_{i(actual)}(t)$	Actual output instantaneous frequency function (Hz); actual output sweep
$y_{chirp}(t)$	Output chirp function

Setting the Output Frame Status

Use Samples per frame parameter to set the block's output frame status, as summarized in the following table. The Sample time parameter sets the sample time of both sample- and frame-based outputs.

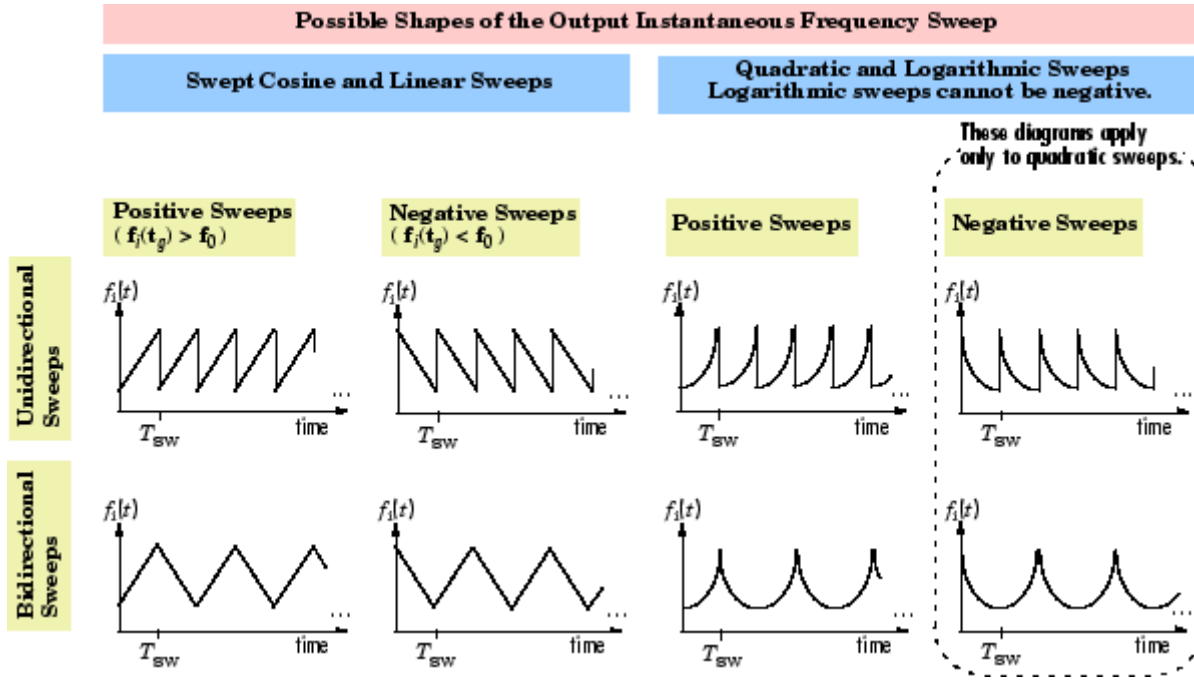
Setting of Samples Per Frame Parameter	Output Frame Status
1	Sample based
n (any integer greater than 1)	Frame based, frame size n

Shaping the Frequency Sweep by Setting Frequency Sweep and Sweep Mode

The basic shape of the output instantaneous frequency sweep, $f_i(t)$, is set by the Frequency sweep and Sweep mode parameters, described in the following table.

Parameters for Setting Sweep Shape	Possible Setting	Parameter Description
Frequency sweep	Linear Quadratic Logarithmic Swept cosine	Determines whether the sweep frequencies vary linearly, quadratically, or logarithmically. Linear and swept cosine sweeps both vary linearly.
Sweep mode	Unidirectional Bidirectional	Determines whether the sweep is unidirectional or bidirectional. For details, see “Unidirectional and Bidirectional Sweep Modes” on page 2-141

The following diagram illustrates the possible shapes of the frequency sweep that you can obtain by setting the Frequency sweep and Sweep mode parameters.



For information on how to set the frequency values in your sweep, see “Setting Instantaneous Frequency Sweep Values” on page 2-142.

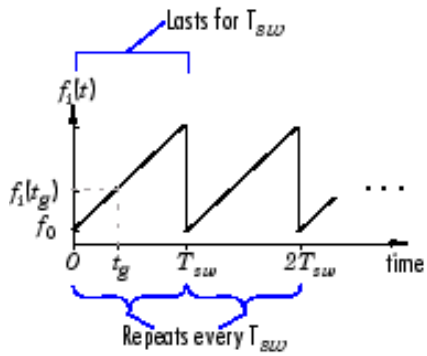
Unidirectional and Bidirectional Sweep Modes

The Sweep mode parameter determines whether your sweep is unidirectional or bidirectional, which affects the shape of your output frequency sweep (see “Shaping the Frequency Sweep by Setting Frequency Sweep and Sweep Mode” on page 2-140). The following table describes the characteristics of unidirectional and bidirectional sweeps.

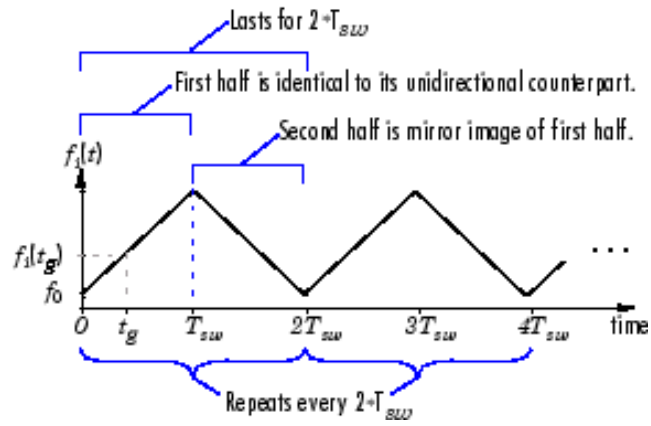
Sweep Mode Parameter Settings	Sweep Characteristics
Unidirectional	• Lasts for one Sweep time , T_{sw} • Repeats once every T_{sw}
Bidirectional	• Lasts for twice the Sweep time , $2*T_{sw}$ • Repeats once every $2*T_{sw}$ • First half is identical to its unidirectional counterpart. • Second half is a mirror image of the first half.

The following diagram illustrates a linear sweep in both sweep modes. For information on setting the frequency values in your sweep, see “Setting Instantaneous Frequency Sweep Values” on page 2-142.

Unidirectional Linear Sweep



Bidirectional Linear Sweep



Setting Instantaneous Frequency Sweep Values

Set the following parameters to tune the frequency values of your output frequency sweep. Note that b ecause this is a source block, the

simulation pauses while the block dialog box is open. You must close the dialog box by clicking OK to resume the simulation.

- Initial frequency (Hz), f_0
- Target frequency (Hz), $f_i(t_g)$
- Target time (seconds), t_g

The following table summarizes the sweep values at specific times for all Frequency sweep settings. For information on the formulas used to compute sweep values at other times, see "Block Computation Methods" on page 2-143.

Instantaneous Frequency Sweep Values

Frequency Sweep	Sweep Value at $t = 0$	Sweep Value at $t = t_g$	Time when Sweep Value Is Target Frequency, $f_i(t_g)$
Linear	f_0	$f_i(t_g)$	t_g
Quadratic	f_0	$f_i(t_g)$	t_g
Logarithmic	f_0	$f_i(t_g)$	t_g
Swept cosine	f_0	$2f_i(t_g) - f_0$	$t_g/2$

Block Computation Methods

The Chirp block uses one of two formulas to compute the block output, depending on the Frequency Sweep parameter setting. For details, see the following sections:

- "Equations for Output Computation" on page 2-144
- "Output Computation Method for Linear, Quadratic, and Logarithmic Frequency Sweeps" on page 2-145

- “Output Computation Method for Swept Cosine Frequency Sweep” on page 2-146

Equations for Output Computation

The following table shows the equations used by the block to compute the user-specified output frequency sweep, $f_i(t)$, the block output, $y_{chirp}(t)$, and the actual output frequency sweep, $f_{i(actual)}(t)$. The only time the user-specified sweep is not the actual output sweep is when the Frequency sweep parameter is set to Swept cosine.

Note The following equations apply only to unidirectional sweeps in which $f_i(0) < f_i(t_g)$. To derive equations for other cases, you might find it helpful to examine the following table and the diagram in “Shaping the Frequency Sweep by Setting Frequency Sweep and Sweep Mode” on page 2-140.

The table below contains the following variables:

- $f_i(t)$ — the user-specified frequency sweep
- $f_{i(actual)}(t)$ — the actual output frequency sweep, usually equal to $f_i(t)$
- y(t) — the Chirp block output
- $\psi(t)$ — the phase of the chirp signal, where $\psi(0) = 0$, and $2\pi f_i(t)$ is the derivative of the phase

$$f_i(t) = \frac{1}{2\pi} \cdot \frac{d\psi(t)}{dt}$$

- ϕ_0 — the Initial phase parameter value, where $y_{chirp}(0) = \cos(\phi_0)$

Equations Used by the Chirp Block for Unidirectional Positive Sweeps

Frequency Sweep	Block Output Chirp Signal	User-Specified Frequency Sweep, $f_i(t)$	β	Actual Frequency Sweep, $f_{i(actual)}(t)$
Linear	$y(t) = \cos(\psi(t) + \phi_0)$	$f_i(t) = f_0 + \beta t$	$\beta = \frac{f_i(t_g) - f_0}{t_g}$	$f_{i(actual)}{}^{(t)} = f_i^{(t)}$
Quadratic	Same as Linear	$f_i(t) = f_0 + \beta t^2$	$\beta = \frac{f_i(t_g) - f_0}{2}$	$f_{i(actual)}{}^{(t)} = f_i^{(t)}$
Logarithmic	Same as Linear	$F_i(t) = f_0 \left(\frac{f_i(t_g)}{f_0} \right)^{\frac{t}{t_g}}$ Where $f_i(t_g) > f_0 > 0$	N/A	$f_{i(actual)}(t) = f_i(t)$
Swept cosine	$y(t) = \cos(2\pi f_i(t)t + \phi_0)$	Same as Linear	Same as Linear	$f_{i(actual)}(t) = f_i(t) + \beta t$

Output Computation Method for Linear, Quadratic, and Logarithmic Frequency Sweeps

The derivative of the phase of a chirp function gives the instantaneous frequency of the chirp function. The Chirp block uses this principle to calculate the chirp output when the Frequency Sweep parameter is set to Linear , Quadratic , or Logarithmic .

$$y_{chirp}(t) = \cos(\psi(t) + \phi_0)$$

Linear, quadratic, or logarithmic chirp signal with phase $\psi(t)$

$$f_i(t) = \frac{1}{2\pi} \cdot \frac{d\psi(t)}{dt}$$

Phase derivative is instantaneous frequency

For instance, if you want a chirp signal with a linear instantaneous frequency sweep, you should set the Frequency Sweep parameter to

Linear , and tune the linear sweep values by setting other parameters appropriately. Note that because this is a source block, the simulation pauses while the block dialog box is open. You must close the dialog box by clicking OK to resume the simulation. The block outputs a chirp signal, the phase derivative of which is the specified linear sweep. This ensures that the instantaneous frequency of the output is the linear sweep you desired. For equations describing the linear, quadratic, and logarithmic sweeps, see "Equations for Output Computation" on page 2-144.

Output Computation Method for Swept Cosine Frequency Sweep

To generate the swept cosine chirp signal, the block sets the swept cosine chirp output as follows.

$$y_{chirp}(t) = \cos(\psi(t) + \phi_0) = \cos(2\pi f_i(t)t + \phi_0)$$

Swept cosine chirp output (Instantaneous frequency equation, shown above, does not hold.)

Note that the instantaneous frequency equation, shown above, does not hold for the swept cosine chirp, so the user-defined frequency sweep, $f_i(t)$, is not the actual output frequency sweep, $f_{i(actual)}(t)$, of the swept cosine chirp. Thus, the swept cosine output might not behave as you expect. To learn more about swept cosine chirp behavior, see "Cautions Regarding the Swept Cosine Sweep" on page 2-146 and "Equations for Output Computation" on page 2-144.

Cautions Regarding the Swept Cosine Sweep

When you want a linearly swept chirp signal, we recommend you use a linear frequency sweep. Though a swept cosine frequency sweep also yields a linearly swept chirp signal, the output might have unexpected frequency content. For details, see the following two sections.

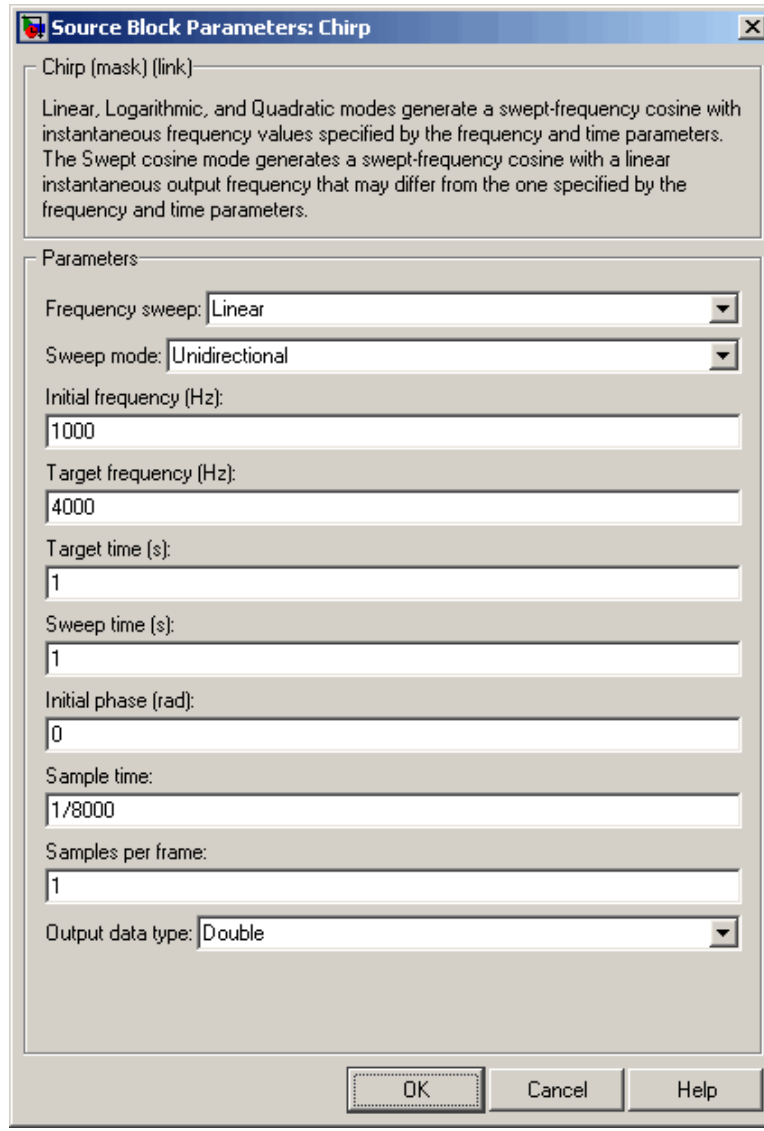
Swept Cosine Instantaneous Output Frequency at the Target Time is not the Target Frequency

The swept cosine sweep value at the Target time is not necessarily the Target frequency. This is because the user-specified sweep is not the actual frequency sweep of the swept cosine output, as noted in “Output Computation Method for Swept Cosine Frequency Sweep” on page 2-146. See the table Instantaneous Frequency Sweep Values on page 2-143 for the actual value of the swept cosine sweep at the Target time.

Swept Cosine Output Frequency Content May Greatly Exceed Frequencies in the Sweep

In Swept cosine mode, you should not set the parameters so that 1/Tsw is very large compared to the values of the Initial frequency and Target frequency parameters. In such cases, the actual frequency content of the swept cosine sweep might be closer to 1/Tsw, far exceeding the Initial frequency and Target frequency parameter values.

Dialog Box


Source Block Parameters: Chirp
Chirp (mask) (link)
Linear, Logarithmic, and Quadratic modes generate a swept-frequency cosine with instantaneous frequency values specified by the frequency and time parameters. The Swept cosine mode generates a swept-frequency cosine with a linear instantaneous output frequency that may differ from the one specified by the frequency and time parameters.
Parameters
Frequency sweep: Linear
Sweep mode: Unidirectional
Initial frequency (Hz):
1000
Target frequency (Hz):
4000
Target time (s):
1
Sweep time (s):
1
Initial phase (rad):
0
Sample time:
1/8000
Samples per frame:
1
Output data type: Double
OK
Cancel
Help

Opening this dialog box causes a running simulation to pause. See "Changing Source Block Parameters During Simulation" in the online Simulink documentation for details.

Frequency sweep

The type of output instantaneous frequency sweep, $f_i(t)$: Linear , Logarithmic , Quadratic , or Swept cosine .

Sweep mode

The directionality of the chirp signal: Unidirectional or Bidirectional .

Initial frequency (Hz)

For Linear , Quadratic , and Swept cosine sweeps, the initial frequency, f_0, of the output chirp signal. For Logarithmic sweeps, Initial frequency is one less than the actual initial frequency of the sweep. Also, when the sweep is Logarithmic , you must set the Initial frequency to be less than the Target frequency . Tunable.

Target frequency (Hz)

For Linear , Quadratic , and Logarithmic sweeps, the instantaneous frequency, $f_i(t_g)$, of the output at the Target time , t_g. For a Swept cosine sweep, Target frequency is the instantaneous frequency of the output at half the Target time , $t_g/2$. When Frequency sweep is Logarithmic , you must set the Target frequency to be greater than the Initial frequency . Tunable.

Target time (s)

For Linear , Quadratic , and Logarithmic sweeps, the time, t_g, at which the Target frequency , $f_i(t_g)$, is reached by the sweep. For a Swept cosine sweep, Target time is the time at which the sweep reaches $2f_i(t_g) - f_0$. You must set Target time to be no greater than Sweep time , $T_{sw} \geq t_g$. Tunable.

Sweep time (s)

In Unidirectional Sweep mode , the Sweep time , T_{sw}, is the period of the output frequency sweep. In Bidirectional Sweep

mode , the Sweep time is half the period of the output frequency sweep. You must set Sweep time to be no less than Target time , $T_{sw} \geq t_g$. Tunable.

Initial phase (rad)

The phase, ϕ_0 , of the cosine output at t=0; $y_{chirp}(t) = \cos(\phi_0)$. Tunable.

Sample time

The sample period, T_s, of the output. The output frame period is $M_o * T_s$.

Samples per frame

The number of samples, M_o, to buffer into each output frame. When the value of this parameter is 1, the block outputs a sample-based signal.

Output data type

The data type of the output, single-precision or double-precision.

Examples

The Þrst few examples demonstrate how to use the Chirp block's main parameters, how to view the output in the time domain, and how to view the output spectrogram:

- “Example 1: Setting a Final Frequency Value for Unidirectional Sweeps” on page 2-151
- “Example 2: Bidirectiona l Sweeps” on page 2-154
- “Example 3: When Sweep Time is Greater Than Target Time” on page 2-156

Examples 4 and 5 illustrate Chirp block settings that might produce unexpected outputs:

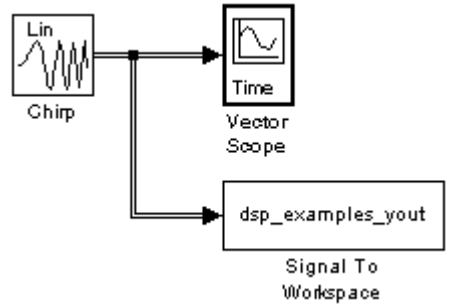
- “Example 4: Output Sweep with Negative Frequencies” on page 2-158
- “Example 5: Output Sweep with Frequencies Greater Than Half the Sampling Frequency” on page 2-159

Example 1: Setting a Final Frequency Value for Unidirectional Sweeps

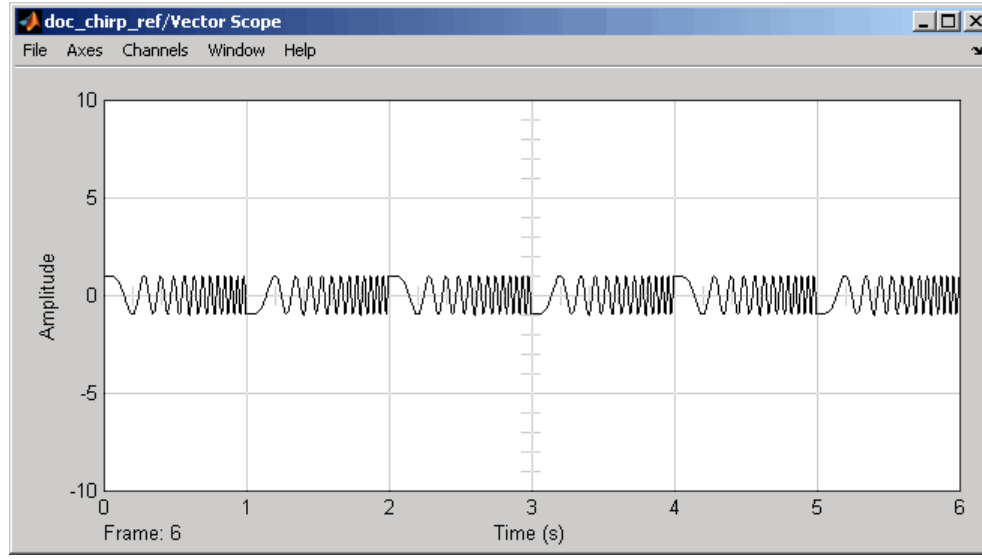
Often times, you might want a unidirectional sweep for which you know the initial and final frequency values. You can specify the final frequency of a unidirectional sweep by setting Target time equal to Sweep time, in which case the Target frequency becomes the final frequency in the sweep. The following model demonstrates this method.

This technique might not work for swept cosine sweeps. For details, see “Cautions Regarding the Swept Cosine Sweep” on page 2-146.

Open the Example 1 model by typing `doc_chirp_ref` at the MATLAB command line. You can also rebuild the model yourself; see the following list for model parameter settings (leave unlisted parameters in their default states).

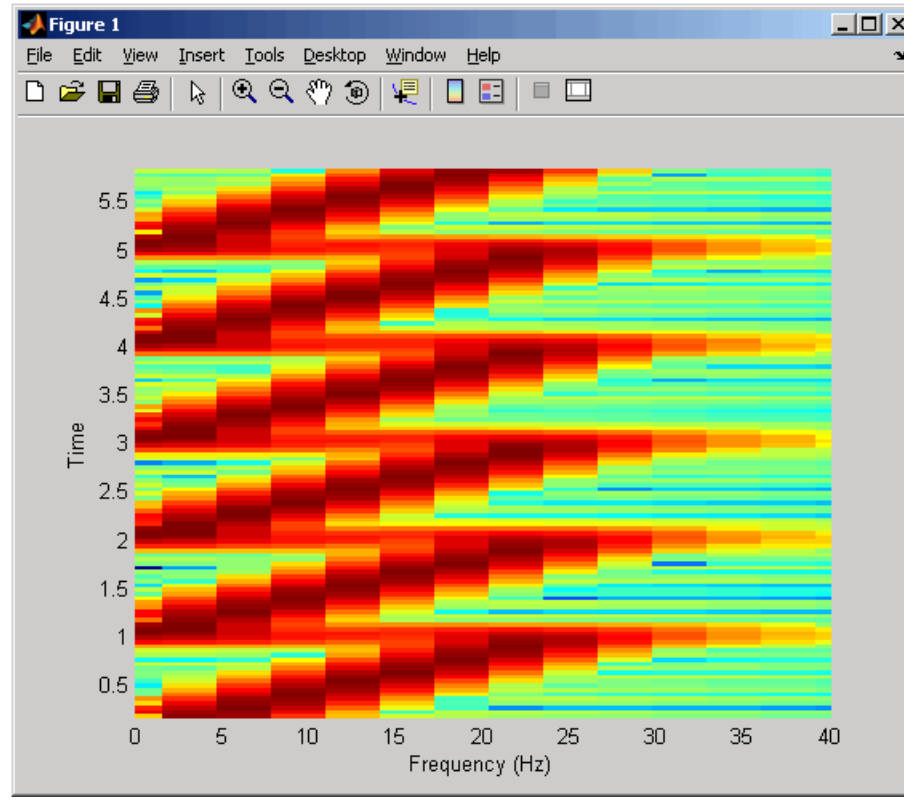


Since Target time is set to equal Sweep time (1 second), the Target frequency (25 Hz) is the final frequency of the unidirectional sweep. Run your model to see the time domain output:



Type the following command to view the chirp output spectrogram:

```
spectrogram(dsp_examples_yout,hamming(128),...
        110,[0:.01:40],400)
```



Chirp Block Parameters for Example 1	
Frequency sweep	Linear
Sweep mode	Unidirectional
Initial frequency	0
Target frequency	25
Target time	1
Sweep time	1

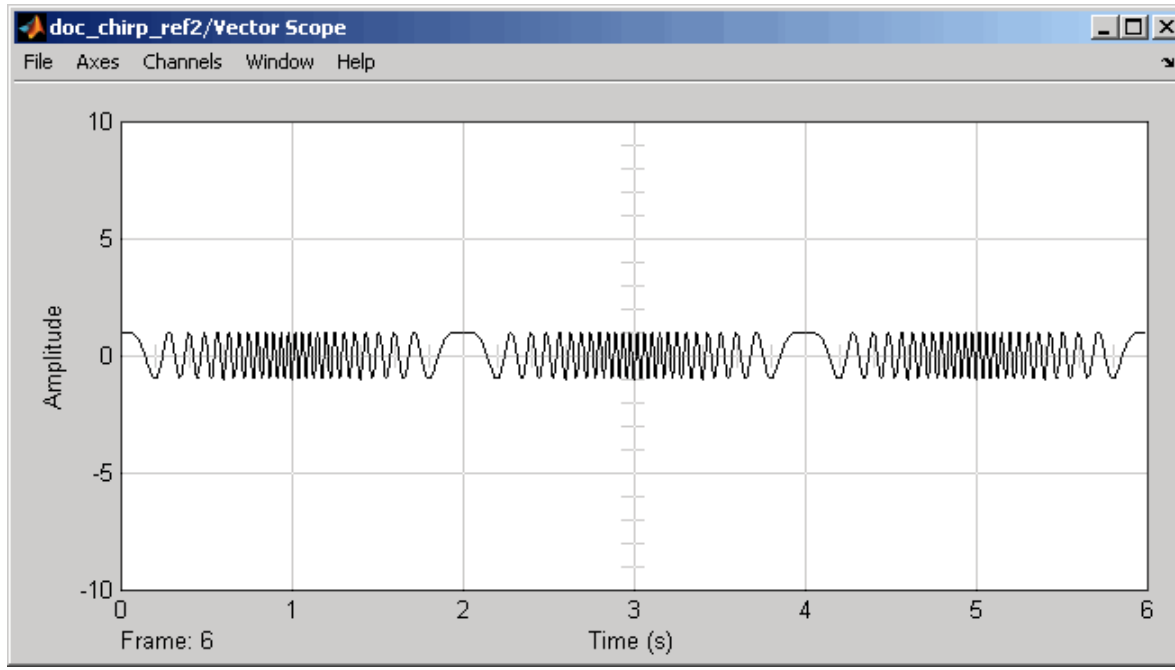
Initial phase	0
Sample time	1/400
Samples per frame	400
Vector Scope Block Parameters for Example 1	
Input domain	Time
Time display span	6
Signal To Workspace Block Parameters for Example 1	
Variable name	dsp_examples_yout
Configuration Dialog Parameters for Example 1	
Stop time	5

Example 2: Bidirectional Sweeps

Change the Sweep mode parameter in the Example 1 model to Bidirectional, and leave all other parameters the same to view the following bidirectional chirp. Note that in the bidirectional sweep, the period of the sweep is twice the Sweep time (2 seconds), whereas it was one Sweep time (1 second) for the unidirectional sweep in Example 1.

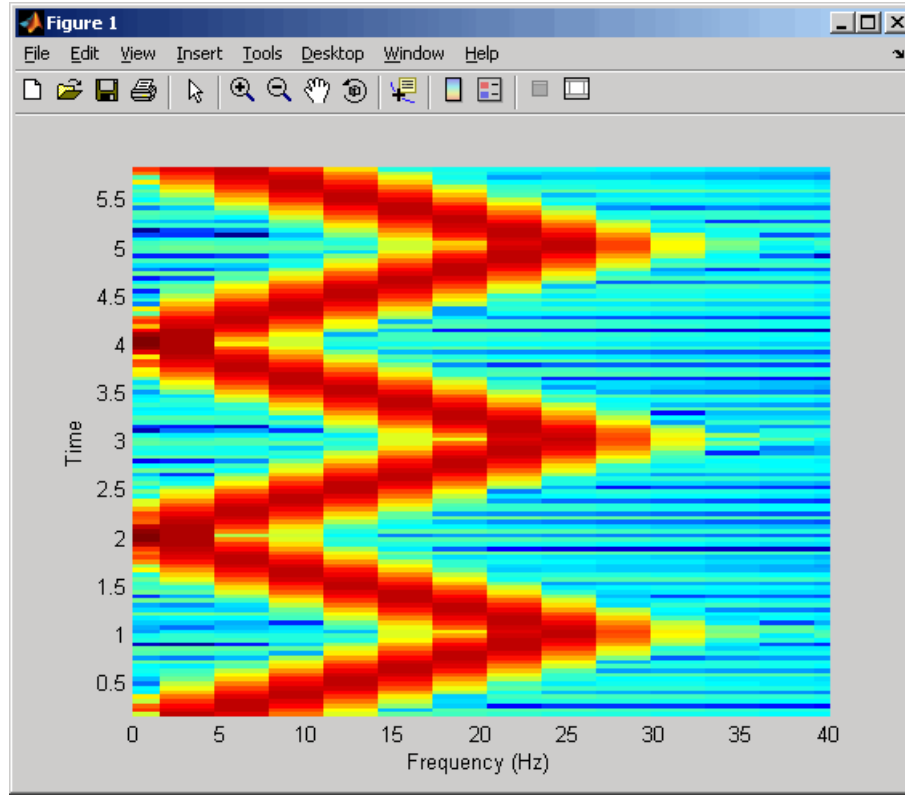
Open the Example 2 model by typing doc_chirp_ref2 at the MATLAB command line.

Run your model to see the time domain output:



Type the following command to view the chirp output spectrogram:

```
spectrogram(dsp_examples_yout,hamming(128),...
        110,[0:.01:40],400)
```



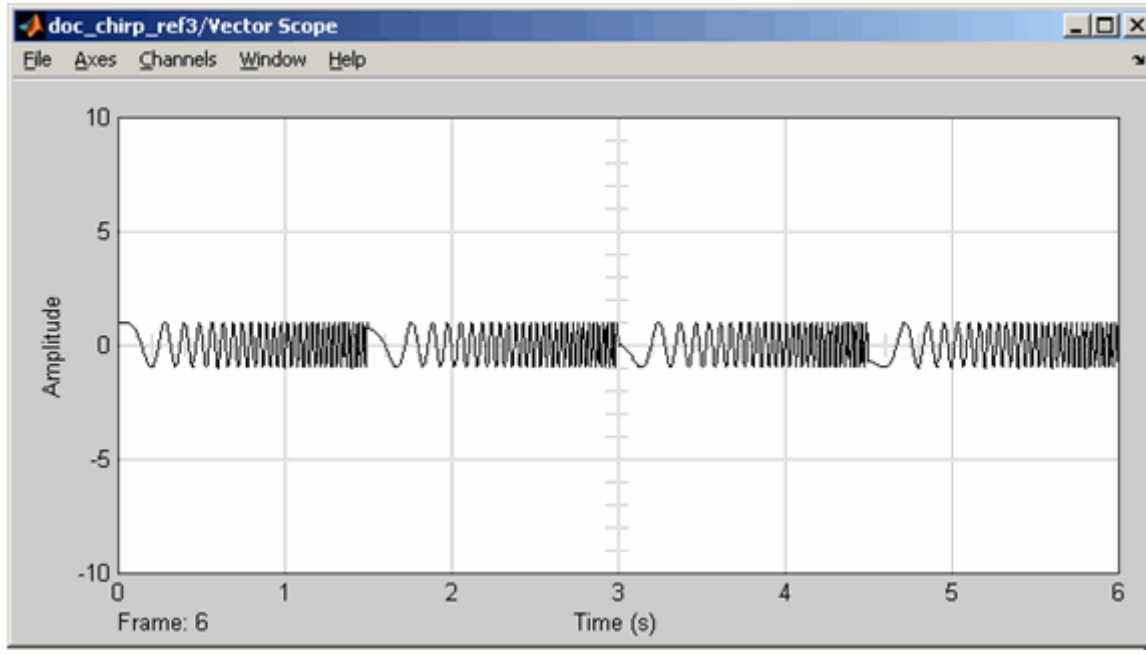
Example 3: When Sweep Time is Greater Than Target Time

Setting Sweep time to 1.5 and leaving the rest of the parameters as in the Example 1 model gives the following output. The sweep still reaches the Target frequency (25 Hz) at the Target time (1 second), but since Sweep time is greater than Target time, the sweep continues on its linear path until one Sweep time (1.5 seconds) is traversed.

Unexpected behavior might arise when you set Sweep time greater than Target time; see “Example 4: Output Sweep with Negative Frequencies” on page 2-158 for details.

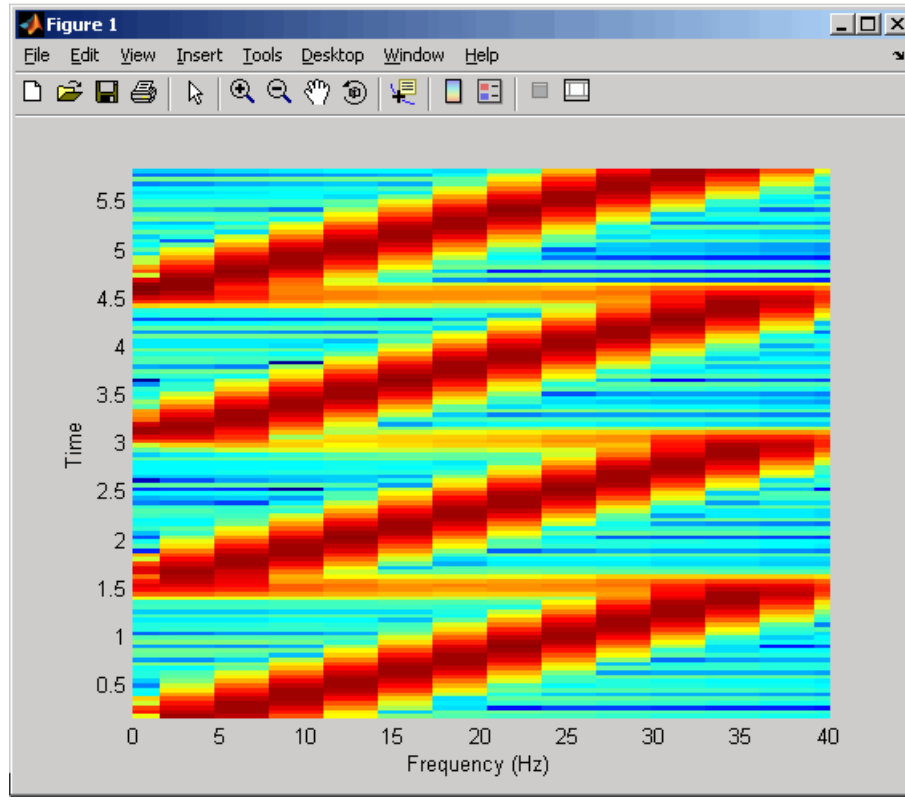
Open the Example 3 model by typing doc_chirp_ref3 at the MATLAB command line.

Run your model to see the time domain output:




```
spectrogram(dsp_examples_yout,hamming(128),...
            110,[0:.01:40],400)
```

Type the following command to view the chirp output spectrogram:



Example 4: Output Sweep with Negative Frequencies

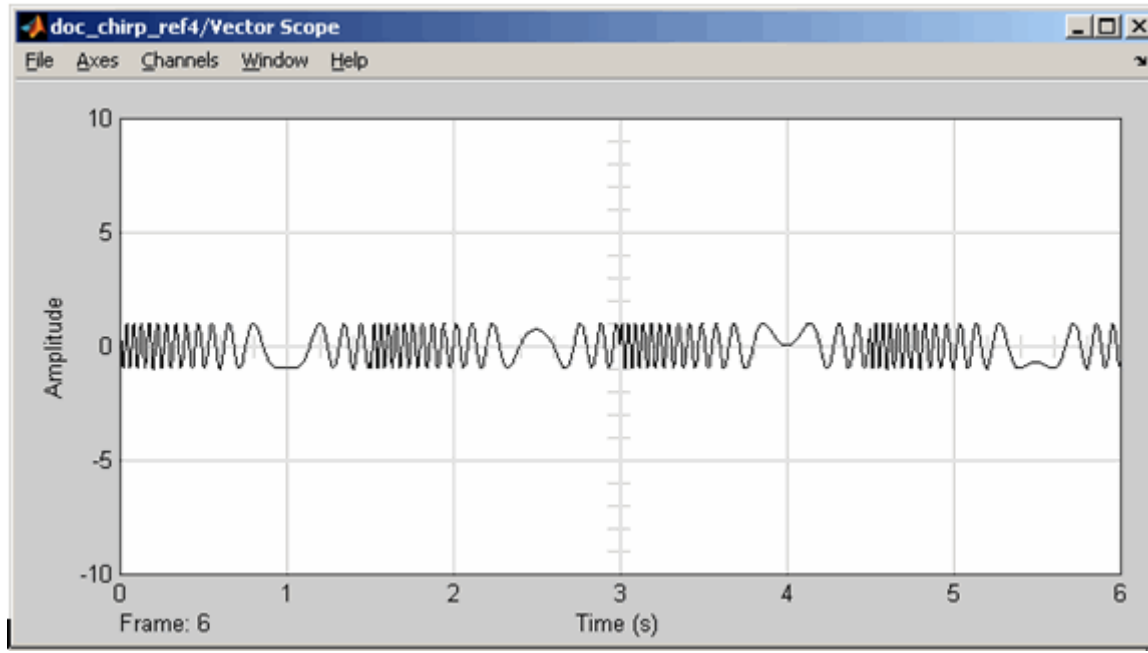
Modify the Example 1 model by changing Sweep time to 1.5, Initial frequency to 25, and Target frequency to 0. The output chirp of this example might not behave as you expect because the sweep contains negative frequencies between 1 and 1.5 seconds. The sweep reaches the Target frequency of 0 Hz at one second, then continues on its negative slope, taking on negative frequency values until it traverses one Sweep time (1.5 seconds).

The spectrogram might re ßect negative sweep frequencies along the x-axis so they appear to be positive. If you unexpectedly get a chirp

output with a spectrogram resembli ng the one below, your chirp's sweep might contain negative frequencies. See the next example for another possible unexpected chirp output.

Open the Example 4 model by typing doc_chirp_ref4 at the MATLAB command line.

Run your model to see the time domain output:



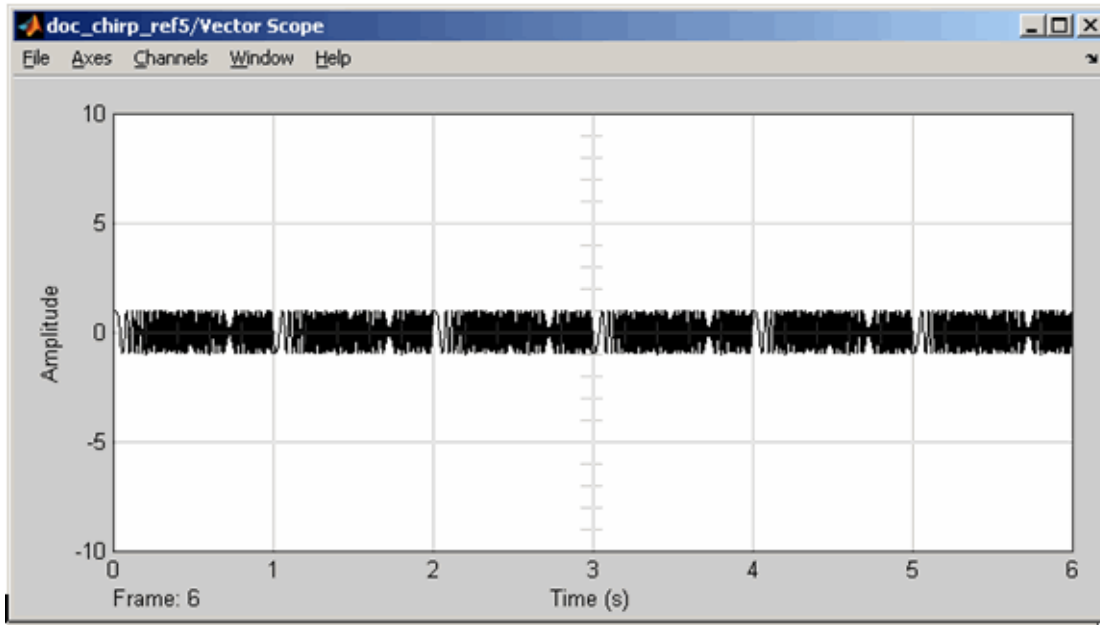
Example 5: Output Sweep with Frequencies Greater Than Half the Sampling Frequency

Modify the Example 1 model by changing the Target frequency parameter to 275. The output chirp of this model might not behave as you expectbecause the sweep contains frequencies greater than half the sampling frequency (200 Hz), which causes aliasing. If you unexpectedly get a chirp output with a spectrogram resembling the one

following, your chirp's sweep mighᵗ contain frequencies greater than half the sampling frequency. Seeᵗ the previous example for another possible unexpected chirp output.

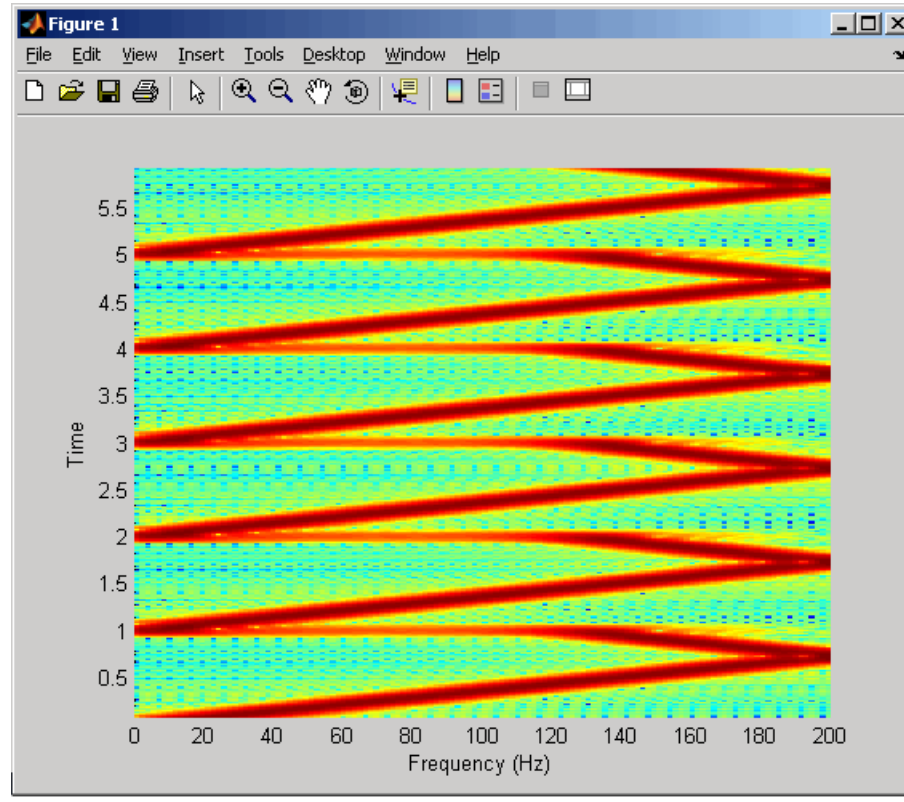
Open the Example 5 model by typing doc_chirp_ref5 at the MATLAB command line.

Run your model to see the time domain output:



Type theᵗ following command to view the chirp output spectrogram:

```
spectrogram(dsp_examples_yout,hamming(64),...
            60,256,400)
```



Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Signal From Workspace	Signal Processing Blockset
Signal Generator	Simulink
Sine Wave	Signal Processing Blockset

chirp	Signal Processing Toolbox
spectrogram	Signal Processing Toolbox

Purpose

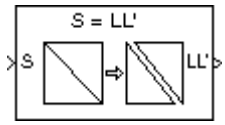
Factor square Hermitian positive definite matrix into triangular components

Library

Math Functions / Matrices and Linear Algebra / Matrix Factorizations

dspfactors

Description



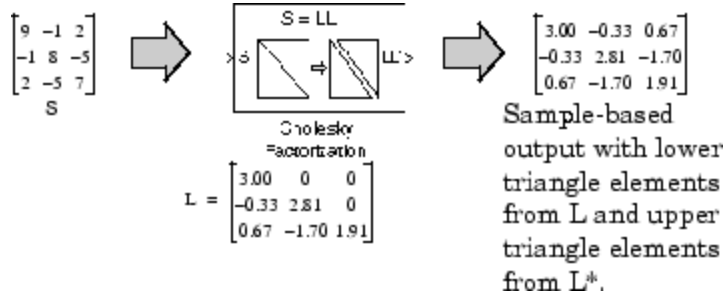
The Cholesky Factorization block uniquely factors the square Hermitian positive definite input matrix S as

$$S = LL^*$$

where L is a lower triangular square matrix with positive diagonal elements and L^* is the Hermitian (complex conjugate) transpose of L. The block outputs a matrix with lower triangle elements from L and upper triangle elements from L^*. The output is always sample based. The output is not in the same form as the output of the MATLAB chol function. In order to convert the output of the Cholesky Factorization block to the MATLAB form, use the following equation:

```
R = triu(LL');
```

Here, LL' is the output of the Cholesky Factorization block. Due to roundoff error, these equations do not produce a result that is exactly the same as the MATLAB result.



Block Output Composed of L and L^*

Input Requirements for Valid Output

The block output is valid only when its input has the following characteristics:

- Hermitian — The block does not check whether the input is Hermitian; it uses only the diagonal and upper triangle of the input to compute the output.
- Real-valued diagonal entries — The block disregards any imaginary component of the input's diagonal entries.
- Positive definite — Set the block to notify you when the input is not positive definite as described in "Response to Nonpositive Definite Input" on page 2-164.

Response to Nonpositive Definite Input

To generate a valid output, the block algorithm requires a positive definite input (see "Input Requirements for Valid Output" on page 2-164). Set the Non-positive definite input parameter to determine how the block responds to a nonpositive definite input:

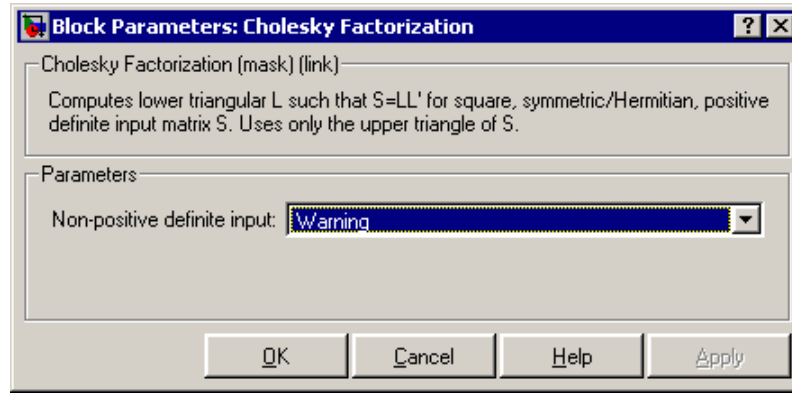
- Ignore — Proceed with the computation and do not issue an alert. The output is not a valid factorization. A partial factorization will be present in the upper left corner of the output.
- Warning — Display a warning message in the MATLAB Command Window, and continue the simulation. The output is not a valid factorization. A partial factorization will be present in the upper left corner of the output.
- Error — Display an error dialog and terminate the simulation.

Note The Non-positive definite input parameter is a diagnostic parameter. Like all diagnostic parameters on the Configuration Parameters dialog box, it is set to Ignore in the code generated for this block by Real-Time Workshop code generation software.

Performance Comparisons with Other Blocks

Note that L and L^* share the same diagonal in the output matrix. Cholesky factorization requires half the computation of Gaussian elimination (LU decomposition), and is always stable.

Dialog Box



Non-positive definite input

Response to nonpositive definite matrix inputs: Ignore , Warning , or Error . See “Response to Nonpositive Definite Input” on page 2-164.

References

Golub, G. H., and C. F. Van Loan. Matrix Computations . 3rd ed. Baltimore, MD: Johns Hopkin s University Press, 1996.

Supported Data Types

Port	Supported Data Types
S	• Double-precision floating point • Single-precision floating point
LL'	• Double-precision floating point • Single-precision floating point

See Also

Autocorrelation LPC	Signal Processing Blockset
Cholesky Inverse	Signal Processing Blockset
Cholesky Solver	Signal Processing Blockset
LDL Factorization	Signal Processing Blockset
LU Factorization	Signal Processing Blockset
QR Factorization	Signal Processing Blockset
chol	MATLAB

See “Matrix Factorizations” for related information.

Purpose

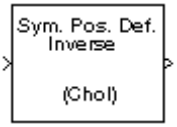
Compute inverse of Hermitian positive definite matrix using Cholesky factorization

Library

Math Functions / Matrices and Linear Algebra / Matrix Inverses

dspinverses

Description



The Cholesky Inverse block computes the inverse of the Hermitian positive definite input matrix S by performing Cholesky factorization.

$$S^{-1} = (LL^*)^{-1}$$

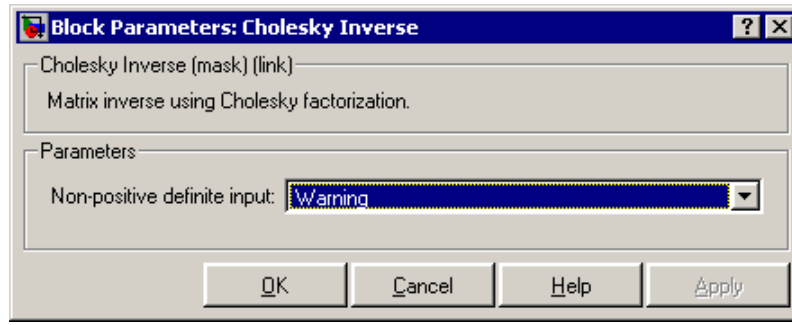
L is a lower triangular square matrix with positive diagonal elements and L^* is the Hermitian (complex conjugate) transpose of L. Only the diagonal and upper triangle of the input matrix are used, and any imaginary component of the diagonal entries is disregarded. Cholesky factorization requires half the computation of Gaussian elimination (LU decomposition), and is always stable. The output is always sample based.

The algorithm requires that the input be Hermitian positive definite. When the input is not positive definite, the block reacts with the behavior specified by the Non-positive definite input parameter. The following options are available:

- Ignore — Proceed with the computation and do not issue an alert. The output is not a valid inverse.
- Warning — Display a warning message in the MATLAB Command Window, and continue the simulation. The output is not a valid inverse.
- Error — Display an error dialog box and terminate the simulation.

Note The Non-positive de finite input parameter is a diagnostic parameter. Like all diagnostic parameters on the Con Þguration Parameters dialog box, it is set to Ignore in the code generated for this block by Real-Time Workshop code generation software.

Dialog Box



Non-positive de finite input
Response to nonpositive de Þnite matrix inputs.

References

Golub, G. H., and C. F. Van Loan. Matrix Computations . 3rd ed. Baltimore, MD: Johns Hopkin s University Press, 1996.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Cholesky Factorization	Signal Processing Blockset
Cholesky Solver	Signal Processing Blockset
LDL Inverse	Signal Processing Blockset
LU Inverse	Signal Processing Blockset

Pseudoinverse	Signal Processing Blockset
inv	MATLAB

See “Matrix Inverses” for related information.

Cholesky Solver

Purpose

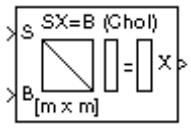
Solve SX=B for X when S is square Hermitian positive definite matrix

Library

Math Functions / Matrices and Linear Algebra / Linear System Solvers

dspsolvers

Description



The Cholesky Solver block solves the linear system SX=B by applying Cholesky factorization to input matrix at the S port, which must be square (M-by-M) and Hermitian positive definite. Only the diagonal and upper triangle of the matrix are used, and any imaginary component of the diagonal entries is disregarded. The input to the B port is the right side M-by-N matrix, B. The output is the unique solution of the equations, M-by-N matrix X, and is always sample based.

When the input is not positive definite, the block reacts with the behavior specified by the Non-positive definite input parameter. The following options are available:

- Ignore — Proceed with the computation and do not issue an alert. The output is not a valid solution.
- Warning — Proceed with the computation and display a warning message in the MATLAB Command Window. The output is not a valid solution.
- Error — Display an error dialog box and terminate the simulation.

Note The Non-positive definite input parameter is a diagnostic parameter. Like all diagnostic parameters on the Configuration Parameters dialog box, it is set to Ignore in the code generated for this block by Real-Time Workshop code generation software.

A length-M vector input for right side B is treated as an M-by-1 matrix.

Algorithm

Cholesky factorization uniquely factors the Hermitian positive definite input matrix S as

$$S = LL^*$$

where L is a lower triangular square ma trix with positive diagonal elements.

The equation SX=B then becomes

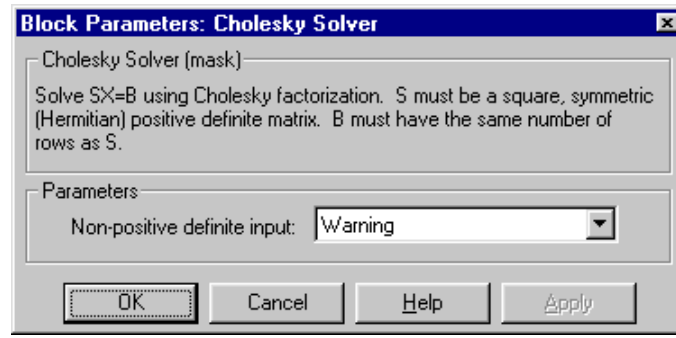
$$LL^*X = B$$

which is solved for X by making the substitution $Y = L^*X$, and solving the following two triangular systems by forward and backward substitution, respectively.

$$LY = B$$

$$L^*X = Y$$

Dialog Box



Non-positive de finite input
Response to nonpositive de Þnite matrix inputs.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Autocorrelation LPC	Signal Processing Blockset
Cholesky Factorization	Signal Processing Blockset
Cholesky Inverse	Signa l Processing Blockset
LDL Solver	Signal Processing Blockset
LU Solver	Signal Processing Blockset
QR Solver	Signal Processing Blockset

chol	MATLAB

See “Linear System Solvers” for related information.

Purpose

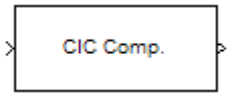
Design CIC compensator

Library

Filtering / Filter Design Toolbox

dspfdesign

Description



This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See “CIC Compensator Filter Design Dialog Box — Main Pane” in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset product.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point

Port	Supported Data Types
	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Purpose

Decimate signal using Cascaded Integrator-Comb Þlter

Library

Filtering / Multirate Filters

dspmlti4

Description

The CIC Decimation block performs a sample rate decrease (decimation) on an input signal by an integer factor. Cascaded Integrator-Comb (CIC) Þlters are a class of linear phase FIR Þlters comprised of a comb part and an integrator part.

The transfer function of a CIC decimator Þlter is

$$H(z) = {H_I}^N(z){H_c}^N(z) = \frac{\left(1 - z^{-RM}\right)^N}{\left(1 - z^{-1}\right)^N} = \left[\sum_{k=0}^{RM-1} z^{-k}\right]^N$$

where

- H_I is the transfer function of the integrator part of the Þlter.
- H_C is the transfer function of the comb part of the Þlter.
- N is the number of sections. The number of sections in a CIC Þlter is deÞned as the number of sections in either the comb part or the integrator part of the Þlter, not as the total number of sections throughout the entire Þlter.
- R is the decimation factor.
- M is the differential delay.

The CIC Decimation block supports real and complex Þxed-point inputs. Each channel of a complex input is tr eated as two real input channels.

CIC Filter Structures

The Þlter structures supported by the CIC Decimation and CIC Interpolation blocks exactly ma tch those created by Filter Design Toolbox mfilt CIC objects. If you have the Filter Design Toolbox and

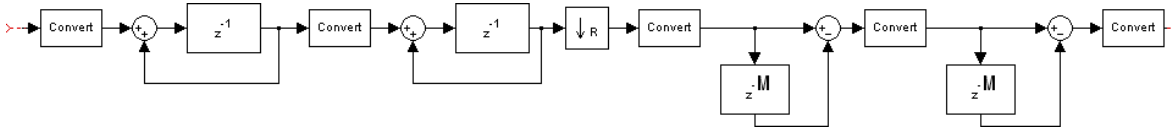
Fixed-Point Toolbox products installed, you can create an mfilt object in any workspace to specify in the Multirate filter variable parameter of this block. Otherwise, you can specify the CIC filter completely using only block dialog parameters.

This block can be used to create either of the following CIC filter structures:

- “Decimator” on page 2-176
- “Zero-latency decimator” on page 2-176

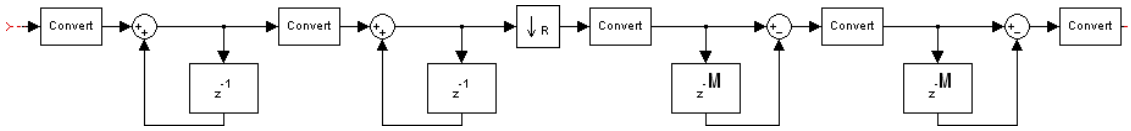
Decimator

This decimator has a latency of N, where N is the number of sections in either the comb or the integrator part of the filter.



Zero-latency decimator

This filter is the classical Hogenauer CIC decimator, which has zero latency.



Examples

The GSM Digital Down Converter demo provides an example of using the CIC Decimation block.

Dialog Box

The CIC Decimation block can operate in two different modes. Select the mode in the Coefficient source group box. If you select

- Dialog parameters, you enter information about the filter such as structure and coefficients in the block mask.

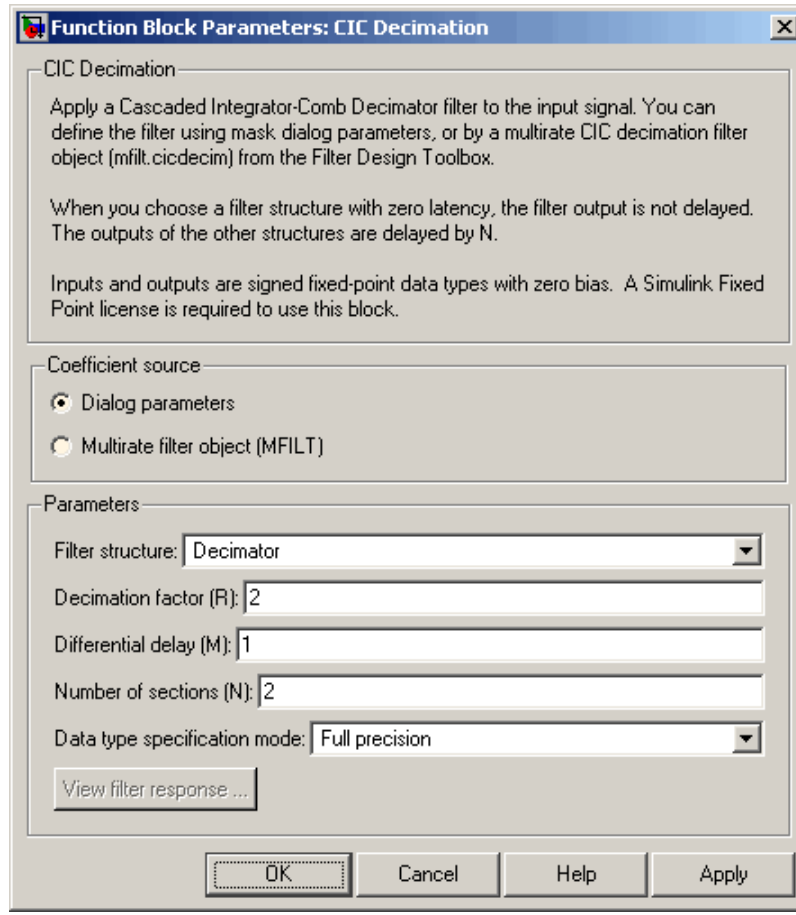
- Multirate filter object (MFILT), you specify the filter using a Filter Design Toolbox mfilt object, if you have the Filter Design Toolbox and Fixed-Point Toolbox products installed.

Different items appear on the CIC Decimation block dialog depending on whether you select Dialog parameters or Multirate filter object (MFILT) in the Coefficient source group box. See the following sections for details:

- "Specify Filter Characteristics in Dialog" on page 2-177
- "Specify Multirate Filter Object" on page 2-182

Specify Filter Characteristics in Dialog

The Main pane of the CIC Decimation block dialog appears as follows when Dialog parameters is selected in the Coefficient source group box.



Filter structure

Select one of the following CIC filter structures:

- Decimator — CIC decimator with latency N
- Zero-latency decimator — Classical Hogenauer CIC decimator with zero latency

See “CIC Filter Structures” on page 2-175 for diagrams of these filter structures.

Decimation factor (R)

Specify the decimation factor of the filter.

Differential delay (M)

Specify the differential delay of the comb part of the filter, M, as shown in the diagrams in “CIC Filter Structures” on page 2-175.

Number of sections (N)

Specify the number of filter sections. This number is equal to the number of sections in either the comb part of the filter or in the integrator part of the filter. This value is not equal to the total number of sections in the comb and integrator parts combined.

Data type specification mode

Choose how you specify the fixed-point word length and fraction length of the filter sections and/or output.

- Full precision — In this mode, the word and fraction lengths of the filter sections and outputs are automatically selected for you. All word lengths are set to

  $$\text{word length} = \text{ceil}(N * \log_2(M * R)) + I$$

  where

  — I = input word length

  — M = differential delay

  — N = number of sections

  — R = decimation factor

  All fraction lengths are set to the input fraction length.

- Minimum section word lengths — In this mode, you specify the word length of the filter output in the Output word length parameter. The word lengths of the filter sections and all fraction lengths are automatically selected for you such that each of the section word lengths is as small as possible. The

precision of each filter section is less than in Full precision mode, but the range of each section is preserved.

- Specify word lengths — In this mode you specify the word lengths of the filter sections and output in the Section word lengths and Output word length parameters. The fraction lengths of the filter sections and output are automatically selected for you such that when least significant bits are discarded at each section, the range of that section is preserved.
- Binary point scaling — In this mode you fully specify the word and fraction lengths of the filter sections and output in the Section word lengths, Section fraction lengths, Output word length, and Output fraction length parameters.

Section word lengths

Specify the word length, in bits, of the filter sections.

This parameter is only visible if Specify word lengths or Binary point scaling is selected for the Data type specification mode parameter.

Section fraction lengths

Specify the fraction length of the filter sections.

This parameter is only visible if Binary point scaling is selected for the Data type specification mode parameter.

Output word length

Specify the word length, in bits, of the filter output.

This parameter is only visible if Minimum section word lengths, Specify word lengths, or Binary point scaling is selected for the Data type specification mode parameter.

Output fraction length

Specify the fraction length of the filter output.

This parameter is only visible if Binary point scaling is selected for the Data type specification mode parameter.

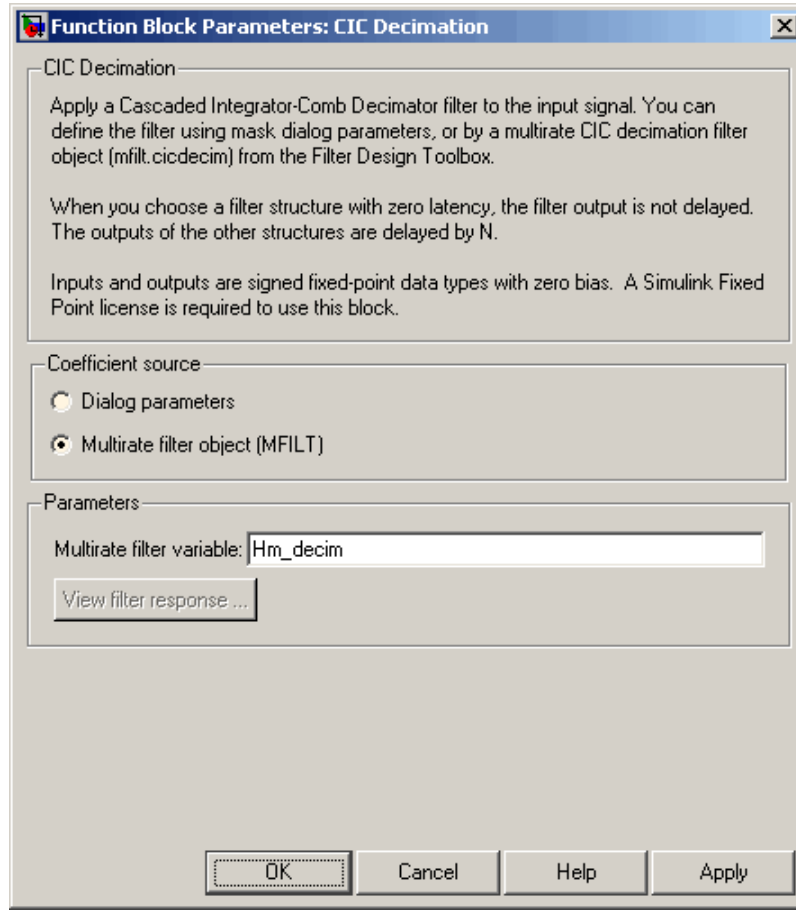
View filter response

This button opens the Filter Visualization Tool (fvtool) from the Signal Processing Toolbox product and displays the filter response of the filter defined in the block. For more information on FVTool, see the Signal Processing Toolbox documentation.

Note This button is only available when the Filter Design Toolbox and Fixed-Point Toolbox products are installed. If you specify a filter in the Multirate filter variable parameter, you must apply the filter by clicking the Apply button before using the View filter response button.

Specify Multirate Filter Object

The Main pane of the CIC Decimation block dialog appears as follows when Multirate filter object (MFILT) is specified in the Coefficient source group box.



Multirate filter variable

Specify the multirate filter object (mfilt) that you would like the block to implement. You can do this in one of three ways:

- You can fully specify the mfilt object in the block mask.
- You can enter the variable name of a mfilt object that is defined in any workspace.
- You can enter a variable name for a mfilt object that is not yet defined, as shown in the default value.

For more information on creating mfilt objects, see the mfilt function reference page in the Filter Design Toolbox documentation.

View filter response

This button opens the Filter Visualization Tool (fvtool) from the Signal Processing Toolbox product and displays the filter response of the mfilt object specified in the Multirate filter variable parameter. For more information on FVTool, see the Signal Processing Toolbox documentation.

Note This button is only available when the Filter Design Toolbox and Fixed-Point Toolbox product s are installed. If you specify a filter in the Multirate filter variable parameter, you must apply the filter by clicking the Apply button before using the View filter response button.

References

[1] Hogenauer, E.B., “An Economi cal Class of Digital Filters for Decimation and Interpolation,” IEEE Transactions on Acoustics, Speech and Signal Processing , ASSP-29(2): pp. 155-162, 1981.

[2] Meyer-Baese, U., Digital Signal Processing with Field Programmable Gate Arrays , Springer Verlag, 2001.

[3] Harris, Fredric J., Multirate Signal Processing for Communication Systems, Prentice Hall PTR, 2004.

Supported Data Types

- Fixed point (signed only)
- 8-, 16-, and 32-bit signed integers

See Also

CIC Interpolation	Signal Processing Blockset
FIR Decimation	Signal Processing Blockset
FIR Interpolation	Signal Processing Blockset
filter	Filter Design Toolbox
mfilt.cicdecim	Filter Design Toolbox
mfilt.cicinterp	Filter Design Toolbox

Purpose

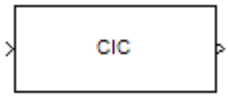
Design Cascaded Integrator-Comb (CIC) Filter

Library

Filtering / Filter Design Toolbox

dspfdesign

Description



This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See “CIC Filter Design Dialog Box — Main Pane” in the Filter Design Toolbox documentation for more information about the parameters of this block.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Fixed point

Purpose

Interpolate signal using Cascaded Integrator-Comb filter

Library

Filtering / Multirate Filters

dspmlti4

Description

The CIC Interpolation block performs a sample rate increase (interpolation) on an input signal by an integer factor. Cascaded Integrator-Comb (CIC) filters are a class of linear phase FIR filters comprised of a comb part and an integrator part.

The transfer function of a CIC interpolator filter is

$$H(z) = {H_I}^N(z){H_C}^N(z) = \frac{(1-z^{-RM})^N}{(1-z^{-1})^N} = \left[\sum_{k=0}^{RM-1} z^{-k}\right]^N$$

where

- H_I is the transfer function of the integrator part of the filter.
- H_C is the transfer function of the comb part of the filter.
- N is the number of sections. The number of sections in a CIC filter is defined as the number of sections in either the comb part or the integrator part of the filter, not as the total number of sections throughout the entire filter.
- R is the interpolation factor.
- M is the differential delay.

The CIC Interpolation block supports real and complex fixed-point inputs. Each channel of a complex input is treated as two real input channels.

CIC Filter Structures

The filter structures supported by the CIC Interpolation and CIC Decimation blocks exactly match those created by Filter Design Toolbox mfilt CIC objects. If you have the Filter Design Toolbox and

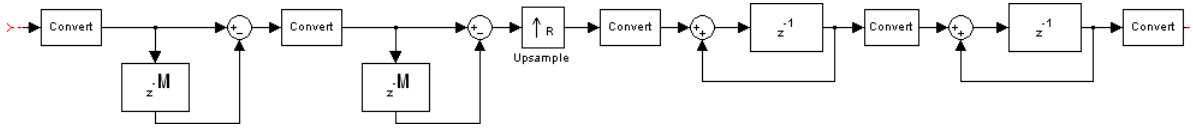
Fixed-Point Toolbox products installed, you can create an mfilt object in any workspace to specify in the Multirate filter variable parameter of this block. Otherwise, you can specify the CIC filter completely using only block dialog parameters.

This block can be used to create either of the following CIC filter structures:

- “Interpolator” on page 2-187
- “Zero-latency interpolator” on page 2-187

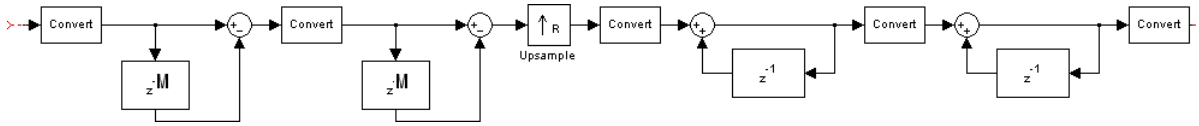
Interpolator

This interpolator has a latency of N, where N is the number of sections in either the comb or the integrator part of the filter.



Zero-latency interpolator

This filter is the classical Hogenauer CIC interpolator, which has zero latency.



Dialog Box

The CIC Interpolation block can operate in two different modes. Select the mode in the Coefficient source group box. If you select

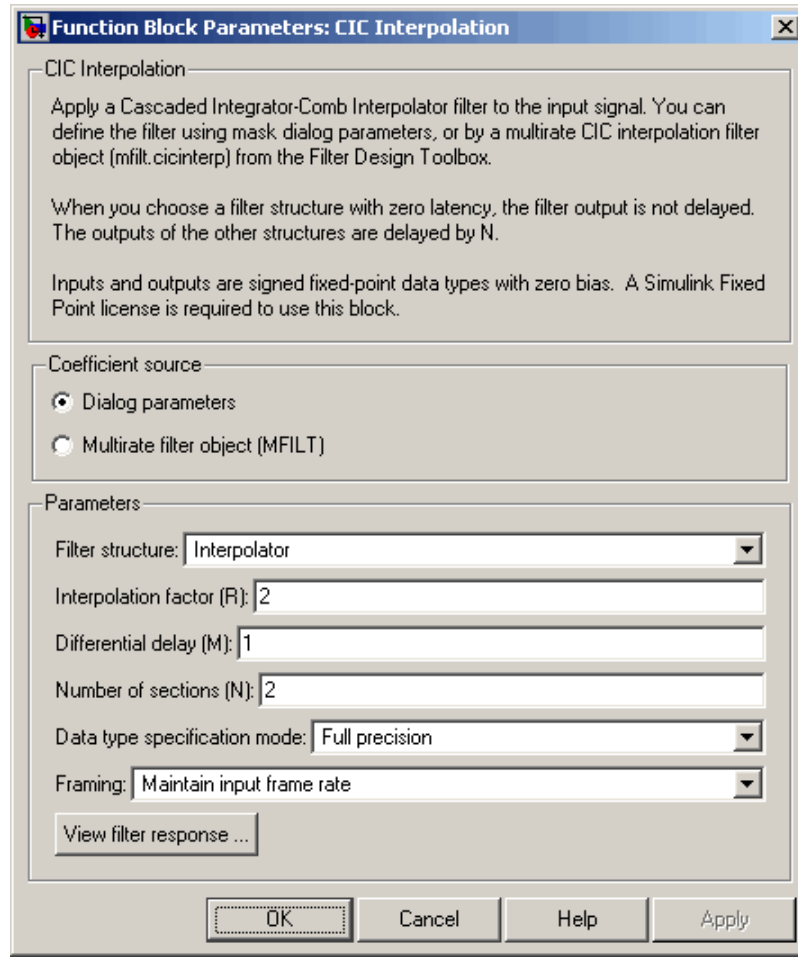
- Dialog parameters, you enter information about the filter such as structure and coefficients in the block mask
- Multirate filter object (MFILT), you specify the filter using a Filter Design Toolbox mfilt object, if you have the Filter Design Toolbox and Fixed-Point Tool box products installed.

Different items appear on the CIC Interpolation block dialog depending on whether you select Dialog parameters or Multirate filter object (MFILT) in the Coefficient source group box. See the following sections for details:

- "Specify Filter Characteristics in Dialog" on page 2-188
- "Specify Multirate Filter Object" on page 2-192

Specify Filter Characteristics in Dialog

The Main pane of the CIC Interpolation block dialog appears as follows when Dialog parameters is selected in the Coefficient source group box.



Filter structure

Select one of the following CIC filter structures:

- Interpolator — CIC interpolator with latency N
- Zero-latency interpolator — Classical Hogenauer CIC interpolator with zero latency

See “CIC Filter Structures” on page 2-186 for diagrams of these filter structures.

Interpolation factor (R)

Specify the interpolation factor of the filter.

Differential delay (M)

Specify the differential delay of the comb portion of the filter, M, as shown in the diagrams in “CIC Filter Structures” on page 2-175.

Number of sections (N)

Specify the number of filter sections. This number is equal to the number of sections in either the comb part of the filter or in the integrator part of the filter. This value is not equal to the total number of sections in the comb and integrator parts combined.

Data type specification mode

Choose how you specify the fixed-point word length and fraction length of the filter sections and/or output.

- Full precision — In this mode, the word and fraction lengths of the filter sections and outputs are automatically selected for you. The output and last section word lengths are set to

$$\text{word length} = \text{ceil}(\log_2(\frac{(R*M)^N}{R})) + I$$

  where

  — I = input word length

  — M = differential delay

  — N = number of sections

  — R = interpolation factor

  The other section word lengths are set is such a way as to accommodate the bit growth, as described in Hogenauer’s paper [1]. All fraction lengths are set to the input fraction length.

- Minimum section word lengths — In this mode, you specify the word length of the filter output in the Output word length

parameter. The word lengths of the filter sections are set in the same way as in Full precision mode.

The section fraction lengths are set to the input fraction length. The output fraction length is set to the input fraction length minus the difference between the last section and output word lengths.

- Specify word lengths — In this mode you specify the word lengths of the filter sections and output in the Section word lengths and Output word length parameters. The fraction lengths of the filter sections are set such that the spread between word length and fraction length is the same as in full-precision mode. The output fraction length is set to the input fraction length minus the difference between the last section and output word lengths.
- Binary point scaling — In this mode you fully specify the word and fraction lengths of the filter sections and output in the Section word lengths, Section fraction lengths, Output word length, and Output fraction length parameters.

Section word lengths

Specify the word length, in bits, of the filter sections.

This parameter is only visible if Specify word lengths or Binary point scaling is selected for the Data type specification mode parameter.

Section fraction lengths

Specify the fraction length of the filter sections.

This parameter is only visible if Binary point scaling is selected for the Data type specification mode parameter.

Output word length

Specify the word length, in bits, of the filter output.

This parameter is only visible if Minimum section word lengths , Specify word lengths , or Binary point scaling is selected for the Data type specification mode parameter.

Output fraction length

Specify the fraction length of the filter output.

This parameter is only visible if Binary point scaling is selected for the Data type specification mode parameter.

Framing

For frame-based operation, specify the method by which to implement the interpolation; increase the output frame rate, or increase the output frame size. This parameter cannot be set to Maintain input frame rate for sample-based signals.

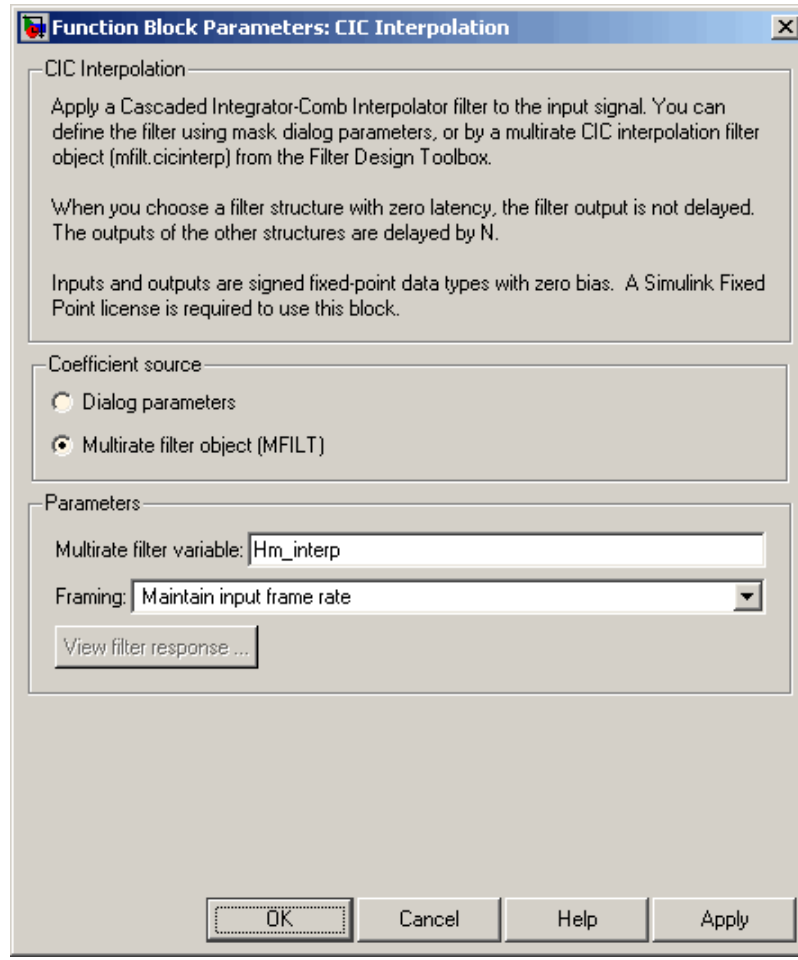
View filter response

This button opens the Filter Visualization Tool (fvtool) from the Signal Processing Toolbox product and displays the filter response of the mfilt object specified in the Multirate filter variable parameter. For more information on FVTool, see the Signal Processing Toolbox documentation.

Note This button is only available when the Filter Design Toolbox and Fixed-Point Toolbox products are installed. If you specify a filter in the Multirate filter variable parameter, you must apply the filter by clicking the Apply button before using the View filter response button.

Specify Multirate Filter Object

The Main pane of the CIC Interpolation block dialog appears as follows when Multirate filter object (MFILT) is specified in the Coefficient source group box.



Multirate filter variable

Specify the multirate filter object (mfilt) that you would like the block to implement. You can do this in one of three ways:

- You can fully specify the mfilt object in the block mask.

- You can enter the variable name of a mfilt object that is defined in any workspace.
- You can enter a variable name for a mfilt object that is not yet defined, as shown in the default value.

For more information on creating mfilt objects, see the mfilt function reference page in the Filter Design Toolbox documentation.

Framing

For frame-based operation, specify the method by which to implement the interpolation; reduce the output frame rate, or reduce the output frame size. This parameter cannot be set to Maintain input frame rate for sample-based signals.

View filter response

This button opens the Filter Visualization Tool (fvtool) from the Signal Processing Toolbox product and displays the filter response of the mfilt object specified in the Multirate filter variable parameter. For more information on FVTool, see the Signal Processing Toolbox documentation.

Note This button is only available when the Filter Design Toolbox and Fixed-Point Toolbox products are installed. If you specify a filter in the Multirate filter variable parameter, you must apply the filter by clicking the Apply button before using the View filter response button.

References

[1] Hogenauer, E.B., "An Economical Class of Digital Filters for Decimation and Interpolation," IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-29(2): pp. 155-162, 1981.

[2] Meyer-Baese, U., Digital Signal Processing with Field Programmable Gate Arrays, Springer Verlag, 2001.

[3] Harris, Fredric J., Multirate Signal Processing for Communication Systems, Prentice Hall PTR, 2004.

Supported Data Types

- Fixed point (signed only)
- 8-, 16-, and 32-bit signed integers

See Also

CIC Decimation	Signal Processing Blockset
FIR Decimation	Signal Processing Blockset
FIR Interpolation	Signal Processing Blockset
filter	Filter Design Toolbox
mfilt.cicdecim	Filter Design Toolbox
mfilt.cicinterp	Filter Design Toolbox

Complex Cepstrum

Purpose

Compute complex cepstrum of input

Library

Transforms

dspxfrm3

Description

The Complex Cepstrum block computes the complex cepstrum of each channel in the real-valued M-by-N input matrix, u. For both sample-based and frame-based inputs, the block assumes that each input column is a frame containing M consecutive samples from an independent channel. The block does not accept complex-valued inputs.

The input is altered by the application of a linear phase term so that there is no phase discontinuity at $\pm \pi$ radians. That is, each input channel is independently zero padded and circularly shifted to have zero phase at π radians.

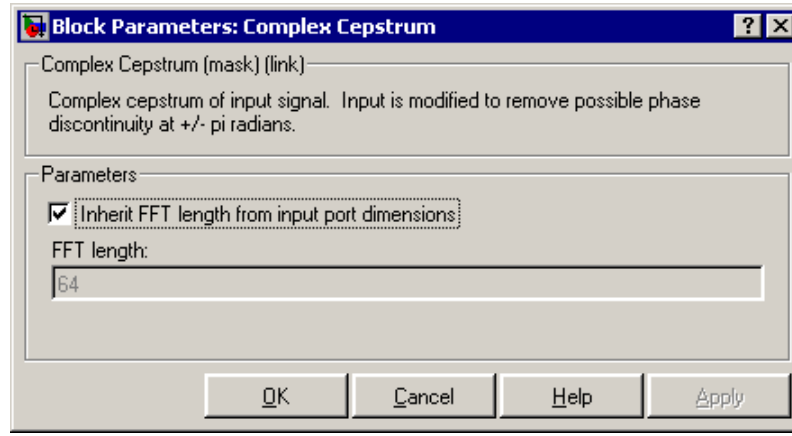
The output is a real M_o-by-N matrix, where M_o is specified by the FFT length parameter. Each output column contains the length-M_o complex cepstrum of the corresponding input column.

```
y = cceps(u,Mo)        % Equivalent MATLAB code
```

When you select the Inherit FFT length from input port dimensions check box, the output frame size matches the input frame size (M_o=M). In this case, a sample-based length-M row vector input is processed as a single channel (that is, as an M-by-1 column vector), and the output is a length-M row vector. A 1-D vector input is always processed as a single channel, and the output is a 1-D vector.

The output is always sample based, and the output port rate is the same as the input port rate.

Dialog Box



Inherit FFT length from input port dimensions

When selected, matches the outpu t frame size to the input frame size.

FFT length

The number of frequency points at which to compute the FFT, which is also the output frame size, M_o. This parameter is available when you do not select Inherit FFT length from input port dimensions .

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

DCT	Signal Processing Blockset
FFT	Signal Processing Blockset
Real Cepstrum	Signal Processing Blockset
cceps	Signal Processing Toolbox

Complex Exponential

Purpose

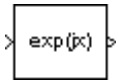
Compute complex exponential function

Library

Math Functions / Math Operations

dspmathops

Description

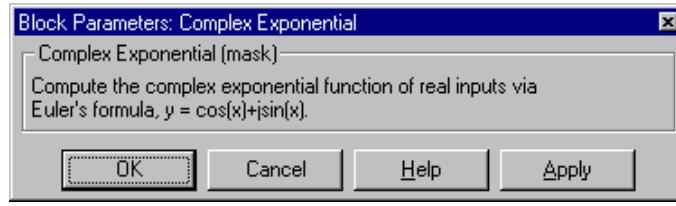


The Complex Exponential block computes the complex exponential function for each element of the real input, u.

$$y = e^{ju} = \cos u + j \sin u$$

where $j = \sqrt{-1}$. The output is complex, with the same size and frame status as the input.

Dialog Box



Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Math Function	Simulink
Sine Wave	Signal Processing Blockset
exp	MATLAB

Purpose

Generate constant value

Library

Signal Processing Sources

dspsrcs4

Description

The Constant block is an implemen tation of the Simulink Constant block. See Constant for more information.

Constant Diagonal Matrix

Purpose

Generate square, diagonal matrix

Library

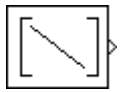
- Signal Processing Sources

  dspsrcs4

- Math Functions / Matrices and Lin ear Algebra / Matrix Operations

  dspmtrx3

Description

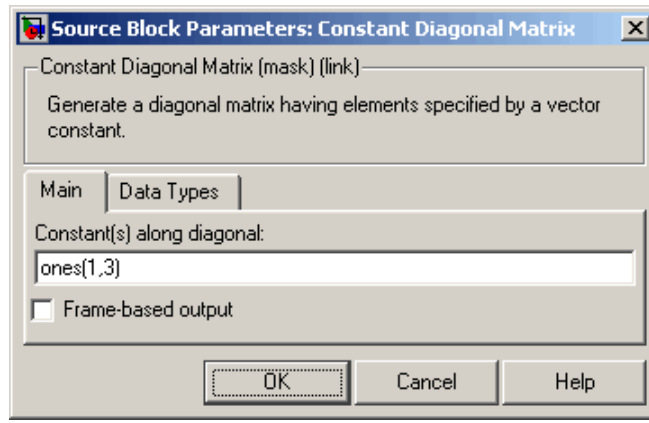


The Constant Diagonal Matrix block outputs a square diagonal matrix constant. The Constant along diagonal parameter determines the values along the matrix diagonal. This parameter can be a scalar to be repeated for all elements along the diagonal, or a vector containing the values of the diagonal elements. To generate the identity matrix, set the Constant along diagonal to 1, or use the Identity Matrix block.

The output is frame based when you select the Frame-based output check box; otherwise, the output is sample based.

Dialog Box

The Main pane of the Constant Diagonal Matrix block dialog appears as follows.



Opening this dialog box causes a running simulation to pause. See “Changing Source Block Parameters During Simulation” in the online Simulink documentation for details.

Constant(s) along diagonal

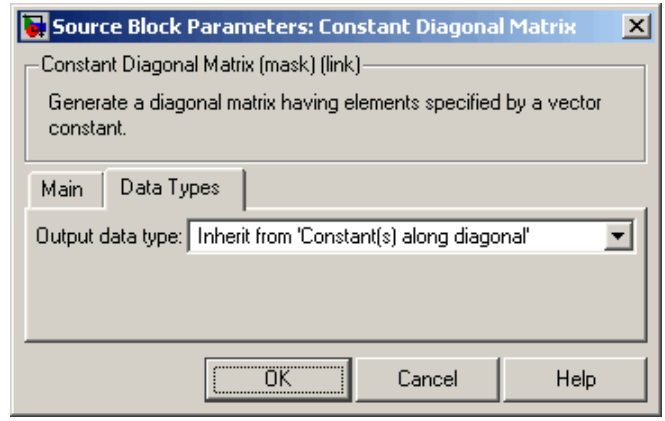
Specify the values of the elements along the diagonal. You can input a scalar or a vector. Tunable.

When you specify any data type information in this field, it is overridden by the value of the Output data type parameter on the Data Types pane, unless you select Inherit from 'Constant(s) along diagonal'.

Frame-based output

Select to cause the output of the block to be frame based. Otherwise, the output is sample based.

The Data types pane of the Constant Diagonal Matrix block dialog appears as follows.



Output data type

Specify the output data type in one of the following ways:

- Choose one of the built-in data types from the list.

- Choose Fixed-point to specify the output data type and scaling in the Signed, Word length, Set fraction length in output to, and Fraction length parameters.
- Choose User-defined to specify the output data type and scaling in the User-defined data type, Set fraction length in output to, and Fraction length parameters.
- Choose Inherit from 'Constant(s) along diagonal' to set the output data type and scaling to match the values of the Constant(s) along diagonal parameter on the Main pane.
- Choose Inherit via back propagation to set the output data type and scaling to match the next block downstream.

The value of this parameter overrides any data type information specified in the Constant(s) along diagonal parameter on the Main pane, except when you select Inherit from 'Constant(s) along diagonal'.

Signed

Select to output a signed fixed-point signal. Otherwise, the signal is unsigned. This parameter is only visible when you select Fixed-point for the Output data type parameter.

Word length

Specify the word length, in bits, of the fixed-point output data type. This parameter is only visible when you select Fixed-point for the Output data type parameter.

User-defined data type

Specify any built-in or fixed-point data type. You can specify fixed-point data types using the following Simulink® Fixed Point™ functions: sfix, ufix, sint, uint, sfrac, and ufrac. This parameter is only visible when you select User-defined for the Output data type parameter.

Set fraction length in output

Specify the scaling of the fixed-point output by either of the following two methods:

- Choose Best precision to have the output scaling automatically set such that the output signal has the best possible precision.
- Choose User-defined to specify the output scaling in the Fraction length parameter.

This parameter is only visible when you select Fixed-point for the Output data type parameter, or when you select User-defined and the specified output data type is a fixed-point data type.

Fraction length

For fixed-point output data types, specify the number of fractional bits, or bits to the right of the binary point. This parameter is only visible when you select Fixed-point or User-defined for the Output data type parameter and User-defined for the Set fraction length in output to parameter.

Supported Data Types

- Double-precision floating point
- Single-precision floating point
- Fixed point
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

See Also

Create Diagonal Matrix	Signal Processing Blockset
Constant	Simulink
Identity Matrix	Signal Processing Blockset
diag	MATLAB

Constant Ramp

Purpose

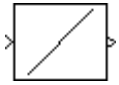
Generate ramp signal with length based on input dimensions

Library

Signal Operations

dspsigops

Description



The Constant Ramp block generates the constant ramp signal

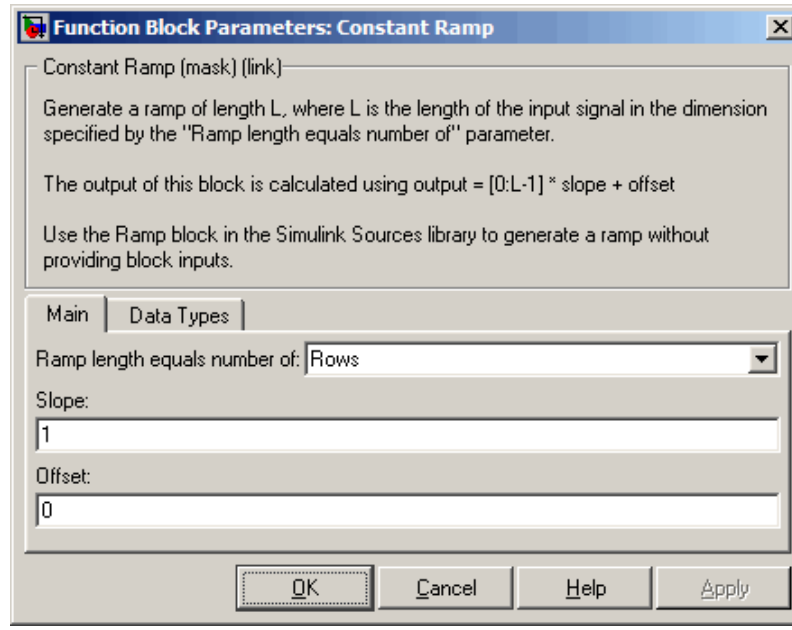
```
y = (0:L-1)*m + b
```

where m is the slope specified by the scalar Slope parameter, and b is the y-intercept specified by the scalar Offset parameter.

For a 1-D vector input, L is equal to the length of the input vector. For an N-D array input, the length L of the output ramp is equal to the length of the input in the dimension specified by the Ramp length equals number of or Dimension parameter. The output, y, is always a 1-D vector.

Dialog Box

The Main pane of the Constant Ramp block dialog appears as follows.



Ramp length equals number of

Specify whether the length of the output ramp is the number of rows, number of columns, or the length of the specified dimension of the input.

Dimension

Specify the one-based dimension of the input array that determines the length of the output ramp.

This parameter is only visible when you select Elements in specified dimension for the Ramp length equals number of parameter.

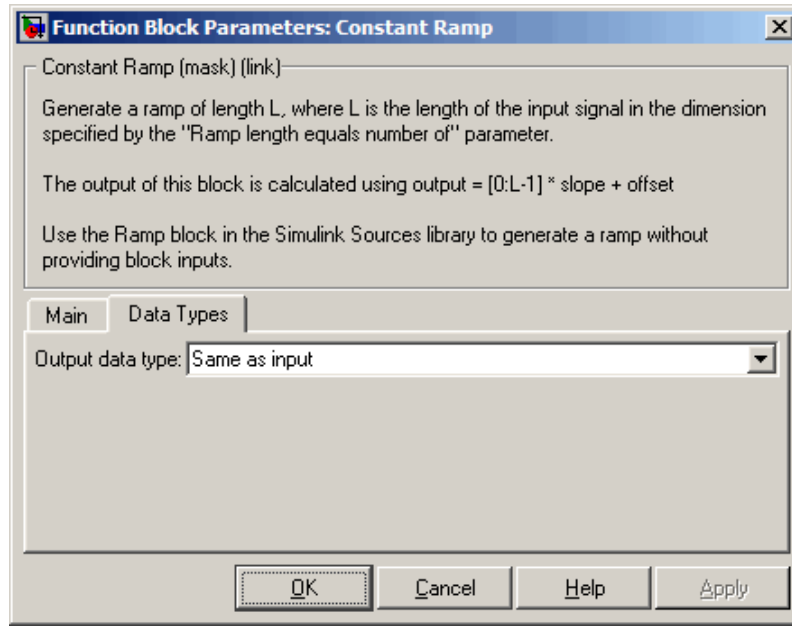
Slope

Specify the scalar slope of the ramp.

Offset

Specify the scalar y-intercept of the ramp.

The Data types pane of the Constant Ramp block dialog appears as follows.



Output data type

Specify the output data type in one of the following ways:

- Select Same as input to force the data type of the output to be the same as the data type of the input to the block.
- Select one of the built-in data types from the list.
- Select Fixed-point to specify the output data type and scaling in the Signed, Word length, Set fraction length in output to, and Fraction length parameters.

- Select User-defined to specify the output data type and scaling in the User-defined data type, Set fraction length in output to, and Fraction length parameters.
- Select Inherit via back propagation to set the output data type and scaling to match the next block downstream.

This block differs from other Signal Processing Blockset blocks in that unless you choose Same as input for this parameter, the data types of the input and the output do not need to be the same.

Signed

Select to output a signed fixed-point signal. Otherwise, the signal is unsigned. This parameter is only visible when you select Fixed-point for the Output data type parameter.

Word length

Specify the word length, in bits, of the fixed-point output data type. This parameter is only visible when you select Fixed-point for the Output data type parameter.

User-defined data type

Specify any built-in or fixed-point data type. You can specify fixed-point data types using the following Simulink Fixed Point functions: sfix, ufix, sint, uint, sfrac, and ufrac. This parameter is only visible when you select User-defined for the Output data type parameter.

Set fraction length in output to

Specify the scaling of the fixed-point output by either of the following two methods:

- Select Best precision to have the output scaling automatically set such that the output signal has the best possible precision.
- Select User-defined to specify the output scaling in the Fraction length parameter.

This parameter is only visible when you select Fixed-point for the Output data type parameter, or when you select

User-defined and the specified output data type is a fixed-point data type.

Fraction length

For fixed-point output data types, specify the number of fractional bits, or bits to the right of the binary point. This parameter is only visible when you select Fixed-point or User-defined for the Output data type parameter and User-defined for the Set fraction length in output to parameter.

Supported Data Types

- Double-precision floating point
- Single-precision floating point
- Fixed point
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

This block differs from other Signal Processing Blockset blocks in that unless you choose Same as input for the Output data type parameter, the data types of the input and the output do not need to be the same.

See Also

Create Diagonal Matrix	Signal Processing Blockset
Constant	Simulink
Identity Matrix	Signal Processing Blockset

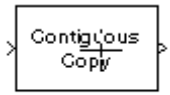
Purpose

Create discontiguous input in contiguous block of memory

Library

dspobslib

Description



Contiguous Copy

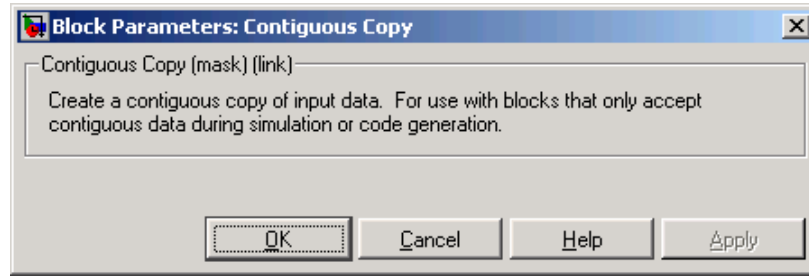
Note The Contiguous Copy block is still supported but is likely to be obsoleted in a future release.

The Contiguous Copy block copies the input to a contiguous block of memory, and passes this new copy to the output. The output is identical to the input, but is guaranteed to reside in a contiguous section of memory.

Because Simulink software employs an efficient copy-by-reference method for propagating data in a model, some operations produce outputs with discontiguous memory locations.

Although this does not present a problem during simulation, blocks linked to versions of DSP Blockset prior to 4.0 may require contiguous inputs for code generation with the Real-Time Workshop product. When such blocks are used in a model intended for code generation, they should be preceded by the Contiguous Copy block to ensure that their inputs are contiguous.

Dialog Box



Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Fixed point
- Boolean
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

Purpose

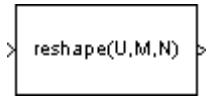
Reshape 1-D or 2-D input to 2-D matrix with speci Þed dimensions

Library

Signal Management / Signal Attributes

dspsigattribs

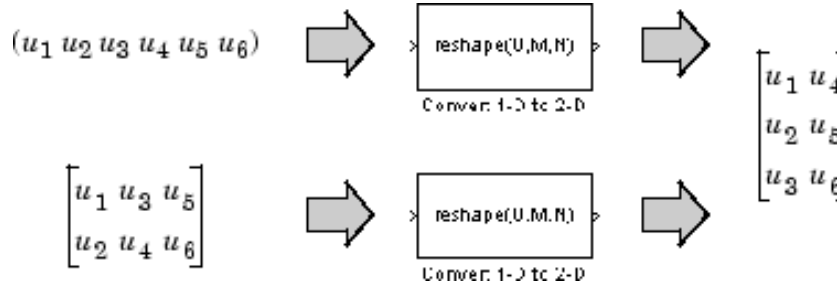
Description



The Convert 1-D to 2-D block reshapes a length- M_i 1-D vector or an M_i-by-N_i matrix to an M_o-by-N_o matrix, where M_o is speciÞed by the Number of output rows parameter, and N_o is speciÞed by the Number of output columns parameter.

```
y = reshape(u,Mo,No)          % Equivalent MATLAB code
```

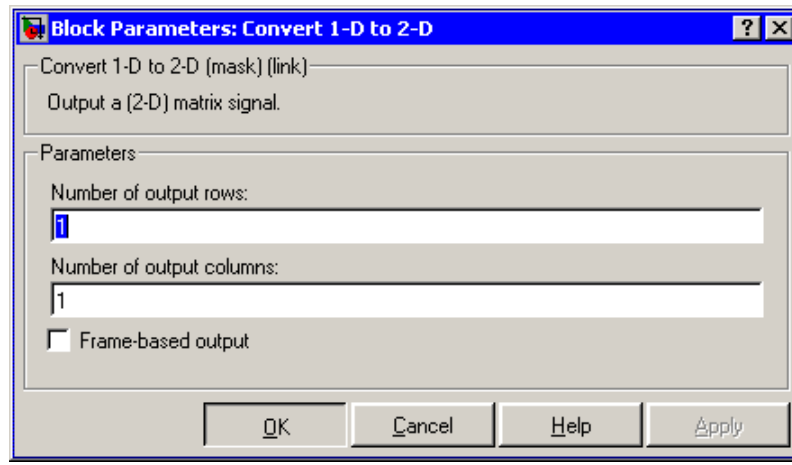
The input is reshaped columnwise, as shown in the two cases below. The length-6 vector and the 2-by-3 matrix are both reshaped to the same 3-by-2 output matrix.



An error is generated when (M_o*N_o)≠(M_i*N_i). That is, the total number of input elements must be conserved in the output.

The output is frame based when you select the Frame-based output check box; otherwise, the output is sample based.

Dialog Box



Number of output rows

The number of rows, M_o, in the output matrix.

Number of output columns

The number of rows, N_o, in the output matrix.

Frame-based output

Creates a frame-based output when selected.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers

Port	Supported Data Types
	• 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Buffer	Signal Processing Blockset
Convert 2-D to 1-D	Signal Processing Blockset
Frame Status Conversion (Obsolete)	Signal Processing Blockset
Reshape	Simulink
Submatrix	Signal Processing Blockset

Convert 2-D to 1-D

Purpose

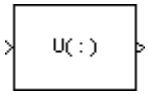
Convert 2-D matrix input to 1-D vector

Library

Signal Management / Signal Attributes

dspsigattribs

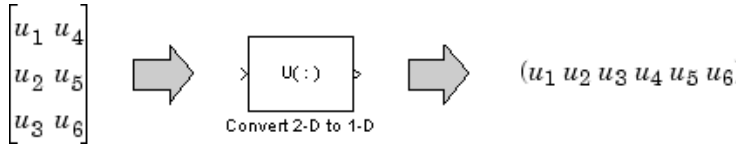
Description



The Convert 2-D to 1-D block reshapes an M-by-N matrix input to a 1-D vector with length M*N.

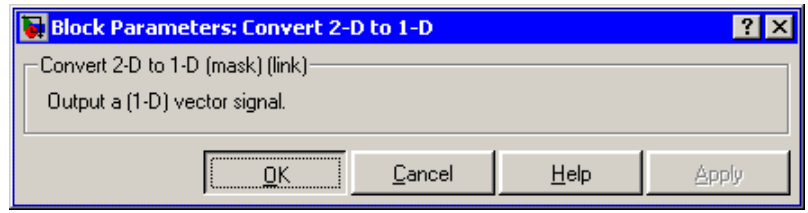
```
y = u(:)        % Equivalent MATLAB code
```

The input is reshaped columnwise, as shown below for a 3-by-2 matrix.



The output is always sample-based.

Dialog Box



Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Buffer	Signal Processing Blockset
Convert 1-D to 2-D	Signal Processing Blockset
Frame Status Conversion (Obsolete)	Signal Processing Blockset
Reshape	Simulink
Submatrix	Signal Pro cessing Blockset

Convolution

Purpose

Compute convolution of two inputs

Library

Signal Operations

dspsigops

Description

The Convolution block convolves the first dimension of a sample-based N-D input array u, with the first dimension of a sample-based N-D input array v. The block can also independently convolve a sample-based vector with the first-dimension of an N-D input array. For frame-based inputs, the Convolution block convolves analogous columns of an M_u-by-N input matrix u and an M_v-by-N input matrix v. The Convolution block can also independently convolve a single-channel frame-based column vector with each column of a multiple-channel frame-based matrix.

The frame status of both inputs to the Convolution block must be the same. The output of the block is always sample-based.

The Convolution block accepts real and complex floating-point and fixed-point inputs except for complex unsigned fixed-point inputs. Fixed-point signals are not supported for the frequency domain.

Convolution with Signal Processing Blockset Blocks

The general equation for convolution is

$$y(k) = \sum_n u(n-k) * h(k)$$

There are two Signal Processing Blockset blocks that can be used for this purpose:

- Convolution
- Digital Filter

The Convolution block assumes that all of u and h are available at each Simulink time step, and computes the entire convolution at every one.

The Digital Filter block can be used for convolving signals in situations where all of h is available at each time step, but u is a sequence that comes in over the life of the simulation. When you use the Digital Filter block, the convolution is computed only once. To convolve inputs with the Digital Filter block, you must set the Transfer function type to FIR (all zeros) .

Use the following questions to he lp you determine which block best Þts your needs:

Question	Answer	Recommended Block(s)
How many convolutions do you intend to perform?	Many convolutions, one at each time step	• Convolution block
	One convolution over the life of the simulation	• Convolution block • Digital Filter block (in FIR mode)
How long are your input sequences?	Both sequences have a Þnite length	• Convolution block • Digital Filter block (in FIR mode)
	One sequence has an inÞnite (not predetermined) length	• Digital Filter block (in FIR mode)
How many of the inputs are scalar sample-based streams?	None	• Convolution block • Digital Filter block (in FIR mode)
	One or both	• Buffer block followed by the Convolution block • Digital Filter block (in FIR mode)

Convolving Frame-Based Inputs

When the inputs to the Convolution block are a frame based M_u-by-N input matrix u and an M_v-by-N input matrix v, the output, y, is a sample-based (M_u+M_v–1)-by-N matrix whose jth column has elements

$$y_{i,j} = \sum_{k=0}^{\max(M_u,M_v)-1} u_{k,j} v_{(i-k),j} \quad 0 \le i \le (M_u + M_v - 2)$$

Inputs u and v are zero when indexed outside of their valid ranges. When both inputs are real, the output is real; when one or both inputs are complex, the output is complex.

When one input is a column vector (single channel) and the other is a matrix (multiple channels), the single-channel input is independently convolved with each channel of the multichannel input. For example, when u is a M_u-by-1 column vector and v is an M_v-by-N matrix, the output is an (M_u+M_v–1)-by-N matrix whose jth column has elements

$$y_{i,j} = \sum_{k=0}^{\max(M_u,M_v)-1} u_{k} v_{(i-k),j} \quad 0 \le i \le (M_u + M_v - 2)$$

Convolving Sample-Based Inputs

The Convolution block supports sample-based N-D input arrays. The convolution of N-D array input is always computed across the first dimension. If both inputs are N-D arrays, the size of their first dimension can differ, but the size of all other dimensions must be equal. For example, when u is an M_u-by-N-by-P array, and v is an M_v-by-N-by-P array, the output is an (M_u+M_v–1)-by-N-by-P array.

When one input is an N-D sample-based array and the other is a vector, the vector is independently convolved with the first dimension of the N-D input. For example, when u is a M_u-by-1 column vector and v is an M_v-by-N-by-P array, the output is an (M_u+M_v–1)-by-N-by-P array.

The Convolution block also accepts two vector inputs. When u and v are sample-based vectors with lengths M_u and M_v, the Convolution block performs the vector convolution

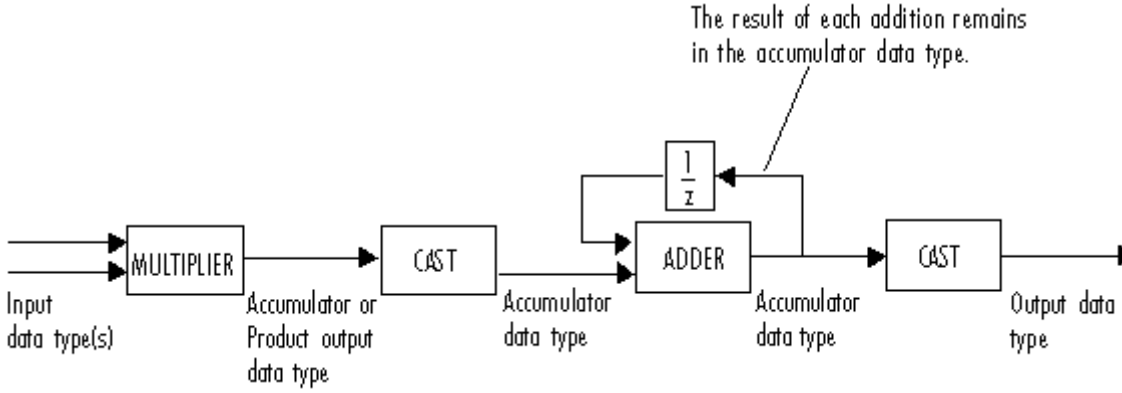
$$y_i = \sum_{k=0}^{\max(M_u,M_v)-1} u_k v_{(i-k)} \quad 0 \le i \le (M_u + M_v - 2)$$

The dimensions of the sample-based output vector are determined by the dimensions of the input vectors:

- When both inputs are row vectors, or when one input is a row vector and the other is a 1-D vector, the output is a 1-by-(M_u+M_v–1) row vector.
- When both inputs are column vectors, or when one input is a column vector and the other is a 1-D vector, the output is a (M_u+M_v–1)-by-1 column vector.
- When both inputs are 1-D vectors, the output is a 1-D vector of length M_u+M_v–1.

Fixed-Point Data Types

The following diagram shows the data types used within the Convolution block for fixed-point signals (time domain only).

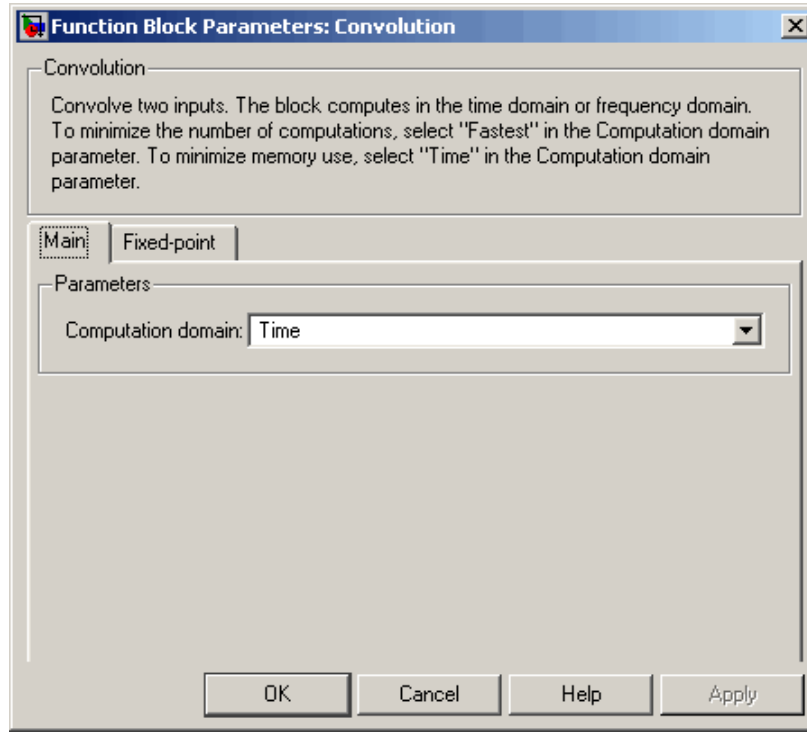


You can set the product output, accumulator, and output data types in the block dialog as discussed in the next section.

The output of the multiplier is in the product output data type when the input is real. When the input is complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see “Multiplication Data Types”.

Dialog Box

The Main pane of the Convolution block dialog appears as follows.

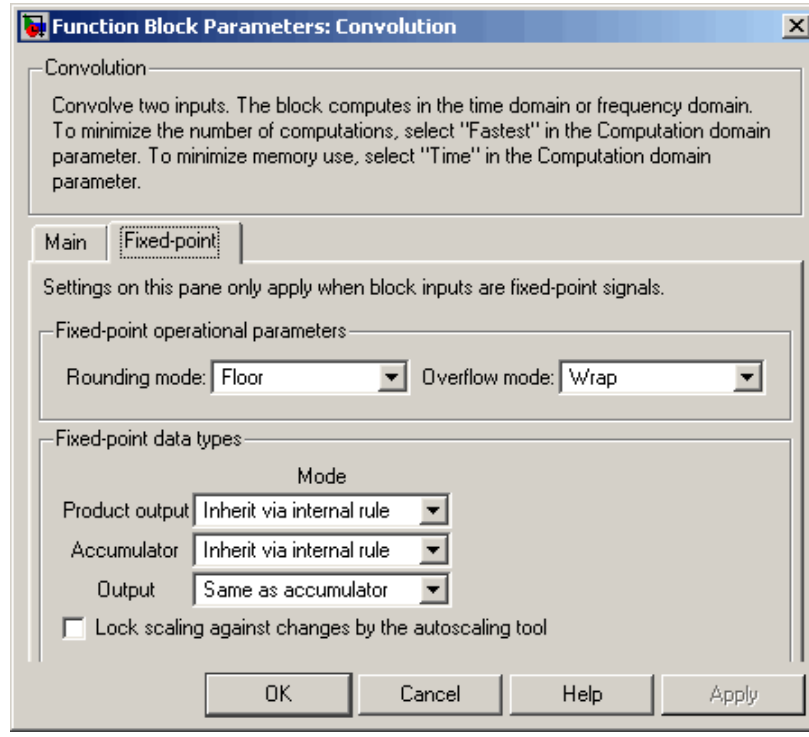


Computation domain

Set the domain in which the block computes convolutions:

- Time — The block computes in the time domain, which minimizes memory use.
- Frequency — The block computes in the frequency domain, which might require fewer computations than computing in the time domain, depending on the input length.
- Fastest — The block computes in the domain, which minimizes the number of computations.

The Fixed-point pane of the Convolution block dialog appears as follows.



Note Fixed-point signals are only supported for the time domain. To use the parameters on this pane, make sure Time is selected for the Computation domain parameter on the Main pane.

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

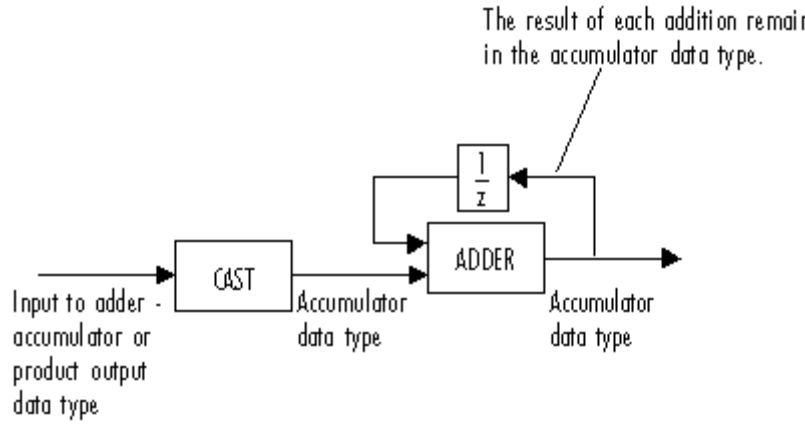
Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how you want to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-44 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Inherit via internal rule, the product output word length and fraction length are calculated automatically. For information about how the product output word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as first input, these characteristics match those of the first input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator



As depicted in this figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how you want to designate this accumulator word and fraction lengths.

You also use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block. See “Multiplication Data Types” for more information.

- When you select Inherit via internal rule, the accumulator word length and fraction length are calculated automatically. For information on about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as product output, these characteristics match those of the product output.
- When you select Same as first input, these characteristics match those of the first input to the block.

- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the data type and scaling of the output of the block:

- When you select Same as accumulator , these characteristics match those of the accumulator.

  A special case occurs when Inherit via internal rule is specified for Accumulator , and both block inputs are complex. In that case, the output word length is one less than the accumulator word length.
- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as first input , these characteristics match those of the first input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Correlation — Signal Processing Blockset

conv — MATLAB

Purpose

Compute cross-correlation of two inputs

Library

Statistics

dspstat3

Description

The Correlation block computes the cross-correlation of the first dimension of a sample-based N-D input array u, and the first dimension of a sample-based N-D input array v. The block can also independently cross-correlate a sample-based vector with the first-dimension of an N-D input array. For frame-based inputs, the Correlation block computes the cross-correlation of analogous columns of an M_u-by-N input matrix u and an M_v-by-N input matrix v. The Correlation block can also independently cross-correlate a single-channel frame-based column vector with each column of a multiple-channel frame-based matrix.

The frame status of both inputs to the Correlation block must be the same. The output of the block is always sample-based.

The Correlation block accepts real and complex floating-point and fixed-point inputs except for complex unsigned fixed-point inputs. Fixed-point signals are not supported for the frequency domain.

Correlating Frame-Based Inputs

When the inputs to the Correlation block are an M_u-by-N frame-based input matrix u and an M_v-by-N frame-based input matrix v, the output, y, is a sample-based (M_u+M_v-1)-by-N matrix whose jth column has elements

$$y_{uv(i,j)} = \sum_{k=0}^{\max(M_u,M_v)-1} u_{k,j}\, v^{*}_{(k-i),j} \qquad 0 \le i < M_v$$

$$y_{uv(i,j)} = y^{*}_{vu(-i,j)} \qquad -M_u < i < 0$$

where * denotes the complex conjugate. Inputs u and v are zero when indexed outside of their valid ranges. When both inputs are real,

the output is real; when one or both inputs are complex, the output is complex.

When one input is a column vector (single channel) and the other is a matrix (multiple channels), the single-channel input is independently cross-correlated with each channel of the multichannel input. Each column of the input represents a separate channel. For example, when u is a M_u-by-1 column vector and v is an M_v-by-N matrix, the output is an (M_u+M_v–1)-by-N matrix whose jth column has elements

$$y_{uv(i,j)} = \sum_{k=0}^{\max(M_u,M_v)-1} u_k v^*_{(k-i),j} \qquad 0 \le i < M_v$$

$$y_{uv(i,j)} = y^*_{vu(-i,j)} \qquad -M_u < i < 0$$

Correlating Sample-Based Inputs

The Correlation block supports sample-based N-D array input. The cross-correlation for sample-based N-D inputs is always computed across the first dimension. If both inputs are N-D arrays, the size of their first dimensions can differ, but the size of all other dimensions must be equal. For example, when u is an M_u-by-N-by-P array and v is an M_v-by-N-by-P array, the output, y, is a sample-based (M_u+M_v–1)-by-N-by-P array.

When one input is an N-D sample-based array and the other is a vector, the vector is independently cross-correlated with each column of the N-D input. For example, when u is a M_u-by-1 column vector and v is an M_v-by-N-by-P array, the output is an (M_u+M_v-1)-by-N-by-P array.

The Correlation block also accepts two vector inputs. When u and v are sample-based column vectors with lengths M_u and M_v, the Correlation block performs the vector cross-correlation according to the following equation:

$$y_{uv(i)} = \sum_{k=0}^{\max(M_u,M_v)-1} u_k v^*_{(k-i)} \qquad 0 \le i < M_v$$

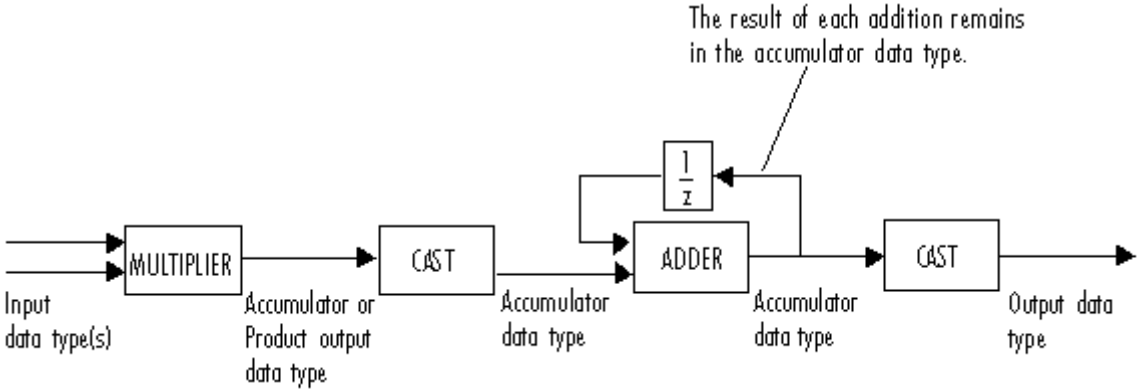
$$y_{uv(i)} = y^*_{vu(-i)} \qquad -M_u < i < 0$$

The dimensions of the sample-based output vector are determined by the dimensions of the input vectors:

- When both inputs are column vectors, or when one input is a column vector and the other is a 1-D vector, the output is a (M_u+M_v-1)-by-1 column vector.
- When both inputs are row vectors, or when one input is a row vector and the other is a 1-D vector, the output is a 1-by-(M_u+M_v-1) row vector.
- When both inputs are 1-D vectors, the output is a 1-D vector of length M_u+M_v-1.

Fixed-Point Data Types

The following diagram shows the data types used within the Correlation block for fixed-point signals (time domain only).

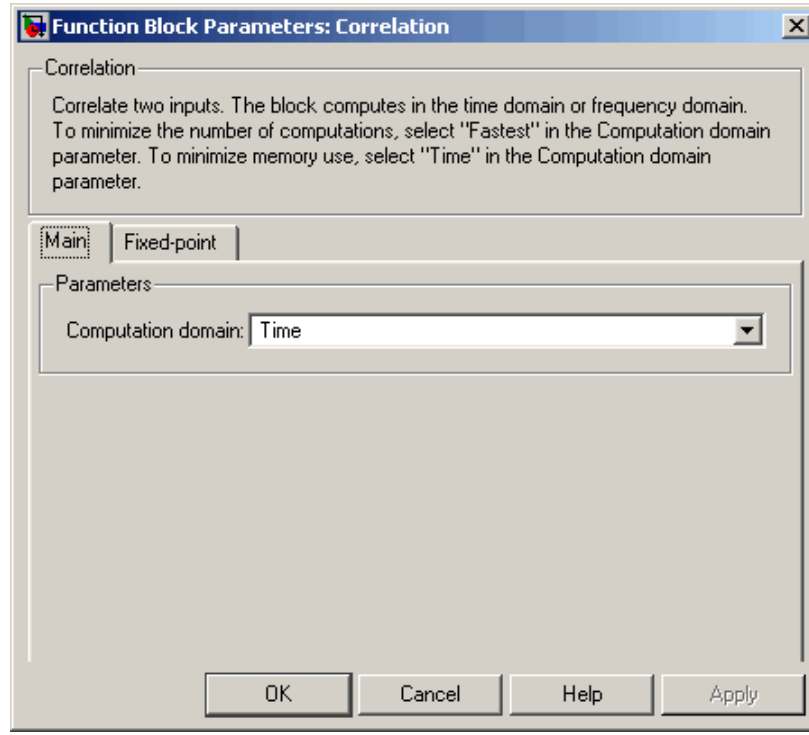


You can set the product output, accumulator, and output data types in the block dialog as discussed in the next section.

The output of the multiplier is in the product output data type when the input is real. When the input is complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see “Multiplication Data Types”.

Dialog Box

The Main pane of the Correlation block dialog appears as follows.

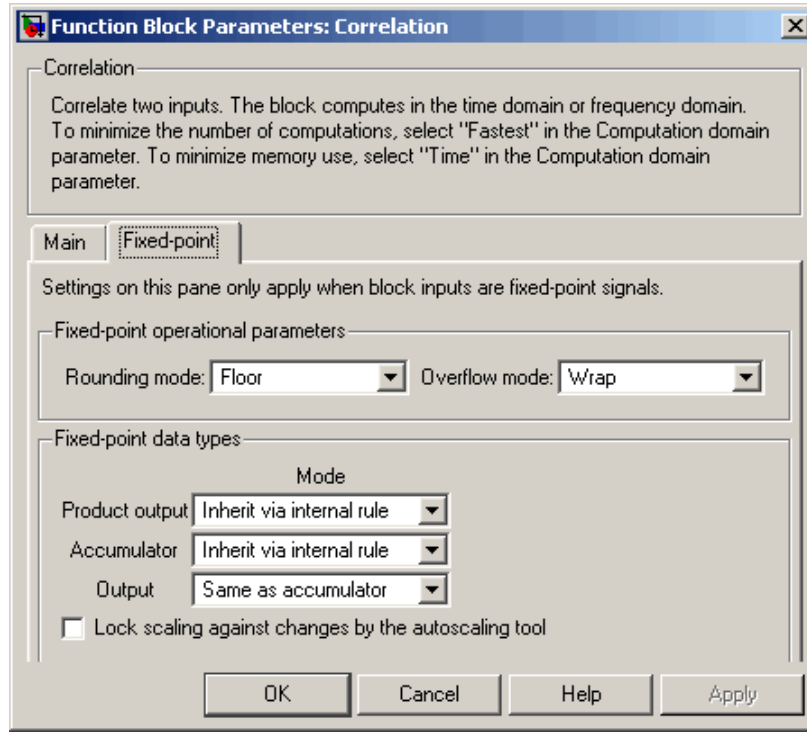


Computation domain

Set the domain in which the block computes correlations:

- Time — The block computes in the time domain, which minimizes memory use.
- Frequency — The block computes in the frequency domain, which might require fewer computations than computing in the time domain, depending on the input length.
- Fastest — The block computes in the domain, which minimizes the number of computations.

The Fixed-point pane of the Correlation block dialog appears as follows.



Note Fixed-point signals are only supported for the time domain. To use the parameters on this pane, make sure Time is selected for the Computation domain parameter on the Main pane.

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

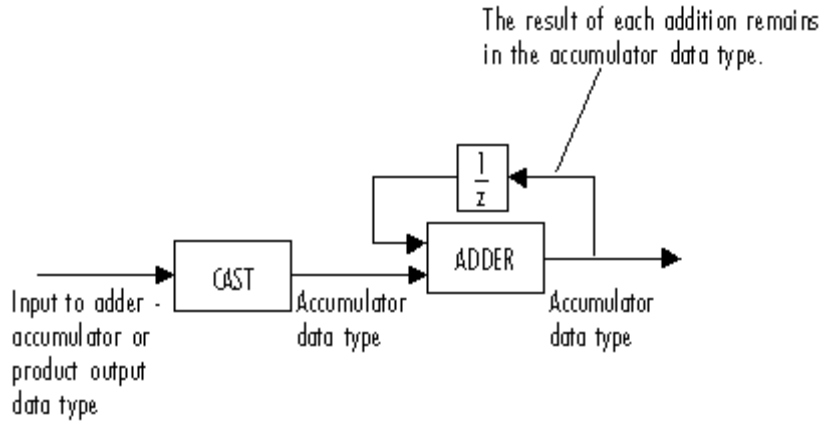
Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how you want to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-44 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Inherit via internal rule, the product output word length and fraction length are calculated automatically. For information about how the product output word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as first input, these characteristics match those of the first input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator



As depicted above, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how you want to designate this accumulator word and fraction lengths.

You also use this parameter to specify the accumulator word and fraction lengths resu lting from a complex-complex multiplication in the block. See “Multiplication Data Types” for more information.

- When you select Inherit via internal rule , the accumulator word length and fraction length are calculated automatically. For informati on about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as first input , these characteristics match those of the Þrst input to the block.

- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the word length and fraction length of the output of the block:

- When you select Same as accumulator , these characteristics match those of the accumulator.

  A special case occurs when Inherit via internal rule is specified for Accumulator , and both block inputs are complex. In that case, the output word length is one less than the accumulator word length.
- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as first input , these characteristics match those of the first input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Autocorrelation	Signal Processing Blockset
Convolution	Signal Processing Blockset
xcorr	Signal Processing Toolbox

Purpose

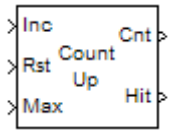
Count up or down through speci Þed range of numbers

Library

Signal Management / Switches and Counters

dspswit3

Description



The Counter block increments or decrements an internal counter each time it receives a trigger event at the Inc/Dec port. A trigger event at the Rst port resets the counter to its initial state.

The input to the Rst port must be a real, sample-based scalar. The input to the Inc/Dec port can be a real, sample-based scalar, or a real, frame-based vector (that is, a single channel). When both inputs are sample based, they must have th e same sample period. When the Inc/Dec input is frame based, the frame period must equal the sample period of the Rst input.

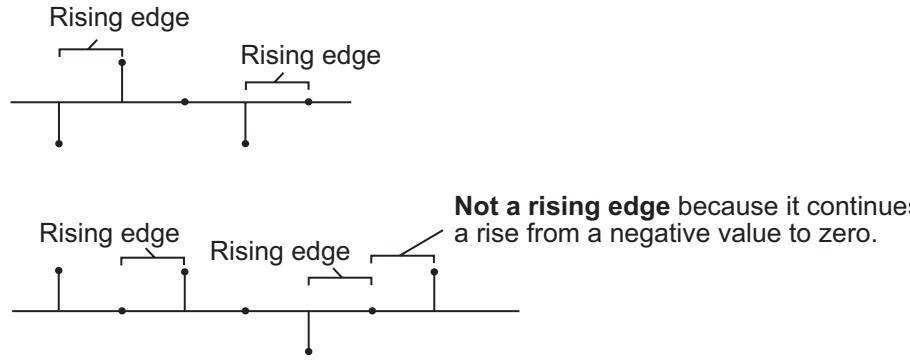
Sections of This Reference Page

- “Setting the Count Event Parameter” on page 2-235
- “Setting the Counter Size and Initial Count Parameters” on page 2-238
- “Sample-Based Operation” on page 2-238
- “Frame-Based Operation” on page 2-239
- “Free-Running Operation” on page 2-240
- “Examples” on page 2-240
- “Dialog Box” on page 2-243
- “Supported Data Types” on page 2-245
- “See Also” on page 2-246

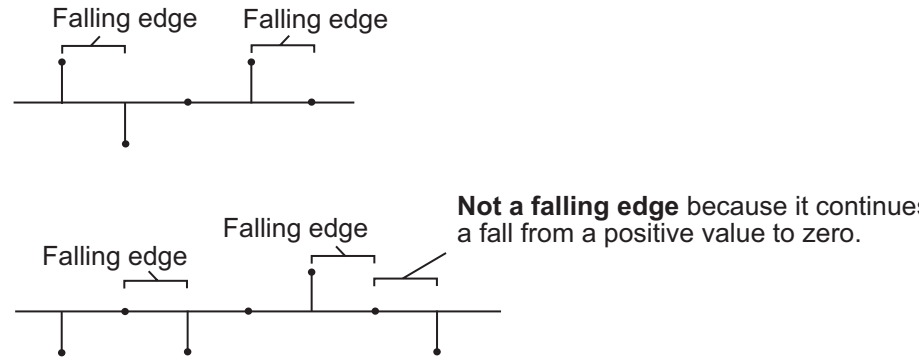
Setting the Count Event Parameter

Specify the trigger event for both inputs through the Count event parameter. Use one of the following valid values:

- Rising edge — Triggers a count or reset operation when the Inc/Dec or Rst input behaves in one of the following ways:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following Þgure)



- Falling edge — Triggers a count or reset operation when the Inc/Dec or Rst input behaves in one of the following ways:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the next Þgure)



- Either edge — Triggers a count or reset operation when the Inc/Dec or Rst input is a Rising edge or Falling edge (as described above).
- Non-zero sample — Triggers a count or reset operation at each sample time when the Inc/Dec or Rst input is not zero.
- Free running disables the Inc/Dec port and enables the Samples per output frame and Sample time parameters. The block increments or decrements the counter at a constant interval, T_s, specified by the Sample time parameter. For more information, see "Free-Running Operation" on page 2-240. The Rst port behaves as if the Count event parameter were set to Non-zero sample .

Note When running simulations in the Simulink MultiTasking mode, sample-based reset signals have a one-sample latency, and frame-based reset and clock signals have one frame of latency. Thus, there is a one-sample or one-frame delay between the time the block detects a trigger event at the Inc/Dec or Rst port and when it applies the trigger. For more information on latency and the Simulink tasking modes, see "Excess Algorithmic Delay (Tasking Latency)" and "Models with Multiple Sample Rates" in the Real-Time Workshop User's Guide .

Setting the Counter Size and Initial Count Parameters

At the start of the simulation, the block sets the counter to the value specified by the Initial count parameter, which can be any unsigned integer in the range defined by the Counter size parameter. The Counter size parameter allows you to choose from three standard counter ranges, or specify an arbitrary counter limit in the dialog box or via input port:

- 8 bits specifies a counter with a range of 0 to 255.
- 16 bits specifies a counter with a range of 0 to 65535.
- 32 bits specifies a counter with a range of 0 to 2^{32}-1.
- User defined enables the Maximum count parameter, which allows you to specify any arbitrary unsigned integer representable by the data type specified at the Max port as the upper-count limit. The range of the counter is then 0 to the Maximum count value. The data type at the Max port must not allow word lengths larger than the Counter data type.
- Specify via input port enables the Max port, which allows you to specify any arbitrary integer representable by the data type specified for the Counter data type parameter as the upper-count limit.

Sample-Based Operation

The block operates in sample-based mode when the Inc/Dec input is a sample-based scalar. Sample-based vectors and matrices are not accepted.

When the Count direction parameter is set to Up, a sample-based trigger event at the Inc (increment) input causes the block to increment the counter by one. The block continues incrementing the counter when triggered until the counter value reaches the upper-count limit. At the next Inc/Dec trigger event, the block resets the counter to 0 and resumes incrementing the counter with the subsequent Inc/Dec trigger event.

When the Count direction parameter is set to Down, a sample-based trigger event at the Dec (decrement) input causes the block to decrement

the counter by one. The block continues decrementing the counter when triggered until the counter value reaches 0. At the next Inc/Dec trigger event, the block resets the counter to the upper-count limit and resumes decrementing the counter with the subsequent Inc/Dec trigger event.

Between triggering events the block holds the output at its most recent value. The block resets the counter to its initial state when the trigger event specified in the Count event menu is received at the optional Rst input. When the Inc/Dec and Rst ports receive trigger events simultaneously, the block first resets the counter, and then increments or decrements appropriately. If you do not need to reset the counter during the simulation, you can disable the Rst port by clearing the Reset input check box.

The Output parameter provides three options for the output port configuration of the block icon:

- Count configures the block icon to show a Cnt port, which produces the current value of the counter as a sample-based scalar with the same sample period as the inputs.
- Hit configures the block icon to show a Hit port, which produces zeros while the value of the counter does not equal the value of any of the integers specified for the Hit values parameter. You can specify an integer scalar or a vector of integers for the Hit values parameter. When the counter value does equal any of the values specified for the Hit values parameter, the block generates a value of 1 at the Hit port. This option produces sample-based outputs with the same sample period as the inputs.
- Count and Hit configures the block icon with both ports.

Frame-Based Operation

Counter operates in frame-based mode when the Inc/Dec input is a frame-based vector. The block does not accept multichannel frame-based inputs.

Frame-based operation is the same as sample-based operation, except that the block increments or decrements the counter by the total

number of trigger events contained in the Inc/Dec input frame. When a trigger event splits across two consecutive frames, it is counted in the frame that contains the conclusion of the event. When the Rst port receives a trigger event Þrst, the block Þrst resets the counter, and then increments or decrements the counter by the number of trigger events contained in the Inc/Dec frame.

The Cnt and Hit outputs are sample-based scalars with sample period equal to the Inc/Dec input frame period.

Free-Running Operation

The block operates in free-running mode when you select Free running for the Count event parameter.

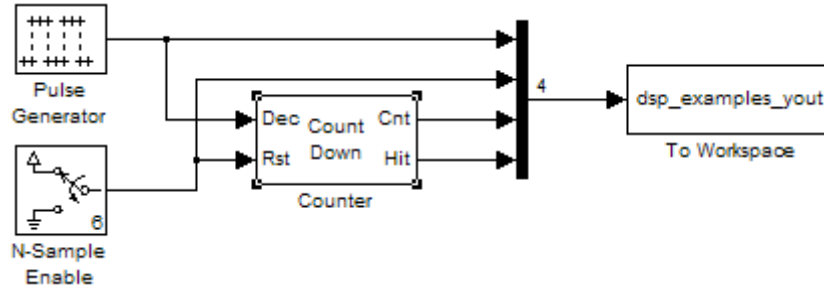
The Rst port behaves as if the Count event parameter were set to Non-zero sample : the block triggers a reset at each sample time that the Rst input is not zero.

The Inc/Dec input port is disabled in this mode, and the block simply increments or decrements the counter using the constant sample period speciÞed by the Sample time parameter, T_s.

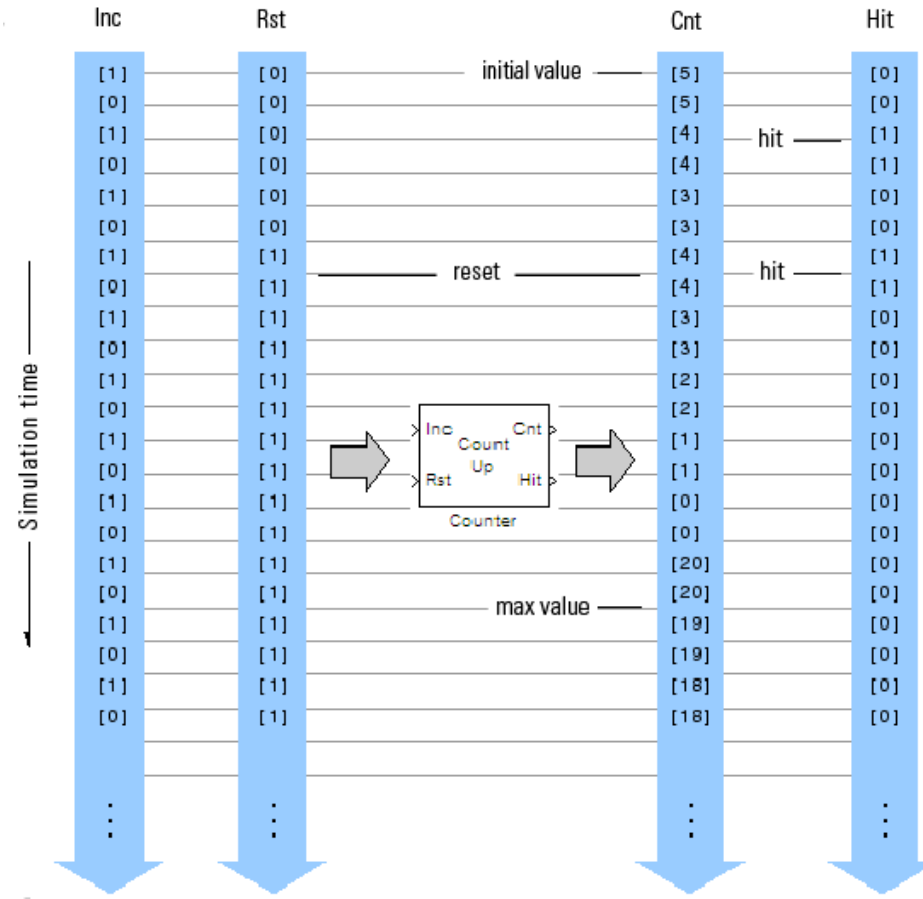
In this mode, the Cnt output is a frame-based M-by-1 matrix containing the count value at each of M consecutive sample times, where M is speciÞed by the Samples per output frame parameter. The Hit output is a frame-based M-by-1 matrix containing the hit status (0 or 1) at each of those M consecutive sample times. Both outputs have a frame period of $M*T_s$.

Examples

In the following model, the Simulin k Pulse Generator block drives the Dec port of the Counter block, and the N-Sample Enable block triggers the Rst port. All the Counter block's inputs and outputs are multiplexed into a single To Workspace block using a 4-port Mux block.

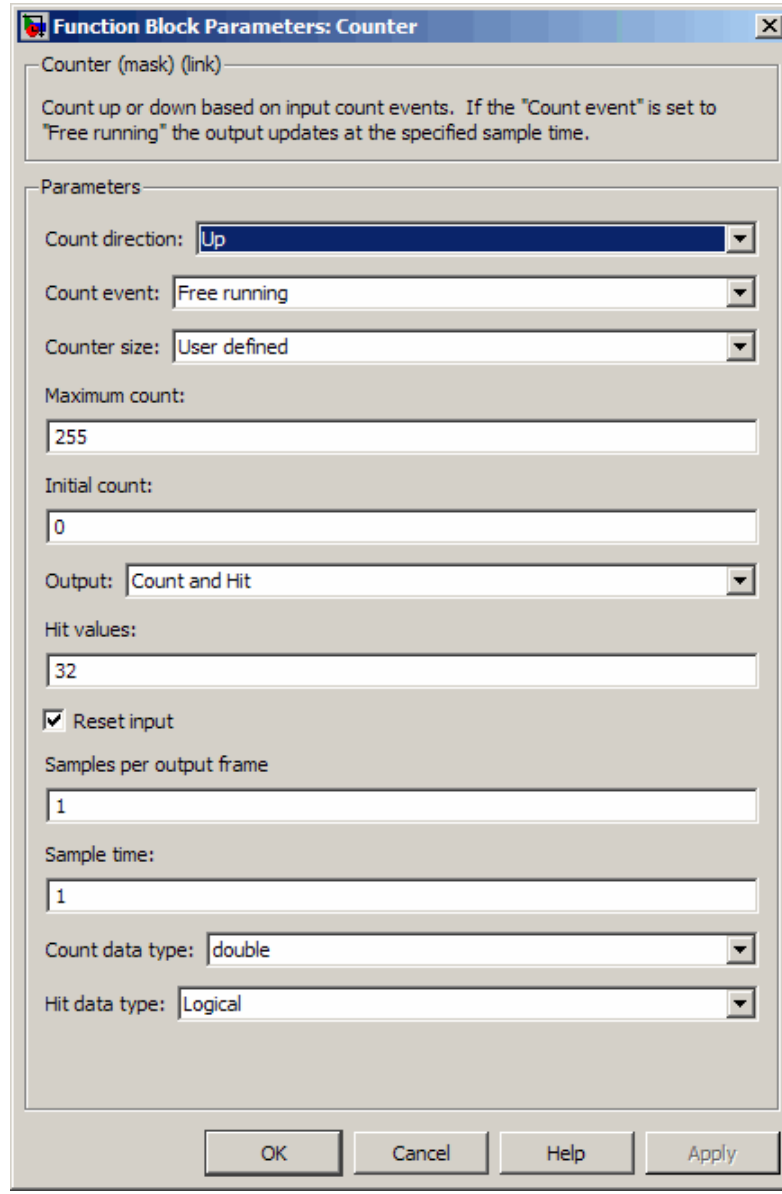


The following Þgure shows the Þrst 22 samples of the model's four-column output, yout .



You can see that the seventh input samples to both the Dec and Rst ports of the Counter block represent tr igger events (rising edges), so at this time step the block Þrst resets the counter to its initial value of 5, and then immediately decrements the count to 4. When the counter reaches its minimum value of 0, it rolls over to its maximum value of 20 with the subsequent trigger event at the Cnt port.

Dialog Box


Function Block Parameters: Counter
Counter (mask) (link)
Count up or down based on input count events. If the "Count event" is set to "Free running" the output updates at the specified sample time.
Parameters
Count direction: Up
Count event: Free running
Counter size: User defined
Maximum count:
255
Initial count:
0
Output: Count and Hit
Hit values:
32
Reset input
Samples per output frame
1
Sample time:
1
Count data type: double
Hit data type: Logical
OK
Cancel
Help
Apply

Count direction

Specify whether to count Up or Down. The port label on the block icon changes to Inc (increment) or Dec (decrement) based on the value of this parameter. Tunable in Simulink Normal mode.

Count event

Specify the type of event that triggers the block to increment, decrement, or reset the counter when received at the Inc/Dec or Rst ports. Free running disables the Inc/Dec port, and counts continuously with the period specified by the Sample time parameter. For more information on all the possible settings, see "Setting the Count Event Parameter" on page 2-235.

Counter size

Specify the range of integer values the block should count through before recycling to zero. For more information, see "Setting the Counter Size and Initial Count Parameters" on page 2-238.

Maximum count

Specify the counter's maximum value when Counter size is set to User defined. You can set the Maximum count parameter to any integer representable by the data type specified for the Counter data type parameter. Tunable in Simulink Normal mode.

Initial count

Specify the counter's initial value at the start of the simulation and after reset. Tunable.

Output

Select the output ports to enable: Cnt, Hit, or both.

Hit values

Specify a scalar or vector of integers whose occurrence in the count should be flagged by a 1 at the (optional) Hit output. This parameter is available when Hit or Count and Hit are selected in the Output menu. Tunable.

Reset input

Select to enable the Rst input port.

Samples per output frame

Specify the number of samples, M, in each output frame. This parameter is available when you select Free running in the Count event menu.

Sample time

Specify the output sample period, T_s, in free-running mode. This parameter is available when you select Free running in the Count event menu.

Count data type

Specify the data type of the Cnt output. This parameter is available when the Output parameter is set to Count or Count and Hit.

Hit data type

Specify the data type of the Hit output. This parameter is available when the Output parameter is set to Hit or the Output parameter is set to Count and Hit and the Count data type parameter is set to Double.

Supported Data Types

Port	Supported Data Types
Inc/Dec	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Rst	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Port	Supported Data Types
Max	• 8-, 16-, and 32-bit unsigned integers
Cnt	• Double-precision ßoating point • Single-precision ßoating point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Hit	• Logical • Boolean — The block might output Boolean values from the Hit output port depending on the setting of the Hit data type parameter.

See Also

Edge Detector Signal Processing Blockset

N-Sample Enable Signal Processing Blockset

N-Sample Switch Signal Processing Blockset

Purpose

Compute estimate of autoregressive (AR) model parameters using covariance method

Library

Estimation / Parametric Estimation

dspparest3

Description

The Covariance AR Estimator block uses the covariance method to fit an autoregressive (AR) model to the input data. This method minimizes the forward prediction error in the least squares sense.

The input is a sample-based vector (row, column, or 1-D) or frame-based vector (column only) representing a frame of consecutive time samples from a single-channel signal, which is assumed to be the output of an AR system driven by white noise. The block computes the normalized estimate of the AR system parameters, A(z), independently for each successive input frame.

$$H(z) = \frac{G}{A(z)} = \frac{G}{1 + a(2)z^{-1} + \ldots + a(p+1)z^{-p}}$$

The order, p, of the all-pole model is specified by the Estimation order parameter. To guarantee a valid output, you must set the Estimation order parameter to be less than or equal to half the input vector length.

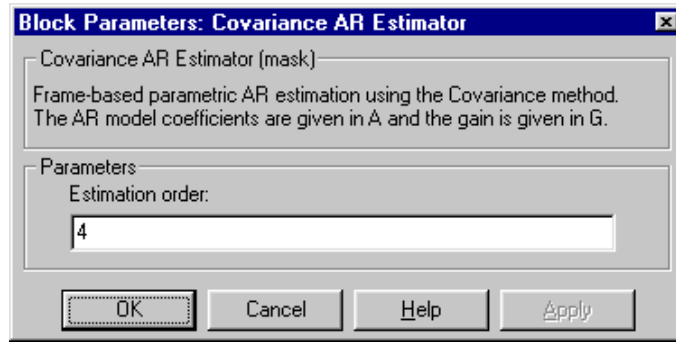
The top output, A, is a column vector of length p+1 with the same frame status as the input, and contains the normalized estimate of the AR model coefficients in descending powers of z.

```
[1  a(2)  ...  a(p+1)]
```

The scalar gain, G, is provided at the bottom output (G).

See the Burg AR Estimator block reference page for a comparison of the Burg AR Estimator, Covariance AR Estimator, Modified Covariance AR Estimator, and Yule-Walker AR Estimator blocks.

Dialog Box



Estimation order

The order of the AR model, p. To guarantee a nonsingular output, you must set p to be less than or equal to half the input length. Otherwise, the output might be singular.

References

Kay, S. M. Modern Spectral Estimation: Theory and Application. Englewood Cliffs, NJ: Prentice-Hall, 1988.

Marple, S. L., Jr., Digital Spectral Analy sis with Applications. Englewood Cliffs, NJ: Prentice-Hall, 1987.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point
A	• Double-precision ßoating point • Single-precision ßoating point
G	• Double-precision ßoating point • Single-precision ßoating point

See Also

Burg AR Estimator	Signal Processing Blockset
Covariance Method	Signal Processing Blockset
Modified Covariance AR Estimator	Signal Processing Blockset
Yule-Walker AR Estimator	Signal Processing Blockset
arcov	Signal Processing Toolbox

Covariance Method

Purpose

Compute parametric spectral estimate using covariance method

Library

Estimation / Power Spectrum Estimation

dspspect3

Description

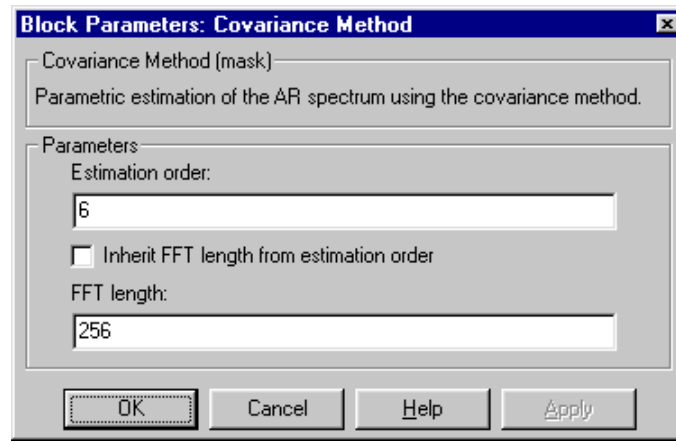
The Covariance Method block estimates the power spectral density (PSD) of the input using the covariance method. This method fits an autoregressive (AR) model to the signal by minimizing the forward prediction error in the least squares sense. The order of the all-pole model is the value specified by the Estimation order parameter, and the spectrum is computed from the FFT of the estimated AR model parameters. To guarantee a valid output, you must set the Estimation order parameter to be less than or equal to half the input vector length.

The input is a sample-based vector (row, column, or 1-D) or frame-based vector (column only) representing a frame of consecutive time samples from a single-channel signal. The block's output (a column vector) is the estimate of the signal's power spectral density at N_{fft} equally spaced frequency points in the range [0, F_s), where F_s is the signal's sample frequency.

When you select Inherit FFT length from estimation order, N_{fft} is one greater than the estimation order. When you do not select Inherit FFT length from estimation order, N_{fft} is specified as a power of 2 by the FFT length parameter, and the block zero pads or wraps the input to N_{fft} before computing the FFT. The output is always sample based.

See the Burg Method block reference for a comparison of the Burg Method, Covariance Method, Modified Covariance Method, and Yule-Walker Method blocks.

Dialog Box



Estimation order

The order of the AR model. To guarantee a nonsingular output, you must set the value of this parameter to be less than or equal to half the input length. Otherwise, the output might be singular.

Inherit FFT length from estimation order

When selected the FFT length is one greater than the estimation order.

FFT length

Enter the number of data points on which to perform the FFT, N_{fft}. When N_{fft} is larger than the input frame size, each frame is zero-padded as needed. When N_{fft} is smaller than the input frame size, each frame is wrapped as needed. This parameter is enabled when you clear the Inherit FFT length from estimation order check box.

References

Kay, S. M. Modern Spectral Estimation: Theory and Application. Englewood Cliffs, NJ: Prentice-Hall, 1988.

Marple, S. L., Jr., Digital Spectral Analy sis with Applications. Englewood Cliffs, NJ: Prentice-Hall, 1987.

Orfanidis, S. J. Introduction to Signal Processing . Englewood Cliffs, NJ: Prentice-Hall, 1995.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point
Output	• Double-precision floating point • Single-precision floating point

See Also

Burg Method	Signal Processing Blockset
Covariance AR Estimator	Signal Processing Blockset
Modified Covariance Method	Signal Processing Blockset
Short-Time FFT	Signal Processing Blockset
Yule-Walker Method	Signal Processing Blockset
pcov	Signal Processing Toolbox

See “Power Spectrum Estimation” for related information.

Purpose

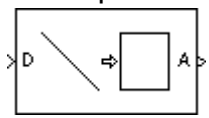
Create square diagonal matrix from diagonal elements

Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

dspmtrx3

Description

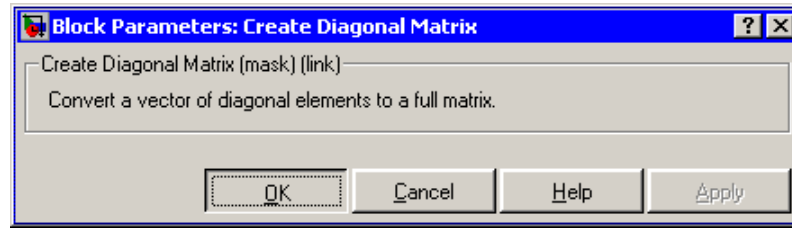


The Create Diagonal Matrix block populates the diagonal of the M-by-M matrix output with the eleme nts contained in the length- M vector input, D. The elements off the diagonal are zero.

A = diag(D) Equivalent MATLAB code

The output is always sample based.

Dialog Box



Supported Data Types

Port	Supported Data Types
D	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
A	• Double-precision ßoating point • Single-precision ßoating point

Port	Supported Data Types
	• Fixed point (signed and unsigned)
Constant Diagonal Matrix	• Boolean • 8-, 16-, and 32-bit signed integers
Extract Diagonal	• 8-, 16-, and 32-bit unsigned integers
diag	

See Also

Constant Diagonal Matrix Signal Processing Blockset

Extract Diagonal Signal Processing Blockset

diag MATLAB

Purpose

Compute cumulative product of channel, column, or row elements

Library

Math Functions / Math Operations

dspmathops

Description

The Cumulative Product block computes the cumulative product of elements in each channel, column, or row of the M-by-N input matrix.

The inputs can be sample-based or frame-based vectors and matrices. The output always has the same dimensions, rate, frame status, data type, and complexity as the input.

The Cumulative Product block accepts real and complex fixed-point and floating-point inputs except for complex unsigned fixed-point inputs.

- "Valid Input" on page 2-255
- "Valid Reset Signal" on page 2-256
- "Output Characteristics" on page 2-256
- "Multiplying Along Channels of Frame-Based Inputs" on page 2-256
- "Multiplying Along Channels of Sample-Based Inputs" on page 2-257
- "Resetting the Cumulative Product Along Channels" on page 2-258
- "Multiplying Along Columns" on page 2-260
- "Multiplying Along Rows" on page 2-261
- "Dialog Box" on page 2-263
- "Supported Data Types" on page 2-268
- "See Also" on page 2-268

Valid Input

The block computes the cumulative product of both sample- and frame-based vector and matrix inputs. Inputs can be real or complex. When multiplying along channels or columns, 1-D unoriented vectors

are treated as column vectors. When multiplying along rows, 1-D vectors are treated as row vectors.

Valid Reset Signal

The optional reset port, Rst , accepts scalar values, which can be any built-in Simulink data type including boolean . The rate of the reset signal must be a positive integer multiple of the rate of the data signal input.

Output Characteristics

The output always has the same dimensions, rate, frame status, data type, and complexity as the data signal input.

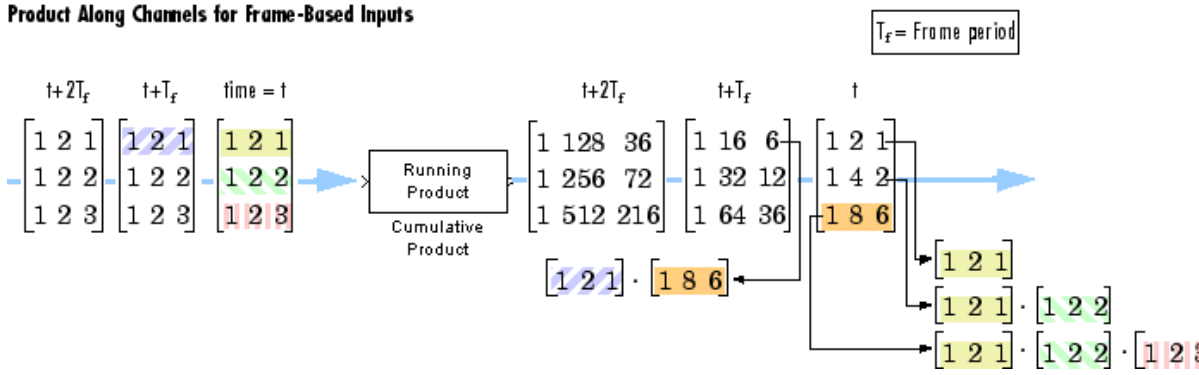
Multiplying Along Channels of Frame-Based Inputs

For frame-based inputs, the bloc k treats each input column as an independent channel. As the following Þgure and equation illustrate, the output has the following characteristics:

- The Þrst row of the Þrst output is the same as the Þrst row of the Þrst input.
- The Þrst row of each subsequent output is the element-wise product of the Þrst row of the current input (time t), and the last row of the previous output (time $t - T_f$, where T_f is the frame period).
- The output has the same size, dime nsion, frame sta tus, data type, and complexity as the input.

Given an M-by-N frame-based input, u, the output, y, is a frame-based M-by-N matrix whose Þrst row has elements

$$y_{1,j}(t) = u_{1,j}(t) \cdot y_{M,j}(t - T_f)$$



Multiplying Along Channels of Sample-Based Inputs

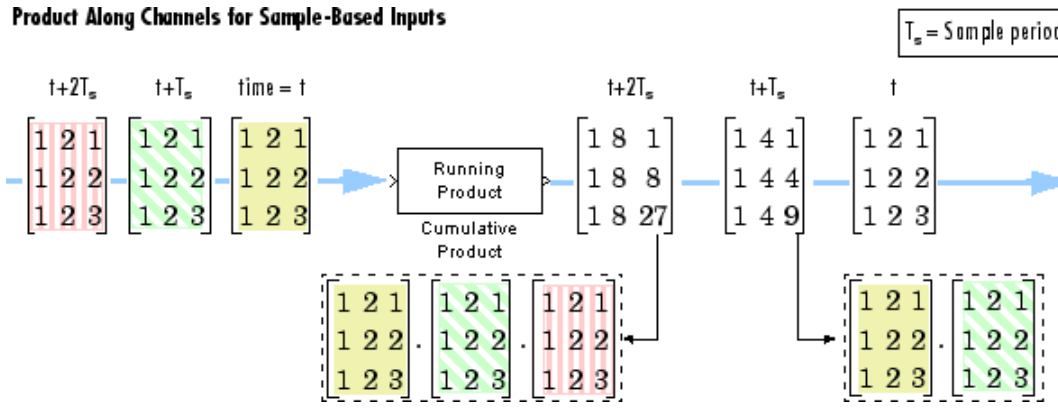
For sample-based inputs, the block treats each element of the input matrix as an independent channel. As the following Þgure and equation illustrate, the output has the following characteristics:

- The Þrst output is the same as the Þrst input.
- Each subsequent output is the ele ment-wise product of the current input (time t) and the previous output (time t - T_s, where T_s is the sample period).
- The output has the same size, dime nsion, frame sta tus, data type, and complexity as the input.

Given an M-by-N sample-based input, u, the output, y, is a sample-based M-by-N matrix with the elements

$$y_{i,j}(t) = u_{i,j}(t) \cdot y_{i,j}(t - T_s) \quad \begin{matrix} 1 \le i \le M \\ 1 \le j \le N \end{matrix}$$

For convenience, length- M 1-D vector inputs are treated as M-by-1 column vectors when multiplying along channels, and the output is a length- M 1-D vector.



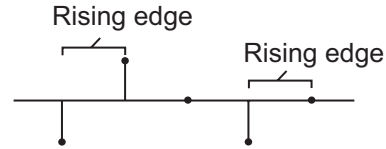
Resetting the Cumulative Product Along Channels

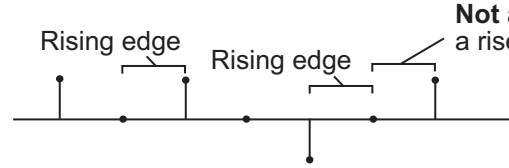
When you set the Multiply input along parameter to Channels (running product) , you can set the block to reset the running product whenever it detects a reset event at the optional Rst port. The reset sample time must be a positive integer multiple of the input sample time. The input to the Rst port can be of the Boolean data type.

When the block is reset for sample-based inputs, the block initializes the current output to the values of t he current input. For frame-based inputs, the block initializes the Þrst row of the current output to the values in the Þrst row of the current input.

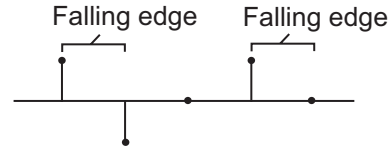
The Reset port parameter speci Þes the reset event, which can be one of the following:

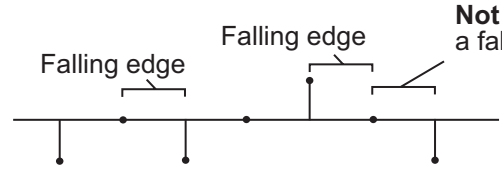
- None disables the Rst port.
- Rising edge — Triggers a reset operation when the Rst input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following Þgure)





- Falling edge — Triggers a reset operation when the Rst input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)





- Either edge — Triggers a reset o peration when the Rst input is a Rising edge or Falling edge (as described above)
- Non-zero sample — Triggers a reset opera tion at each sample time that the Rst input is not zero

Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see "Excess Algorithmic Delay (Tasking Latency)" and "Models with Multiple Sample Rates" in the Real-Time Workshop User's Guide.

Multiplying Along Columns

When the Multiply input along parameter is set to Columns, the block computes the cumulative product of each column of the input, where the current cumulative product is independent of the cumulative products of previous inputs.

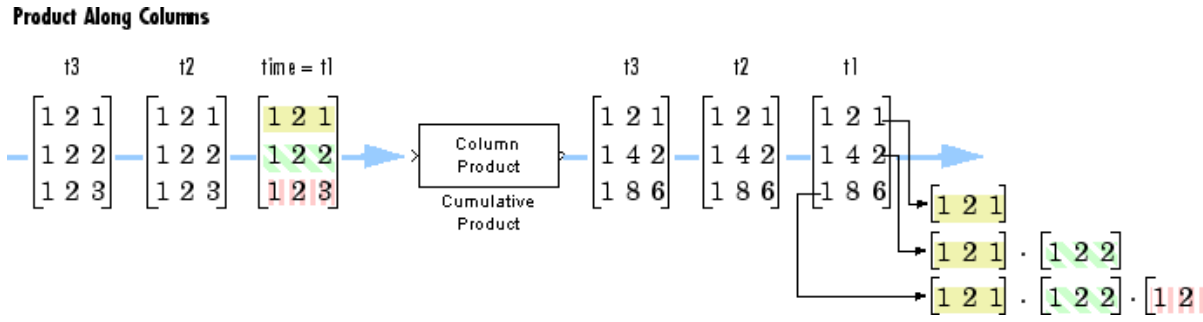
```
y = cumprod(u)          % Equivalent MATLAB code
```

The output has the same size, dimension, frame status, data type, and complexity as the input. The mth output row is the element-wise product of the first m input rows.

Given an M-by-N input, u, the output, y, is an M-by-N matrix whose jth column has elements

$$y_{i,j} = \prod_{k=1}^{i} u_{k,j} \qquad 1 \le i \le M$$

The block treats length-M 1-D vector inputs as M-by-1 column vectors when multiplying along columns.



Multiplying Along Rows

When the Multiply input along parameter is set to Rows, the block computes the cumulative product of the row elements, where the current cumulative product is independent of the cumulative products of previous inputs.

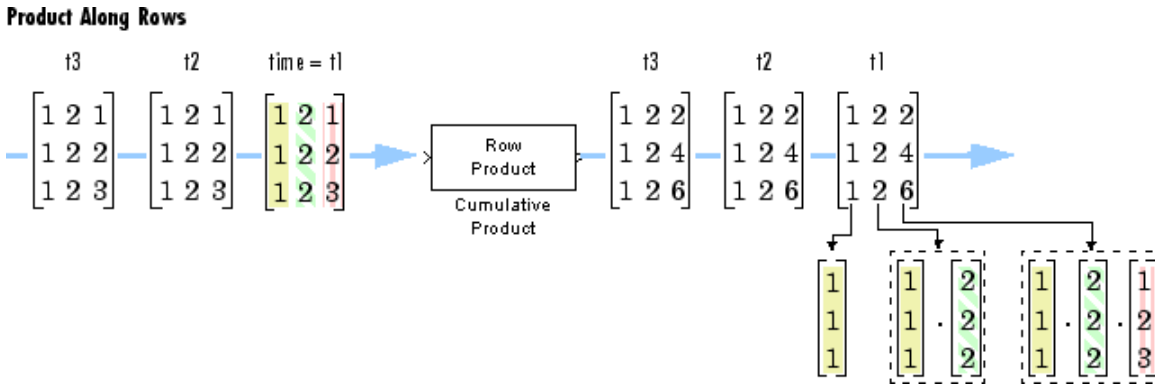
```
y = cumprod(u,2)          % Equivalent MATLAB code
```

The output has the same size, dimension, frame status, and data type as the input. The nth output column is the element-wise product of the first n input columns.

Given an M-by-N input, u, the output, y, is an M-by-N matrix whose ith row has elements

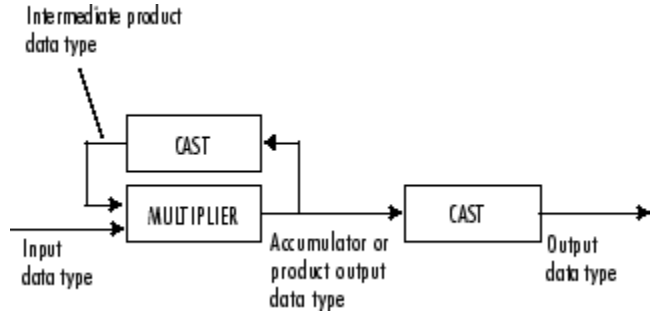
$$y_{i,j} = \prod_{k=1}^{j} u_{i,k} \qquad 1 \le j \le N$$

The block treats length-N 1-D vector inputs as 1-by-N row vectors when multiplying along rows.



Fixed-Point Data Types

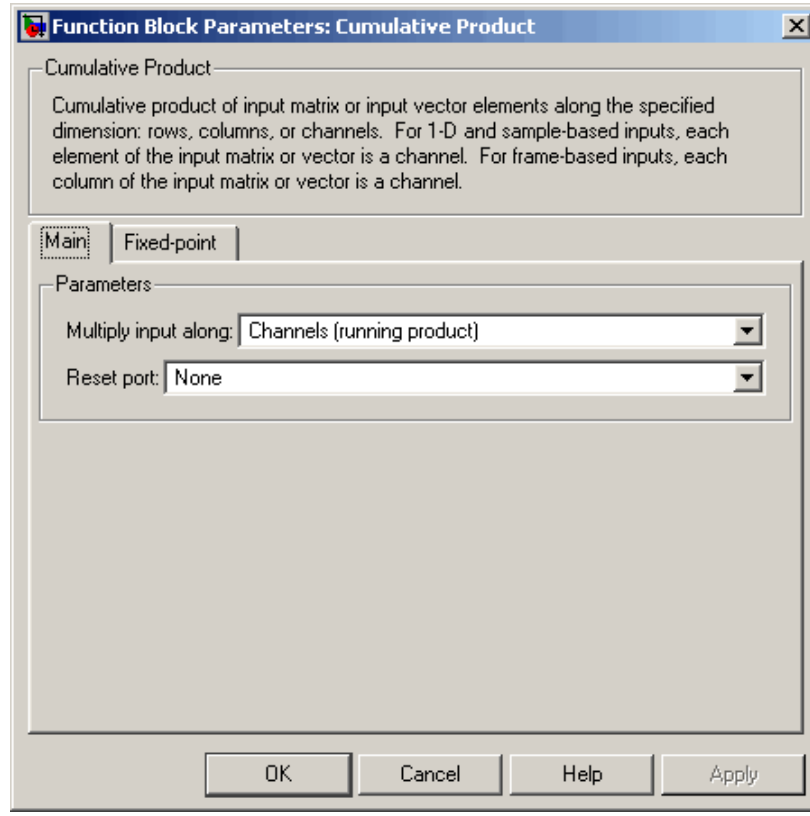
The following diagram shows the data types used within the Cumulative Product block for fixed-point signals.



The output of the multiplier is in the product output data type when at least one of the inputs to the multiplier is real. When both of the inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see “Multiplication Data Types”. You can set the accumulator, product output, intermediate product, and output data types in the block dialog as discussed in “Dialog Box” on page 2-263.

Dialog Box

The Main pane of the Cumulative Product block dialog appears as follows.



Multiply input along

The dimension along which to compute the cumulative products. The options allow you to multiply along Channels (running product), Columns, and Rows. For more information, see the following sections:

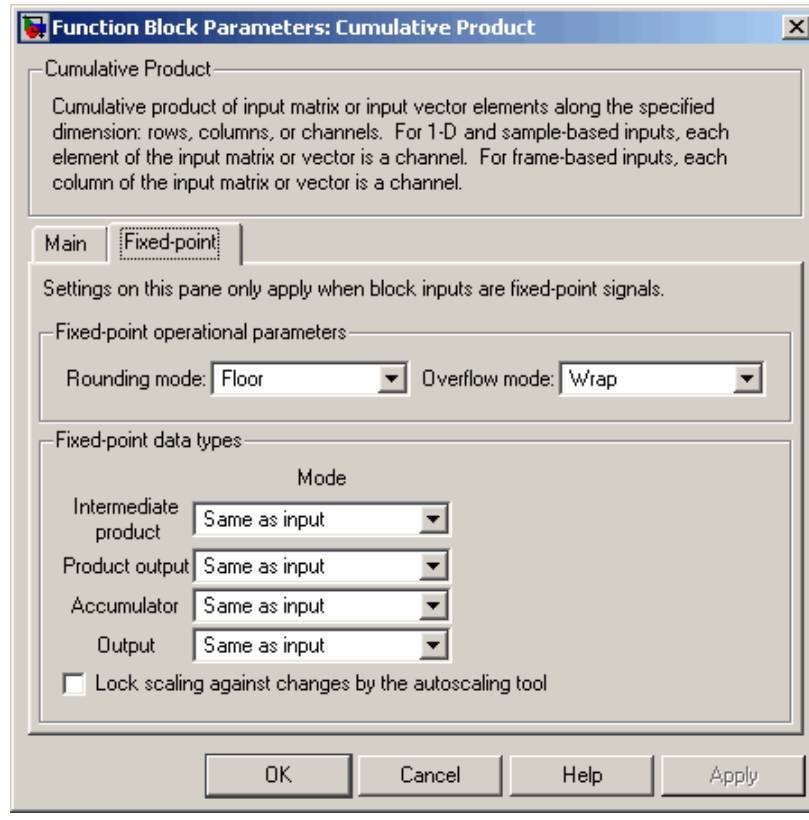
- "Multiplying Along Channels of Frame-Based Inputs" on page 2-256

- “Multiplying Along Channels of Sample-Based Inputs” on page 2-257
- “Multiplying Along Columns” on page 2-260
- “Multiplying Along Rows” on page 2-261

Reset port

Determines the reset event that causes the block to reset the product along channels. The rate of the reset signal must be a positive integer multiple of the rate of the data signal input. This parameter is enabled only when you set the Multiply input along parameter to Channels (running product) . For more information, see “Resetting the Cumulative Product Along Channels” on page 2-258.

The Fixed-point pane of the Cumulative Product block dialog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Intermediate product

As shown in “Fixed-Point Data Types” on page 2-262, the output of the multiplier is cast to the intermediate product data type before the next element of the input is multiplied into it. Use

this parameter to specify how you would like to designate the intermediate product word and fraction lengths:

- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the intermediate product, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the intermediate product. This block requires power-of-two slope and a bias of zero.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-262 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths. See “Fixed-Point Data Types” on page 2-262 and “Multiplication Data Types” for illustrations depicting the use of the accumulator data type in this block. Note that the accumulator data type is only used when both inputs to the multiplier are complex:

- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the word length and fraction length of the output of the block:

- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any Þxed-point scaling you specify in this block mask f rom being overridden by the autoscaling tool in the Fixed-Point Tool.

Supported Data Types

Input and Output Ports	Supported Data Types
Data input port, In	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Reset input port, Rst	All built-in Simulink data types: • Double-precision ßoating point • Single-precision ßoating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output port	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Cumulative Sum	Signal Processing Blockset
Matrix Product	Signal Processing Blockset
cumprod	MATLAB

Purpose

Compute cumulative sum of channel, column, or row elements

Library

Math Functions / Math Operations

dspmathops

Description

The Cumulative Sum block computes the cumulative sum of the elements in each channel, column, or row of the M-by-N input matrix.

The inputs can be sample-based or frame-based vectors and matrices. The output always has the same dimensions, rate, frame status, data type, and complexity as the input.

The Cumulative Sum block accepts real and complex fixed-point and floating-point inputs except for complex unsigned fixed-point inputs.

Sections of This Reference Page

- "Input and Output Characteristics" on page 2-269
- "Summing Along Channels" on page 2-270
- "Resetting the Cumulative Sum Along Channels" on page 2-272
- "Summing Along Columns" on page 2-274
- "Summing Along Rows" on page 2-275
- "Dialog Box" on page 2-277
- "Supported Data Types" on page 2-280
- "See Also" on page 2-280

Input and Output Characteristics

Valid Input

The block computes the cumulative sum of both sample- and frame-based vector and matrix inputs. Inputs can be real or complex. When summing along channels or columns, 1-D unoriented vectors are treated as column vectors. When summing along rows, 1-D vectors are treated as row vectors.

Valid Reset Signal

The optional reset port, Rst , accepts scalar values, which can be any built-in Simulink data type including boolean . The rate of the reset signal must be a positive integer multiple of the rate of the data signal input.

Output Characteristics

The output always has the same dimensions, rate, frame status, data type, and complexity as the data signal input.

Summing Along Channels

When the Sum input along parameter is set to Channels (running sum) , the block computes the cumulative sum of the elements in each input channel. The running sum of the current input takes into account the running sum of all previous inputs. See the following sections for more information:

- “Summing Along Channels of Frame-Based Inputs” on page 2-270
- “Summing Along Channels of Sample-Based Inputs” on page 2-271
- “Resetting the Cumulative Sum Along Channels” on page 2-272

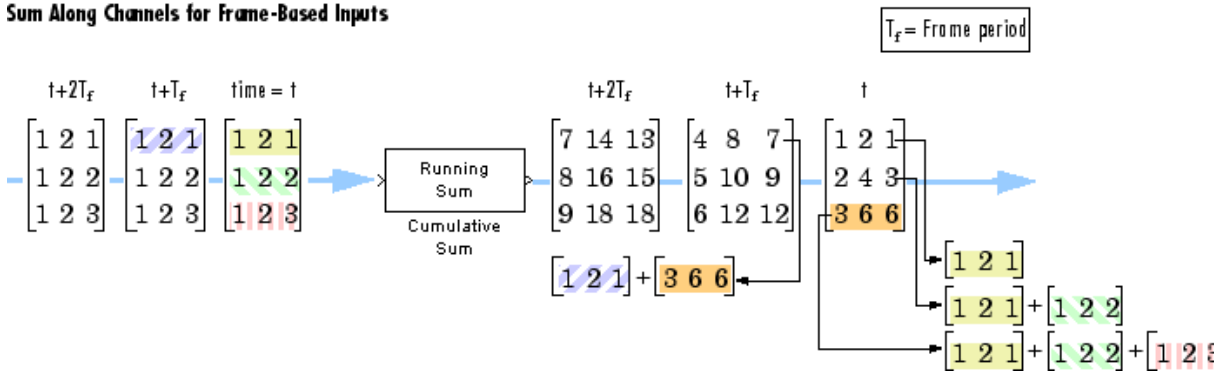
Summing Along Channels of Frame-Based Inputs

For frame-based inputs, the bloc k treats each input column as an independent channel. As the following Þgure and equation illustrate, the output has the following characteristics:

- The Þrst row of the Þrst output is the same as the Þrst row of the Þrst input.
- The Þrst row of each subsequent output is the sum of the Þrst row of the current input (time t), and the last row of the previous output (time $t - T_f$, where T_f is the frame period).
- The output has the same size, dime nsion, frame sta tus, data type, and complexity as the input.

Given an M-by-N frame-based input, u, the output, y, is a frame-based M-by-N matrix whose first row has elements

$$y_{1,j}(t) = u_{1,j}(t) + y_{M,j}(t - T_f)$$



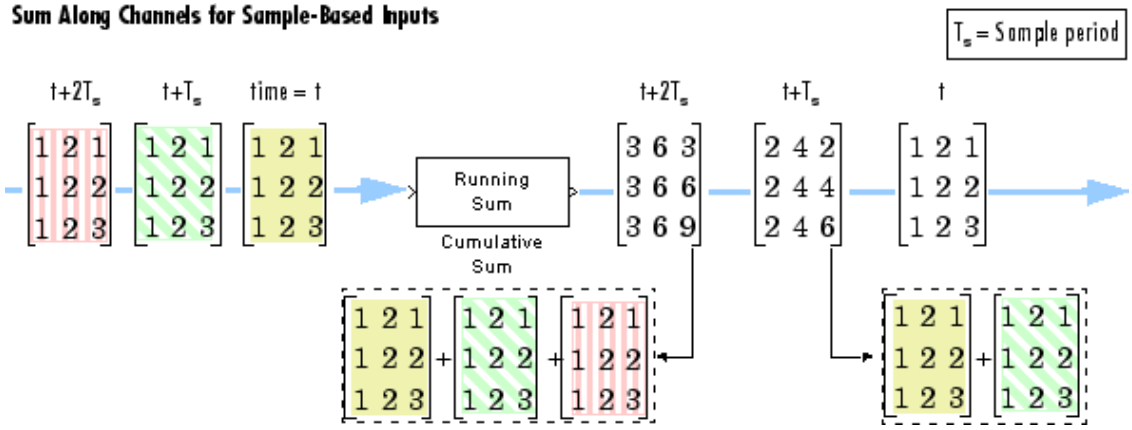
Summing Along Channels of Sample-Based Inputs

For sample-based inputs, the block treats each element of the input matrix as an independent channel. As the following figure and equation illustrate, the output has the following characteristics:

- The first output is the same as the first input.
- Each subsequent output is the sum of the current input (time t) and the previous output (time t - T_s, where T_s is the sample period).
- The output has the same size, dimension, frame status, data type, and complexity as the input.

Given an M-by-N sample-based input, u, the output, y, is a sample-based M-by-N matrix with the elements

$$y_{i,j}(t) = u_{i,j}(t) + y_{i,j}(t - T_s) \qquad \begin{matrix} 1 \le i \le M \\ 1 \le j \le N \end{matrix}$$



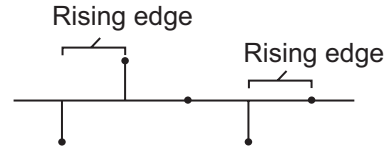
Resetting the Cumulative Sum Along Channels

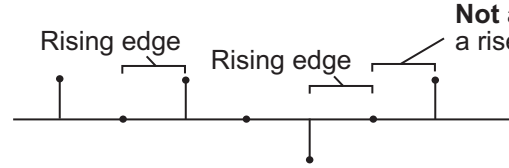
When you set the Sum input along parameter to Channels (running sum), you can set the block to reset the running sum whenever it detects a reset event at the optional Rst port. The reset sample time must be a positive integer multiple of the input sample time. The input to the Rst port can be of the boolean data type.

When the block is reset for sample-based inputs, the block initializes the current output to the values of the current input. For frame-based inputs, the block initializes the Þrst row of the current output to the values in the Þrst row of the current input.

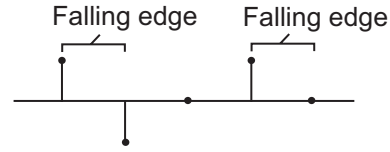
The Reset port parameter speciÞes the reset event, which can be one of the following:

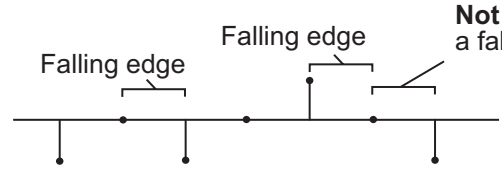
- None disables the Rst port.
- Rising edge — Triggers a reset operation when the Rst input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following Þgure)





- Falling edge — Triggers a reset operation when the Rst input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)





- Either edge — Triggers a reset o peration when the Rst input is a Rising edge or Falling edge (as described above)
- Non-zero sample — Triggers a reset opera tion at each sample time that the Rst input is not zero

Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see "Excess Algorithmic Delay (Tasking Latency)" and the topic on models with multiple sample rates in the Real-Time Workshop documentation.

Summing Along Columns

When the Sum input along parameter is set to Columns, the block computes the cumulative sum of each column of the input, where the current cumulative sum is independent of the cumulative sums of previous inputs.

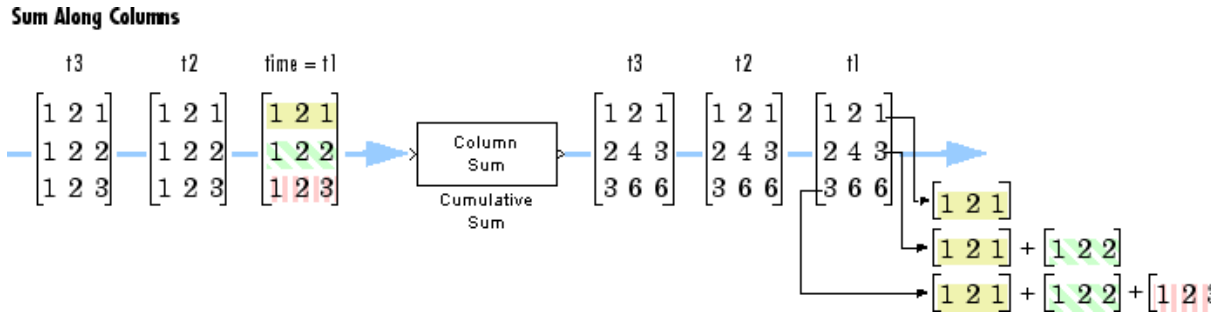
```
y = cumsum(u)          % Equivalent MATLAB code
```

The output has the same size, dimension, frame status, data type, and complexity as the input. The mth output row is the sum of the first m input rows.

Given an M-by-N input, u, the output, y, is an M-by-N matrix whose jth column has elements

$$y_{i,j} = \sum_{k=1}^{j} u_{k,j} \qquad 1 \le i \le M$$

The block treats length-M 1-D vector inputs as M-by-1 column vectors when summing along columns.



Summing Along Rows

When the Sum input along parameter is set to Rows, the block computes the cumulative sum of the row elements, where the current cumulative sum is independent of the cumulative sums of previous inputs.

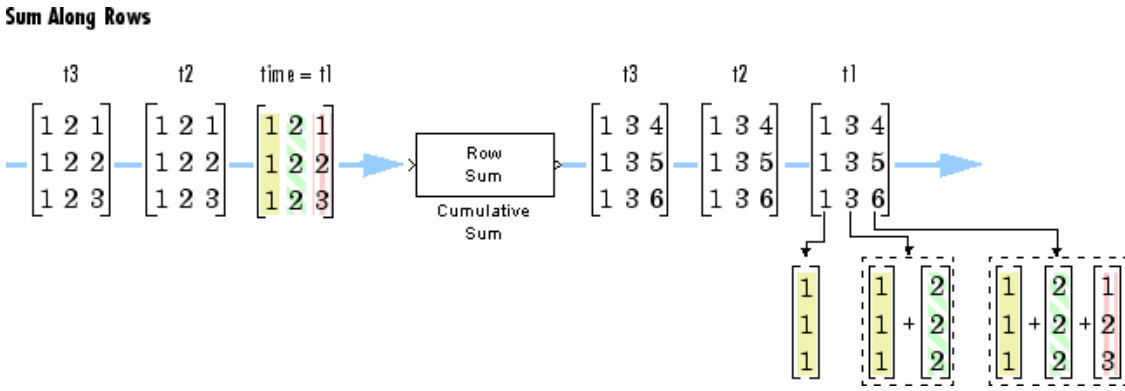
```
y = cumsum(u,2)          % Equivalent MATLAB code
```

The output has the same size, dimension, frame status, and data type as the input. The nth output column is the sum of the first n input columns.

Given an M-by-N input, u, the output, y, is an M-by-N matrix whose ith row has elements

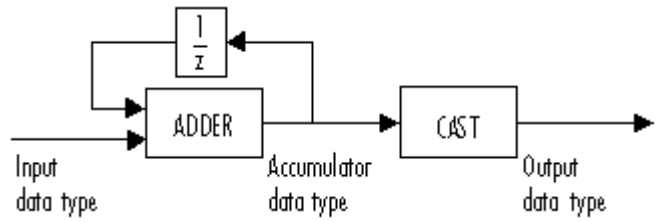
$$y_{i,j} = \sum_{k=1}^{j} u_{i,k} \qquad 1 \le j \le N$$

The block treats length- N 1-D vector inputs as 1-by- N row vectors when summing along rows.



Fixed-Point Data Types

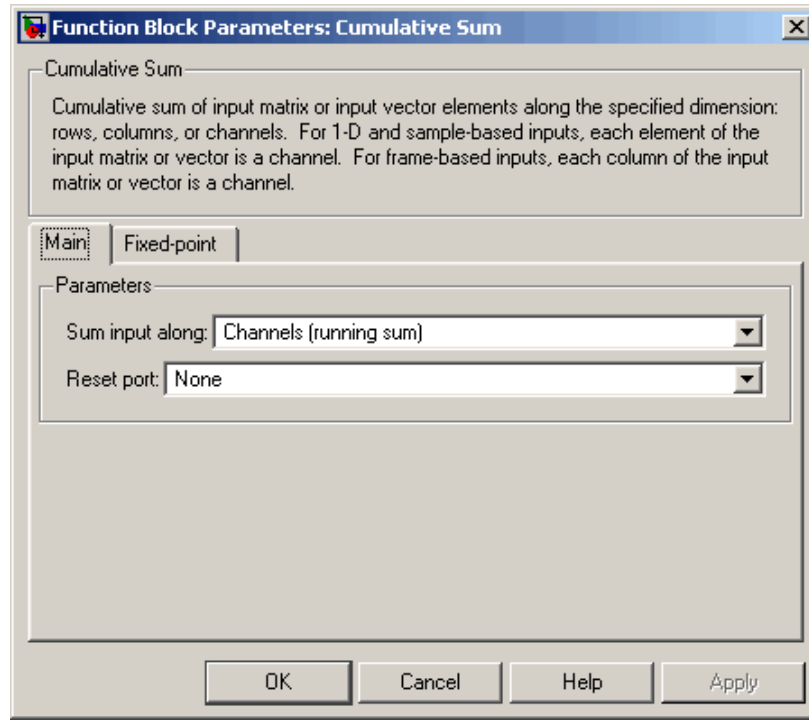
The following diagram shows the data types used within the Cumulative Sum block for fixed-point signals.



You can set the accumulator and output data types in the block dialog as discussed in “Dialog Box” on page 2-277.

Dialog Box

The Main pane of the Cumulative Sum block dialog appears as follows.



Sum input along

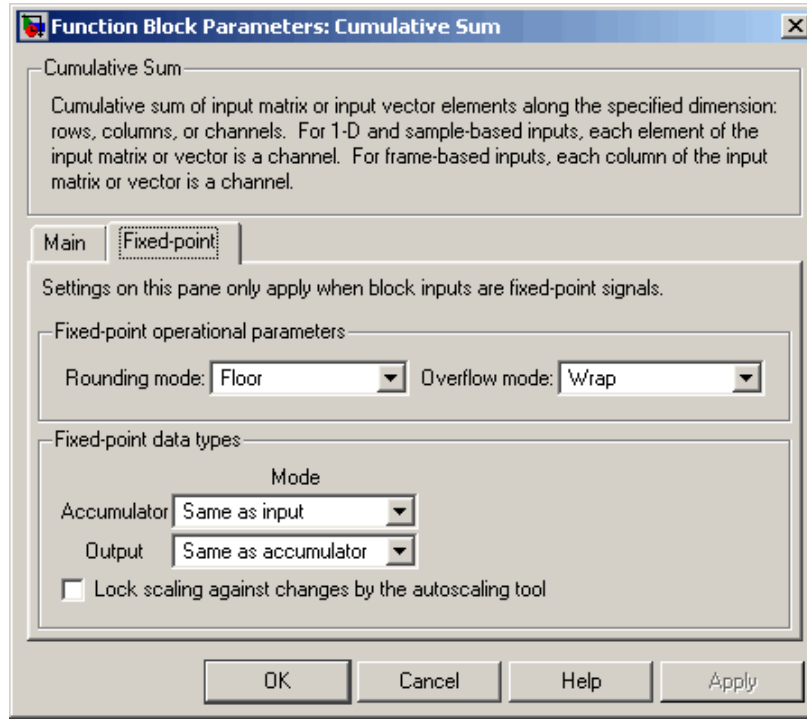
The dimension along which to compute the cumulative summations. The options allow you to sum along Channels (running sum) , Columns , and Rows. For more information, see the following sections:

- “Summing Along Channels” on page 2-270
- “Summing Along Columns” on page 2-274
- “Summing Along Rows” on page 2-275

Reset port

Determines the reset event that causes the block to reset the sum along channels. The rate of the reset signal must be a positive integer multiple of the ra te of the data signal input. This parameter is enabled only when you set the Sum input along parameter to Channels (running sum) . For more information, see “Resetting the Cumulativ e Sum Along Channels” on page 2-272.

The Fixed-point pane of the Cumulative Sum block dialog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Accumulator

Use this parameter to specify how you would like to designate this accumulator word and fraction lengths:

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as accumulator, these characteristics match those of the accumulator.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool.

Supported Data Types

Input and Output Ports	Supported Data Types
Data input port, In	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Reset input port, Rst	All built-in Simulink data types: • Double-precision ßoating point • Single-precision ßoating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output port	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Cumulative Product	Signal Processing Blockset
Difference	Signal Processing Blockset
Matrix Sum	Signal Processing Blockset
cumsum	MATLAB

Purpose

Convert input signal to specified data type

Library

Signal Management / Signal Attributes

dspsigattribs

Description

The Data Type Conversion block is an implementation of the Simulink Data Type Conversion block. See Data Type Conversion for more information.

dB Conversion

Purpose

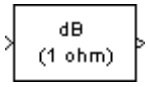
Convert magnitude data to decibels (dB or dBm)

Library

Math Functions / Math Operations

dspmathops

Description



The dB Conversion block converts a linearly scaled power or amplitude input to dB or dBm. The reference power is 1 Watt for conversions to dB and 1 mWatt for conversions to dBm. The Input signal parameter specifies whether the input is a power signal or a voltage signal, and the Convert to parameter controls the scaling of the output. When selected, the Add eps to input to protect against "log(0) = -inf" parameter adds a value of eps to all power and voltage inputs. When this option is not enabled, zero-valued inputs produce -inf at the output.

The size and frame status of the output are the same as the input.

Power Inputs

Select Power as the Input signal parameter when the input, u, is a real, nonnegative, power signal (units of watts). When the Convert to parameter is set to dB, the block performs the dB conversion

```
y = 10*log10(u)          % Equivalent MATLAB code
```

When the Convert to parameter is set to dBm, the block performs the dBm conversion

```
y = 10*log10(u) + 30
```

The dBm conversion is equivalent to performing the dB operation after converting the input to milliwatts.

Voltage Inputs

Select Amplitude as the Input signal parameter when the input, u, is a real voltage signal (units of volts). The block uses the scale factor specified in ohms by the Load resistance parameter, R, to convert the voltage input to units of power (watts) before converting to dB or dBm.

When the Convert to parameter is set to dB, the block performs the dB conversion

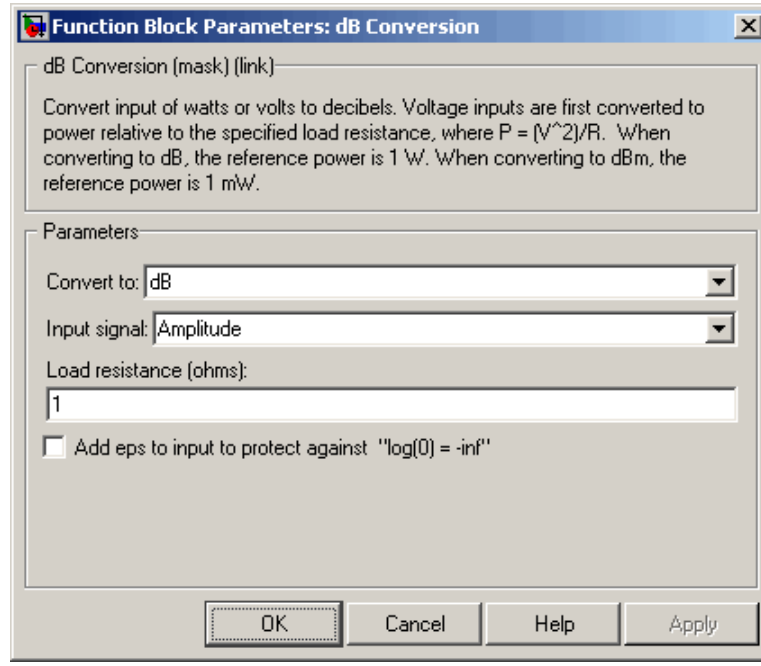
```
y = 10*log10(abs(u)^2/R)
```

When the Convert to parameter is set to dBm, the block performs the dBm conversion

```
y = 10*log10(abs(u)^2/R) + 30
```

The dBm conversion is equivalent to performing the dB operation after converting the (abs(u)^2/R) result to milliwatts.

Dialog Box



Convert to
: The logarithmic scaling to which the input is converted, dB or dBm The reference power is 1 W for conversions to dB and 1 mW for conversions to dBm. Tunable.

Input signal
: The type of input signal, Power or Amplitude .

Load resistance
: The scale factor used to convert voltage inputs to units of power. Tunable.

Add eps to input to protect against “log(0) = -inf”
: When selected, adds eps to all input values (power or voltage). Tunable.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

dB Gain	Signal Processing Blockset
Math Function	Simulink
log10	MATLAB

Purpose

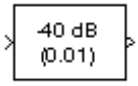
Apply decibel gain

Library

Math Functions / Math Operations

dspmathops

Description



The dB Gain block multiplies the input by the decibel values specified in the Gain parameter. For an M-by-N input matrix u with elements u_{ij}, the Gain parameter can be a real M-by-N matrix with elements g_{ij} to be multiplied element-wise with the input, or a real scalar.

$$y_{ij} = 10u_{ij}^{(g_{ij}/k)}$$

The value of k is 10 for power signals (select Power as the Input signal parameter) and 20 for voltage signals (select Amplitude as the Input signal parameter).

The value of the equivalent linear gain

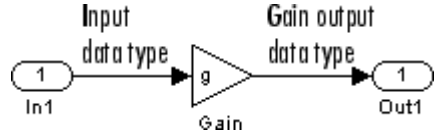
$$g_{ij}^{lin} = 10^{(g_{ij}/k)}$$

is displayed in the block icon below the dB gain value. The size and frame status of the output are the same as the input.

The dB Gain block supports real and complex floating-point and fixed-point data types.

Fixed-Point Data Types

The following diagram shows the data types used within the dB Gain subsystem block for fixed-point signals.

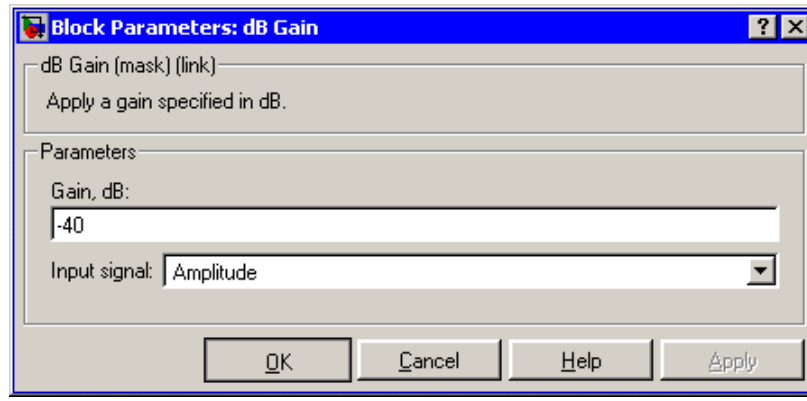


The settings for the fixed-point parameters of the Gain block in the diagram above are as follows:

- Integer rounding mode: Floor
- Saturate on integer over ßow — unselected
- Parameter data type mode — Inherit via internal rule
- Output data type mode — Inherit via internal rule

See the Gain reference page for more information.

Dialog Box



Gain

The dB gain to apply to the input, a scalar or a real M-by-N matrix. Tunable.

Input signal

The type of input signal: Power or Amplitude . Tunable.

Note This block does not support tunability in generated code.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Fixed point (signed and unsigned)

- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

See Also

dB Conversion	Signal Processing Blockset
Math Function	Simulink
log10	MATLAB

Purpose

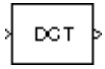
Compute discrete cosine transform (DCT) of input

Library

Transforms

dspxfrm3

Description



The DCT block computes the unitary discrete cosine transform (DCT) of each channel in the M-by-N input matrix, u.

```
y = dct(u)        % Equivalent MATLAB code
```

When the input is a sample-based row vector, the DCT block computes the discrete cosine transform across the vector dimension of the input. For all other sample-based N-D arrays, the block computes the DCT across the first dimension of the input.

For both sample-based and frame-based inputs, the block assumes that each input column is a frame containing M consecutive samples from an independent channel. The frame size, M, must be a power of two. To work with other frame sizes, use the Pad block to pad or truncate the frame size to a power-of-two length.

When the input is an M-by-N matrix, the DCT block outputs an M-by-N matrix whose lth column contains the length- M DCT of the corresponding input column.

$$y(k,l) = w(k)\sum_{m=1}^{M} u(m,l)\cos\frac{\pi(2m-1)(k-1)}{2M}, \quad k = 1,\ldots,M$$

where

$$w(k) = \begin{cases} \dfrac{1}{\sqrt{M}}, & k = 1 \\ \sqrt{\dfrac{2}{M}}, & 2 \le k \le M \end{cases}$$

The output is always sample based, and the output port rate and data type (real/complex) are the same as those of the input port.

For convenience, length- M 1-D vector inputs and sample-based length- M row vector inputs are processed as single channels (that is, as M-by-1 column vectors), and the output has the same dimension as the input.

The Sine and cosine computation parameter determines how the block computes the necessary sine and cosine values. This parameter has two settings, each with its advantages and disadvantages, as described in the following table.

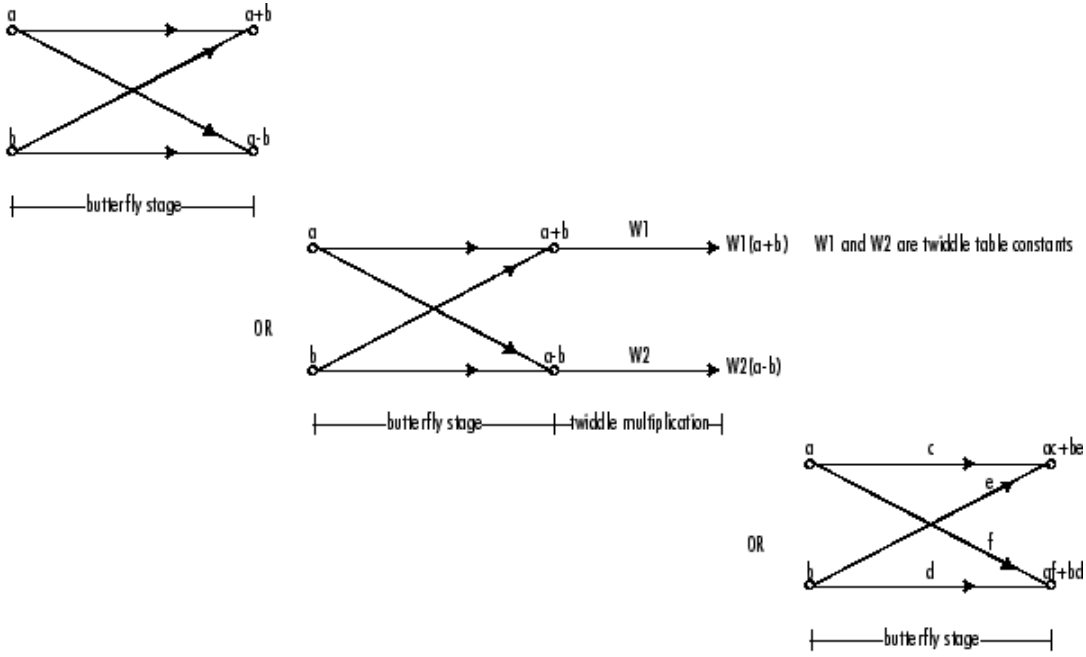
Sine and Cosine Computation Parameter Setting	Sine and Cosine Computation Method	Effect on Block Performance
Table lookup	The block computes and stores the trigonometric values before the simulation starts, and retrieves them during the simulation. When you generate code from the block, the processor running the generated code stores the trigonometric values computed by the block in a speed-optimized table, and retrieves the values during code execution.	The block usually runs much more quickly, but requires extra memory for storing the precomputed trigonometric values.
Trigonometric fcn	The block computes sine and cosine values during the simulation. When you generate code from the block, the processor running the generated code computes the sine and cosine values while the code runs.	The block usually runs more slowly, but does not need extra data memory. For code generation, the block requires a support library to emulate the trigonometric functions, increasing the size of the generated code.

This block supports Simulink virtual buses.

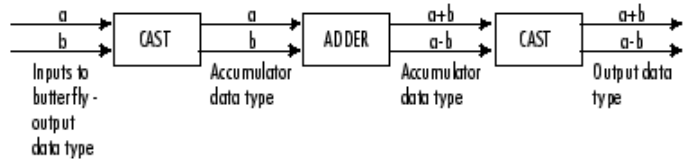
Fixed-Point Data Types

The following diagrams show the data types used within the DCT block for fixed-point signals. You can set the sine table, accumulator, product output, and output data types displayed in the diagrams in the DCT block dialog as discussed in “Dialog Box” on page 2-293.

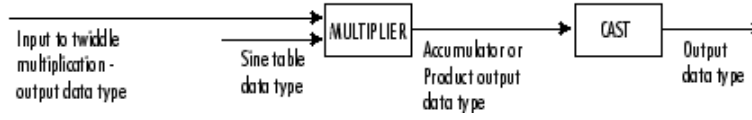
Inputs to the DCT block are first cast to the output data type and stored in the output buffer. Each butterfly stage processes signals in the accumulator data type, with the final output of the butterfly being cast back into the output data type.



Butterfly Stage Data Types



Twiddle Multiplication Data Types

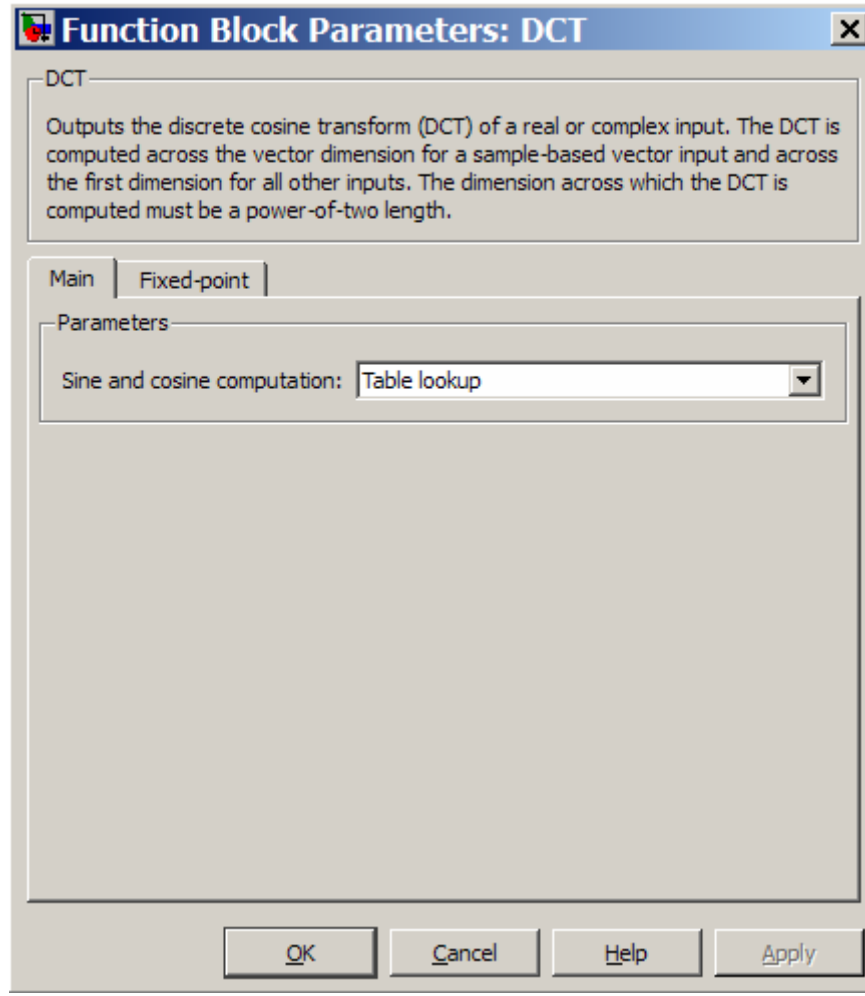


The output of the multiplier is in the product output data type when at least one of the inputs to the multiplier is real. When both of the

inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see “Multiplication Data Types”.

Dialog Box

The Main pane of the DCT block dialog appears as follows.

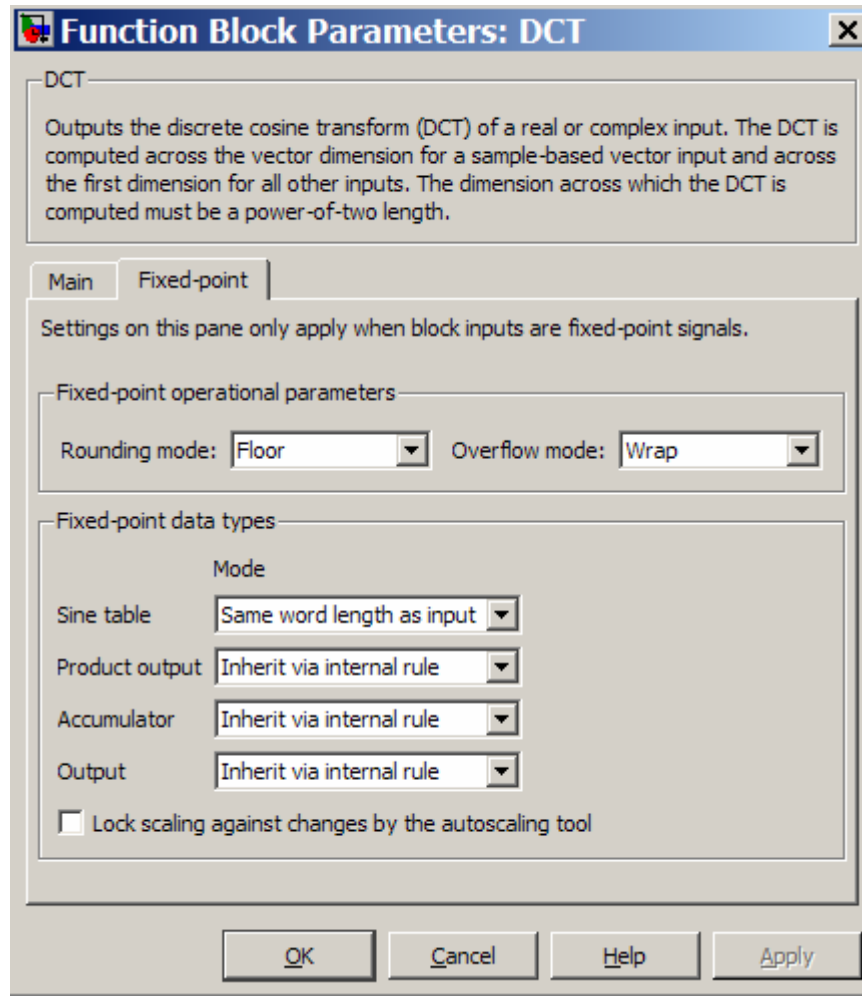


Sine and cosine computation

Sets the block to compute sines an d cosines by either looking up sine and cosine values in a speed-optimized table (Table lookup),

or by making sine and cosine function calls (Trigonometric fcn). See the previous table.

The Fixed-point pane of the DCT block dialog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations. The sine table values do not obey this parameter; they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations.

Sine table

Choose how you specify the word length of the values of the sine table. The fraction length of the sine table values is always equal to the word length minus one:

- When you select Same word length as input, the word length of the sine table values match that of the input to the block.
- When you select Specify word length, you can enter the word length of the sine table values, in bits.

The sine table values do not obey the Rounding mode and Overflow mode parameters; they are always saturated and rounded to Nearest.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See "Fixed-Point Data Types" on page 2-290 and "Multiplication Data Types" for illustrations depicting the use of the product output data type in this block:

- When you select Inherit via internal rule, the product output word length and fraction length are calculated automatically. For information on how the product output word and fraction lengths are calculated when an internal rule is used, see "Inherit via Internal Rule".
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.

- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths. See “Fixed-Point Data Types” on page 2-290 and “Multiplication Data Types” for illustrations depicting the use of the accumulator data type in this block:

- When you select Inherit via internal rule, the accumulator word length and fraction length are calculated automatically. For information about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as product output, these characteristics match those of the product output.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Inherit via internal rule, the output word length and fraction length are calculated automatically. The internal rule first calculates an ideal output word length and fraction length using the following equations:

$$WL_{ideal\ output} = WL_{input} + floor(\log_2(DCT\ length - 1)) + 1$$

$$FL_{ideal\ output} = FL_{input}$$

Using these ideal results, the internal rule then selects word lengths and fraction lengths that are appropriate for your hardware. For more information, see “Inherit via Internal Rule”.

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Complex Cepstrum	Signal Processing Blockset
FFT	Signal Processing Blockset
IDCT	Signal Processing Blockset
Real Cepstrum	Signal Processing Blockset
dct	Signal Processing Toolbox

Purpose

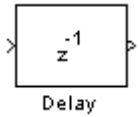
Delay discrete-time input by specified number of samples or frames

Library

Signal Operations

dspsigops

Description



The Delay block delays a discrete-time input by the number of samples or frames specified in the Delay units and Delay parameters. The Delay value must be an integer value greater than or equal to zero. When you enter a value of zero for the Delay parameter, any initial conditions you might have entered have no effect on the output.

The Delay block allows you to set the initial conditions of the signal that is being delayed. The initial conditions must be numeric.

This block reference contains the following topics:

- "Sample-Based Operation" on page 2-299 — Use the Delay block with a sample-based input signal
- "Frame-Based Operation" on page 2-300 — Use the Delay block with a frame-based input signal

Sample-Based Operation

When the input is a sample-based N-D array, each sample of the input is treated as an independent channel. Thus, the total number of channels is equal to the product of the input dimensions. The dimension of the output is the same as that of the input.

When the input is a sample-based N-D array, the Delay parameter can be a scalar integer by which to equally delay all channels or an N-D array of the same dimensions as the input array, containing nonnegative integers that specify the number of sample intervals to delay each channel of the input.

There are four different choices for initial conditions. The initial conditions can be the same or different for each channel. They can also be the same or different within a channel. See the "Sample-Based Operation Examples" on page 2-303 section for more information.

Frame-Based Operation

When the input is a frame-based M-by-N matrix, the block treats each of the N columns as an independent channel, and delays each channel as specified by the Delay parameter.

When the input is frame based, the Delay parameter can be a scalar integer by which to equally delay all channels or a vector whose length is equal to the number of channels.

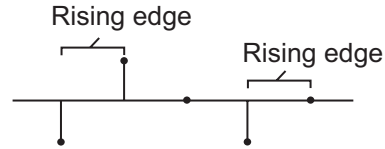
There are four different choices for initial conditions. The initial conditions can be the same or different for each channel. They can also be constant or varying along each channel. See the “Frame-Based Operation Examples” on page 2-308 section for more information.

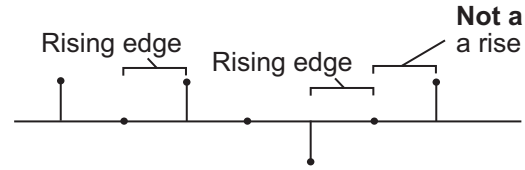
Resetting the Delay

The Delay block resets the delay whenever it detects a reset event at the optional Rst port. The reset sample time must be a positive integer multiple of the input sample time.

The reset event is specified by the Reset port parameter, and can be one of the following:

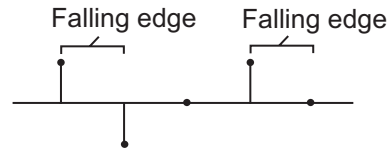
- None disables the Rst port.
- Rising edge triggers a reset operation when the Rst input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following figure)

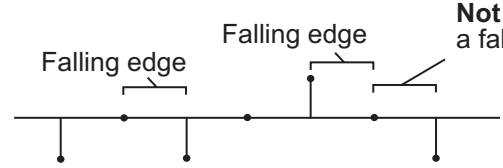




Not a rising edge because it continues a rise from a negative value to zero.

- Falling edge triggers a reset operation when the Rst input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)





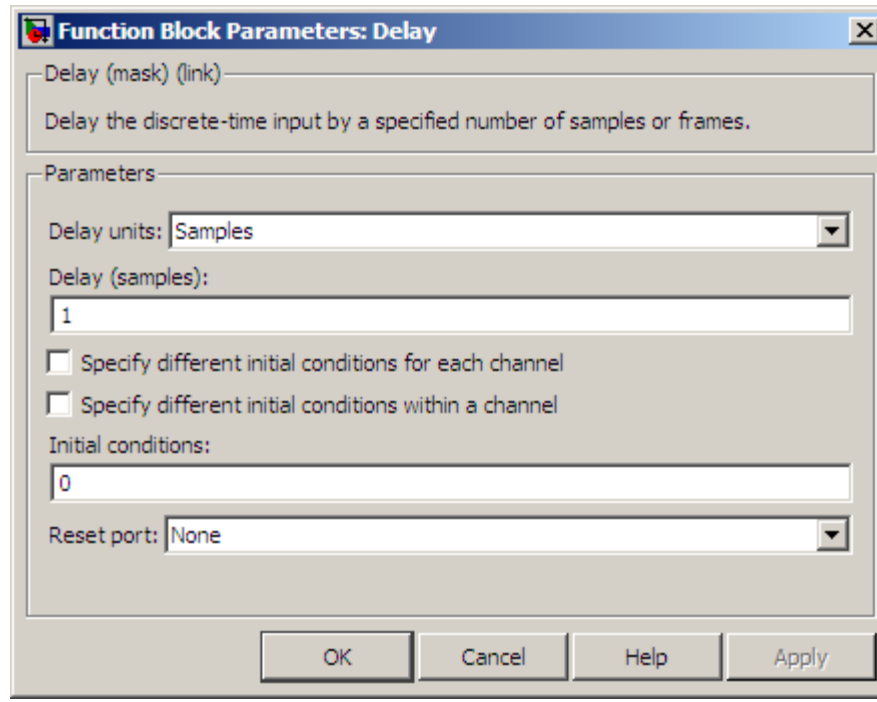
Not a falling edge because it continues a fall from a positive value to zero.

- Either edge triggers a reset operation when the Rst input is Rising edge or Falling edge (as described earlier).
- Non-zero sample triggers a reset operation at each sample time that the Rst input is not zero.

Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample lat ency. Therefore, when the block detects a reset event, there is a one -sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see “Ex cess Algorithmic Delay (Tasking Latency)” and “Models with Multiple Sample Rates” in the Real-Time Workshop User’s Guide.

This block supports Simulink virtual buses.

Dialog Box



Delay units

Select whether you want to delay your input by a specified number of Samples or Frames. You can choose to delay your signal by a certain number of samples or frames, regardless of whether your input is sample or frame based.

Delay (samples) or Delay (frames)

See "Sample-Based Operation" on page 2-299 and "Frame-Based Operation" on page 2-300 for a description of what format to use for each configuration of the block dialog.

Specify different initial conditions for each channel

Select this check box when you want the initial conditions to vary across the channels. When you do not select this check box, the initial conditions are the same across the channels.

Specify different initial conditions within a channel

Select this check box when you want the initial conditions to vary within the channels. When you do not select this check box, the initial conditions are the same within the channels.

Initial conditions

Enter a scalar, vector, matrix, or cell array of initial condition values, depending on your choice for the Specify different initial conditions for each channel and Specify different initial conditions within a channel check boxes. See "Sample-Based Operation" on page 2-299 and "Frame-Based Operation" on page 2-300 for a description of what format to use for each configuration of the block dialog.

Reset port

Determines the reset event that causes the block to reset the delay. For more information, see "Resetting the Delay" on page 2-300.

Examples

Sample-Based Operation Examples

There are four different choices for initial conditions. The initial conditions can be the same or different for each channel. They can also

be the same or different along each channel. The next sections describe the behavior of the block for each of these four cases:

- "Case 1 — Use the Same Initial Conditions for Each Channel and Within a Channe l" on page 2-304
- "Case 2 — Use Dif ferent Initial Conditions for Each Channel and the Same Initial Co nditions Within a Channel" on page 2-305
- "Case 3 — Use the Same Initial Conditions for Each Channel and Different Init ial Conditions Within a Channel" on page 2-306
- "Case 4 — Use Dif ferent Initial Conditions for Each Channel and Within a Chann el" on page 2-307

Case 1 — Use the Same Initial Conditions for Each Channel and Within a Channel

Enter a scalar value for the initial conditions. This value is used as the constant initial condition value for each of the channels.

For example, suppose your inpu t is a sample-based matrix.

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}, \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}, \ldots$$

You want the i nitial conditions of your four-channel signal to be identical an d zero for the Þrst two samples:

1 For the Delay (samples) parameter, type 2.

2 Clear the Specify different initial conditions for each channel and Specify different initial conditions within a channel check boxes.

3 For the Initial conditions parameter, specify a scalar value of 0.

The output o f the delay block is

$$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix},\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix},\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix},\begin{bmatrix}2 & 2\\2 & 2\end{bmatrix},\begin{bmatrix}3 & 3\\3 & 3\end{bmatrix},\ldots$$

`0`, the scalar initial condition value, is used for each channel and within the channels. It is the output at sample time zero and sample time one.

Case 2 — Use Different Initial Conditions for Each Channel and the Same Initial Conditions Within a Channel

The initial conditions must be an N-D array for N-D input. The initial conditions must have the same dimensions as the input data. These initial condition values are used as the constant initial condition value for each of the channels.

For example, suppose your input is a sample-based matrix.

$$\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix},\begin{bmatrix}2 & 2\\2 & 2\end{bmatrix},\begin{bmatrix}3 & 3\\3 & 3\end{bmatrix},\ldots$$

You want the initial conditions of your four-channel signal to be

$$\begin{bmatrix}7 & 9\\11 & 13\end{bmatrix}$$

for the first two samples:

1. For the **Delay (samples)** parameter, type `2`.
2. Select the **Specify different initial conditions for each channel** check box.
3. Clear the **Specify different initial conditions within a channel** check box.
4. For the **Initial conditions** parameter, type `[7 9; 11 13]`.

   The output of the delay block is

$$\begin{bmatrix} 7 & 9 \\ 11 & 13 \end{bmatrix}, \begin{bmatrix} 7 & 9 \\ 11 & 13 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}, \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}, \ldots$$

The initial condition matrix is the output at sample time zero and sample time one. Different initial conditions are used for each channel; the same initial condition value is used within a channel.

Case 3 — Use the Same Initial Conditions for Each Channel and Different Initial Conditions Within a Channel

In this case, for N-D sample-based inputs, the initial conditions parameter must be a vector whose length is equal to the delay value, specified by the Delay parameter. The values in this vector are used as the initial condition values along each of the channels to be delayed.

For example, suppose your input is a sample-based matrix.

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}, \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}, \ldots$$

You want the initial conditions of your four channel signal to be the same along each of the channels to be delayed:

1. For the Delay (samples) parameter, type `2`.
2. Clear the Specify different initial conditions for each channel check box.
3. Select the Specify different initial conditions within a channel check box.
4. For the Initial conditions parameter, type `[10 20]`.

   The output of the delay block is

$$\begin{bmatrix} 10 & 10 \\ 10 & 10 \end{bmatrix}, \begin{bmatrix} 20 & 20 \\ 20 & 20 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}, \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}, \ldots$$

The first element of the initial conditions vector is the output, for all channels, at sample time zero. The second element of the initial conditions vector is the output, for all channels, at sample time one. The same initial conditions are used for each channel, but different initial condition values are used within a channel.

Case 4 — Use Different Initial Conditions for Each Channel and Within a Channel

Enter a cell array for your initial condition values. The cell array must be the same size as your input signal. Each cell of the cell array represents the delay values for one channel, and must be a vector of size equal to the delay value. If you have a vector or scalar input and a scalar delay value, you can enter the initial conditions as a matrix.

For example, suppose your input is a sample-based vector.

$$[1 \quad 1],\ [2 \quad 2],\ [3 \quad 3],\ldots$$

You want the initial conditions of your two channel signal to be different for each channel and along each channel:

1. For the **Delay (samples)** parameter, type `2`.
2. Select the **Specify different initial conditions for each channel** and **Specify different initial conditions within a channel** check boxes.
3. For the **Initial conditions** parameter, type `[10 20; 30 40]`

   The output of the delay block is

   $$[10 \quad 20],\ [30 \quad 40],\ [1 \quad 1],\ [2 \quad 2]\ldots$$

   The first row of the initial conditions vector is the output at sample time zero. The second row of the initial conditions vector is the output at sample time one. Different initial conditions are used for each channel and within the channels.

In addition, suppose your input is a sample-based matrix.

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}, \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}, \ldots$$

You want the initial conditions of your two-channel signal to be different for each channel and along each channel:

1 For the Delay (samples) parameter, type 2.

2 Select the Specify different initial conditions for each channel and the Specify different initial conditions within a channel check boxes.

3 For the Initial conditions parameter, type {[11 15] [12 16]; [13 17] [14 18]}. The dimensions of the cell array match the dimensions of the input. Also, each element of the cell array represents the initial conditions within one channel.

The output of the delay block is

$$\begin{bmatrix} 11 & 12 \\ 13 & 14 \end{bmatrix}, \begin{bmatrix} 15 & 16 \\ 17 & 18 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}, \ldots$$

Each element of the cell array represents the initial conditions within a channel. The first element, a vector, represents the initial conditions within channel 1. The second element, a vector, represents the initial conditions within channel 2, and so on. Different initial conditions are used for each channel and within the channels.

Frame-Based Operation Examples

There are four different choices for initial conditions. The initial conditions can be the same or different for each channel. They can also be constant or varying along each channel. The next sections describe the behavior of the block for each of these four cases:

- “Case 1 — Use the Same Initial Conditions for Each Channel and Within a Channel” on page 2-309
- “Case 2 — Use Different Initial Conditions for Each Channel and the Same Initial Conditions Within a Channel” on page 2-310
- “Case 3 — Use the Same Initial Conditions for Each Channel and Different Initial Conditions Within a Channel” on page 2-311
- “Case 4 — Use Different Initial Conditions for Each Channel and Within a Channel” on page 2-312

Case 1 — Use the Same Initial Conditions for Each Channel and Within a Channel

Enter a scalar value for the initial conditions. This value is used as the constant initial condition value for each of the channels.

For example, suppose your input is a frame-based matrix.

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 4 & 4 \\ 5 & 5 & 5 \\ 6 & 6 & 6 \end{bmatrix}, \begin{bmatrix} 7 & 7 & 7 \\ 8 & 8 & 8 \\ 9 & 9 & 9 \end{bmatrix}, \ldots$$

You want the initial conditions of your three-channel signal to be identical and zero for the first frame:

1 For the Delay (frames) parameter, type `1`.

2 Clear the Specify different initial conditions for each channel and the Specify different initial conditions within a channel check boxes.

3 For the Initial conditions parameter, specify a scalar value of `0`.

The output of the delay block is

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 4 & 4 \\ 5 & 5 & 5 \\ 6 & 6 & 6 \end{bmatrix}, \begin{bmatrix} 7 & 7 & 7 \\ 8 & 8 & 8 \\ 9 & 9 & 9 \end{bmatrix}, \ldots$$

0, the scalar initial condition value, is used across the channels and within the channels for the first frame. This frame is the output at sample time zero.

Case 2 — Use Different Initial Conditions for Each Channel and the Same Initial Conditions Within a Channel

The initial conditions must be a vector of length N, where $N \geq 1$. N is also equal to the number of channels in your signal. These initial condition values are used as the constant initial condition value for each of the channels.

For example, suppose your input is a frame-based matrix.

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 4 & 4 \\ 5 & 5 & 5 \\ 6 & 6 & 6 \end{bmatrix}, \begin{bmatrix} 7 & 7 & 7 \\ 8 & 8 & 8 \\ 9 & 9 & 9 \end{bmatrix}, \ldots$$

You want the initial conditions of your three-channel signal to be [0 10 20] for the first frame:

1. For the Delay (frames) parameter, type 1.
2. Select the Specify different initial conditions for each channel check box.
3. Clear the Specify different initial conditions within a channel check box.
4. For the Initial conditions parameter, type [0 10 20] .

   The output of the delay block is

$$\begin{bmatrix} 0 & 10 & 20 \\ 0 & 10 & 20 \\ 0 & 10 & 20 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 4 & 4 \\ 5 & 5 & 5 \\ 6 & 6 & 6 \end{bmatrix}, \begin{bmatrix} 7 & 7 & 7 \\ 8 & 8 & 8 \\ 9 & 9 & 9 \end{bmatrix}, \ldots$$

The initial condition vector expands to create the frame that is output at sample time zero. Different init ial conditions are used for each channel, but the same initial condition value is used with a channel.

Case 3 — Use the Same Initial Conditions for Each Channel and Different Initial Conditions Within a Channel

In this case, the Delay parameter can be a scalar integer by which to equally delay all channels or a vector whose length is equal to the number of channels. All the values of this vector must be equal.

Enter the initial conditions as a vector. These values are used as the initial condition value along each of the channels to be delayed. The initial condition vector must have length equal to the value of the Delay (frames) parameter multiplied by the frame length. For example, if you want to delay your signal by two frames with frame length two and an initial condition value of 3, enter your initial condition vector as [3 3 3 3] .

For example, suppose your input is a frame-based matrix.

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 4 & 4 \\ 5 & 5 & 5 \\ 6 & 6 & 6 \end{bmatrix}, \begin{bmatrix} 7 & 7 & 7 \\ 8 & 8 & 8 \\ 9 & 9 & 9 \end{bmatrix}, \ldots$$

You want the initial conditions of your three-channel signal to be the same along each of the channels to be delayed:

1. For the Delay (frame) parameter, type 1.
2. Clear the Specify different initial conditions for each channel check box.

3 Select the Specify different initial conditions within a channel check box.

4 For the Initial conditions parameter, type [10 20 30] .

The output of the delay block is

$$\begin{bmatrix} 10 & 10 & 10 \\ 20 & 20 & 20 \\ 30 & 30 & 30 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 4 & 4 \\ 5 & 5 & 5 \\ 6 & 6 & 6 \end{bmatrix}, \begin{bmatrix} 7 & 7 & 7 \\ 8 & 8 & 8 \\ 9 & 9 & 9 \end{bmatrix}, \ldots$$

The initial condition vector de Þnes the initial condition values within each of the three channels. The same initial conditions are used for each channel, but different initial condition values are used with a channel.

Case 4 — Use Different Initial C onditions for Each Channel and Within a Channel

Enter a cell array for your initial c ondition values. Or, when you have a scalar delay value, you can enter the initial conditions as a matrix.

For example, suppose your input is a frame-based matrix.

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 4 & 4 \\ 5 & 5 & 5 \\ 6 & 6 & 6 \end{bmatrix}, \begin{bmatrix} 7 & 7 & 7 \\ 8 & 8 & 8 \\ 9 & 9 & 9 \end{bmatrix}, \ldots$$

You want the initial conditions of your three-channel signal to be different for each channel and along each channel.

1 For the Delay (frames) parameter, type 1.

2 Select the Specify different initial conditions for each channel and the Specify different initial conditions within a channel check boxes.

3 For the Initial conditions parameter, type either [10 20 30; 40 50 60; 70 80 90] or {[10 40 70];[20 50 80];[30 60 90]}. Each cell of the cell array represents the delay along one channel.

Regardless of whether you use a matrix or cell array, the output of the delay block is

$$\begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 4 & 4 \\ 5 & 5 & 5 \\ 6 & 6 & 6 \end{bmatrix}, \begin{bmatrix} 7 & 7 & 7 \\ 8 & 8 & 8 \\ 9 & 9 & 9 \end{bmatrix} \ldots$$

The initial condition matrix is the output at sample time zero. The elements of the initial condition cell array define the initial condition values within each channel. The first element, a vector, represents the initial conditions within channel 1. The second element, a vector, represents the initial conditions within channel 2, and so on. Different initial conditions are used for each channel and within the channels.

Supported Data Types

- Double-precision floating point
- Single-precision floating point
- Fixed point (signed and unsigned)
- Boolean
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

See Also

Unit Delay	Simulink
Variable Fractional Delay	Signal Processing Blockset
Variable Integer Delay	Signal Processing Blockset

Delay Line

Purpose

Rebuffer sequence of inputs with one-sample shift

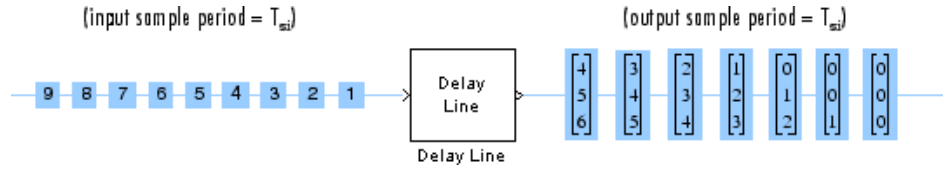
Library

Signal Management / Buffers

dspbuff3

Description

The Delay Line block buffers the input samples into a sequence of overlapping or underlapping matrix outputs. In the most typical use (sample-based inputs), each output differs from the preceding output by only one sample, as illustrated below for scalar input.



Note that the first output of the block in the example above is all zeros; this is because the Initial Conditions parameter is set to zero. Due to the latency of the Delay Line block, all outputs are delayed by one frame, the entries of which are defined by the Initial Conditions parameter.

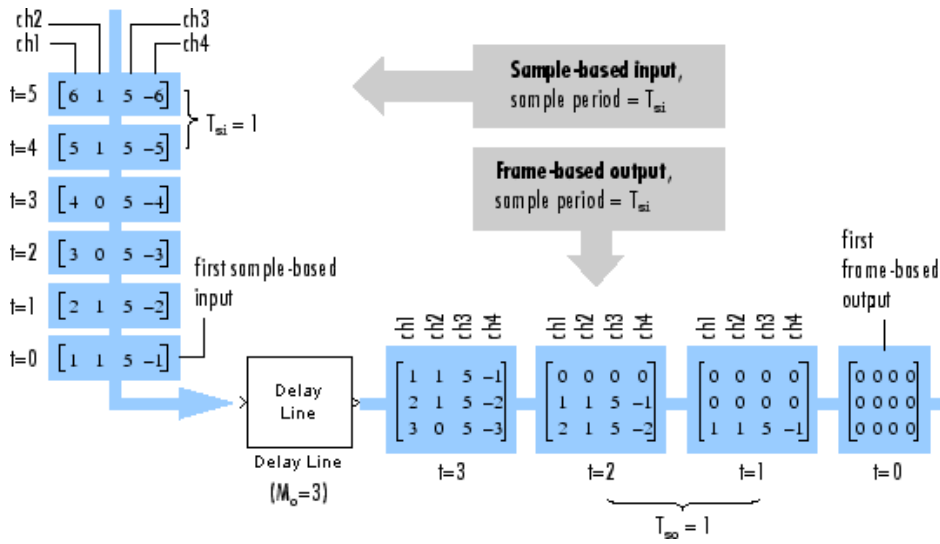
Sample-Based Operation

In sample-based operation, the Delay Line block buffers a sequence of sample-based length- N vector inputs (1-D, row, or column) into a sequence of overlapping frame-based M_o-by-N matrix outputs, where M_o is specified by the Delay line size parameter (M_o>1). That is, each input vector becomes a row in the frame-based output matrix.

At each sample time the new input vector is added in the last row of the output, so each output overlaps the previous output by M_o-1 samples. Therefore, the output sample period and frame period is the same as the input sample period ($T_{so}=T_{si}$, and $T_{fo}=T_{si}$). When $M_o=1$, the input is simply passed through to the output and retains the same dimension, but becomes frame based. The latency of the block always causes an initial delay in the output; the value of the first output is specified by the Initial conditions parameter (see “Initial Conditions” on page 2-317). Sample-based full-dimensional matrix inputs are not accepted.

The Delay Line block's sample-based operation is similar to that of a Buffer block with Buffer size equal to M_o and Buffer overlap equal to M_o-1, except that the Buffer block has a different latency.

In the following model, the block operates on a sample-based input with a Delay line size of 3.



The input vectors in the example above do not begin appearing at the output until the second row of the second matrix due to the block's latency (see "Initial Conditions" on page 2-317). The first output matrix (all zeros in this example) reflects the block's Initial conditions setting. As for any sample-based input, the output frame rate and output sample rate are both equal to the input sample rate.

Frame-Based Operation

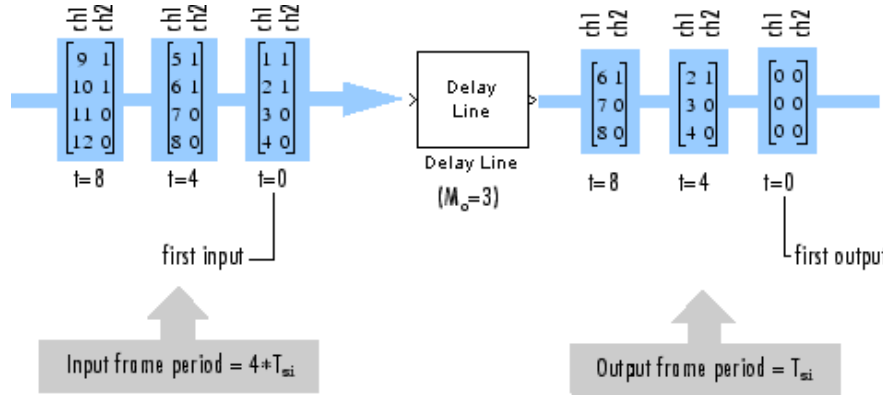
In frame-based operation, the Delay Line block rebuffers a sequence of frame-based M_i-by-N matrix inputs into a sequence of frame-based M_o-by-N matrix outputs, where M_o is the output frame size specified by the Delay line size parameter. Depending on whether M_o is greater than, less than, or equal to the input frame size, M_i, the output frames

can be underlapped or overlapped. Each of the N input channels is rebuffered independently.

When $M_o > M_i$, the output frame overlap is the difference between the output and input frame size, $M_o - M_i$. When $M_o < M_i$, the output is underlapped; the Delay Line block discards the first $M_i - M_o$ samples of each input frame so that only the last M_o samples are buffered into the corresponding output frame. When $M_o = M_i$, the output data is identical to the input data, but is delayed by the latency of the block. Due to the block's latency, the outputs are always delayed by one frame, the entries of which are specified by the Initial conditions (see "Initial Conditions" on page 2-317).

The output frame period is equal to the input frame period ($T_{fo}=T_{fi}$). The output sample period, T_{so}, is therefore equal to T_{fi}/M_o, or equivalently, $T_{si}(M_i/M_o)$

In the following model, the block rebuffers a two-channel frame-based input with a Delay line size of 3.



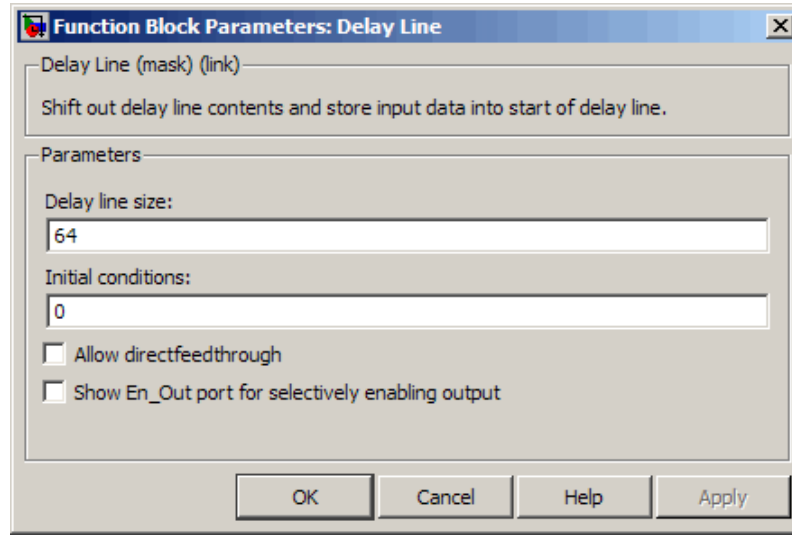
The first output frame in the example is a product of the latency of the Delay Line block; it is all zeros because the Initial conditions is set to be zero. Since the input frame size, 4, is larger than the output frame size, 3, only the last three samples in each input frame are propagated to the corresponding output frame. The frame periods of the input and

output are the same, and the output sample period is $T_{si}(M_i/M_o)$, or 4/3 the input sample period.

Initial Conditions

The Delay Line block's buffer is initialized to the value specified by the Initial condition parameter. The block outputs this buffer at the first simulation step (t=0). When the block's output is a vector, the Initial condition can be a vector of the same size, or a scalar value to be repeated across all elements of the initial output. When the block's output is a matrix, the Initial condition can be a matrix of the same size, a vector (of length equal to the number of matrix rows) to be repeated across all columns of the initial output, or a scalar to be repeated across all elements of the initial output.

Dialog Box



Delay line size
The number of rows in output matrix, M_o.

Initial conditions
The value of the block's initial output, a scalar, vector, or matrix.

Allow direct feedthrough

When you select this check box, the input data is not delayed by an extra frame before it is available at the output buffer. Instead, the input data is available immediately at the output port of the block.

Show En_Out port for selectively enabling output

When you select this check box, the En_Out port appears on the block icon. This block uses a circular buffer internally even though the output is linear. This means that for valid output, data from the circular buffer has to be linearized. The En_Out port determines whether or not a valid output needs to be computed based on the value of its Boolean input. If the input value to the En_Out port is 1, the block output is linearized, and thus is valid. Otherwise, the output is not linearized, and is invalid. This allows the block to be more efficient when the tapped Delay Line's output is not required at each sample time.

Note that when the input value to the En_Out port is 0, the block can give different results depending on the state of the model. The results can appear to match valid results or can be invalid, and they cannot be predicted. You should ignore the block output in all cases when the input to the En_Out port is 0.

Hold previous value when the output is disabled

This parameter only appears and applies when the Show En_Out port for selectively enabling output parameter is selected. Use this parameter to specify the block output at those time steps when the internal state buffer is not being linearized to output valid data.

When you do not select this check box, the block memory is free to be used by other parts of the model, and the signal on the output port is invalid. When you select this check box, the most recent valid value is held on the output port, and slightly more memory is used by the block.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Buffer	Signal Processing Blockset
Triggered Delay Line (Obsolete)	Signal Processing Blockset

Detrend

Purpose

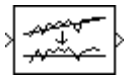
Remove linear trend from vectors

Library

Statistics

dspstat3

Description



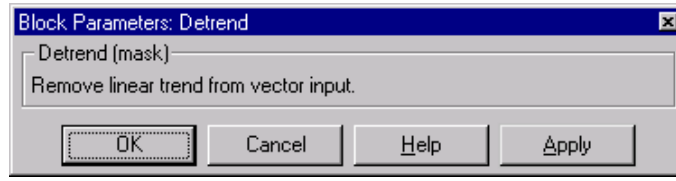
The Detrend block removes a linear trend from the length- M input vector, u, by subtracting the straight line that best fits the data in the least squares sense.

The least squares line, û = ax + b, is the line with parameters a and b that minimizes the quantity

$$\sum_{i=1}^{M}(u_i - u_i)^2$$

for M evenly-spaced values of x, where u_i is the ith element in the input vector. The output, y = u-û, is an M-by-1 column vector (regardless of the input vector dimension) with the same frame status as the input.

Dialog Box



Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

Cumulative Sum	Signal Processing Blockset
Difference	Signal Processing Blockset

Least Squares Polynomial Fit	Signal Processing Blockset
Unwrap	Signal Processing Blockset
detrend	MATLAB

Difference

Purpose

Compute element-to-element difference along specified dimension of input

Library

Math Functions / Math Operations

dspmathops

Description

The Difference block computes the difference between adjacent elements in rows, columns, or a specified dimension of the input array u. This block accepts real and complex fixed-point and floating-point inputs, except for complex unsigned fixed-point inputs.

Columnwise Differencing

When the Difference along parameter is set to Columns, the block computes differences between adjacent elements along each column.

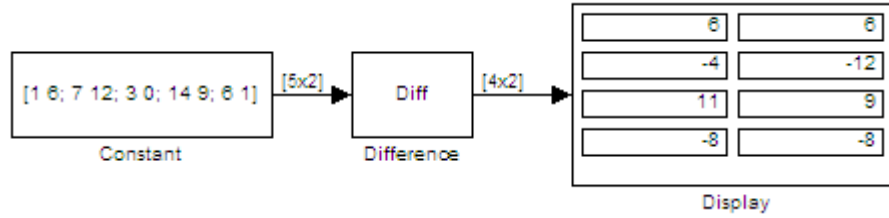
```
y = diff(u)          % Equivalent MATLAB code
```

For sample-based inputs, the output is a sample-based (M-1)-by-N matrix whose jth column has elements

$$y_{i,j} = u_{i+1,j} - u_{i,j} \quad 1 \le i \le (M-1)$$

For convenience, length-M 1-D vector inputs are treated as M-by-1 column vectors for columnwise differencing, and the output is 1-D.

For example, the following figure shows the block output for sample-based inputs:



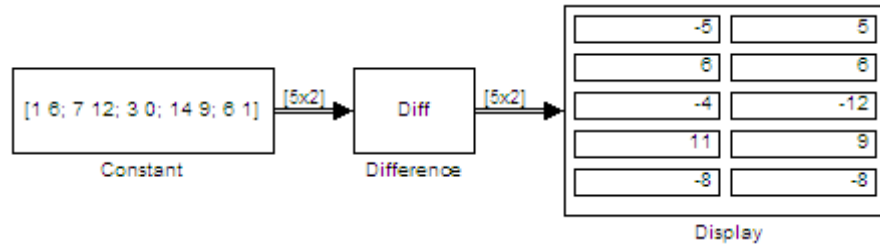
For frame-based inputs, the output is a frame-based M-by-N matrix whose jth column has elements

$$y_{i,j} = u_{i+1,j} - u_{i,j} \quad 2 \le i \le (M-1)$$

The first element of the output for each column is the first input element minus the last input element of the previous frame. For the first frame, zero is subtracted from the first input element.

$$y_{1,j}(t) = u_{1,j}(t) - u_{M,j}(t - T_f)$$

For example, the following figure shows the second frame of the block output for a frame-based input:



Rowwise Differencing

When the Difference along parameter is set to Rows, the block computes differences between adjacent elements along each row. The result is the same regardless of the frame status of the input signal.

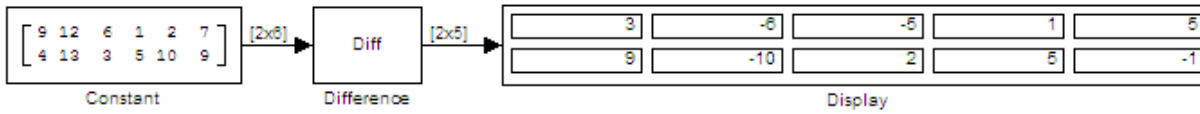
```
y = diff(u,[],2)          % Equivalent MATLAB code
```

The output is an M-by-(N-1) matrix whose ith row has elements

$$y_{i,j} = u_{i,j+1} - u_{i,j} \quad 1 \le j \le (N-1)$$

The frame status of the output is the same as the input. For convenience, length-N 1-D vector inputs are treated as 1-by-N row vectors for rowwise differencing, and the output is 1-D.

For example, the following figure shows the block output for sample-based inputs. The output is the same for frame-based inputs:



Differencing Along Arbitrary Dimensions

When the Difference along parameter is set to Specified dimension, the behavior of the block is an extension of the rowwise differencing described earlier. The block computes differences between adjacent elements along the dimension specified by the Dimension parameter.

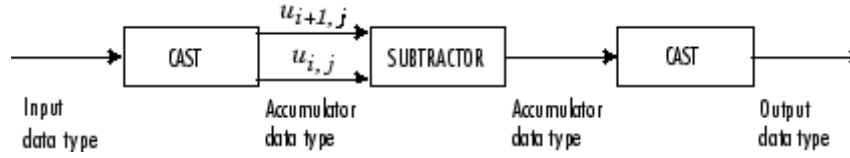
```
y = diff(u,[],d) % Equivalent MATLAB code where d is the dimension
```

The output is an array whose length in the specified dimension is one less than that of the input, and whose lengths in other dimensions are unchanged. For example, consider an M-by-N-by-P-by-R input array with elements u(i,j,k,l) and assume that Dimension is 3. The output of the block is an M-by-N-by-(P–1)-by-R array with elements

$$y_{i,j,k,l} = u_{i,j,k+1,l} - u_{i,j,k,l} \quad 1 \le k \le (P-1)$$

Fixed-Point Data Types

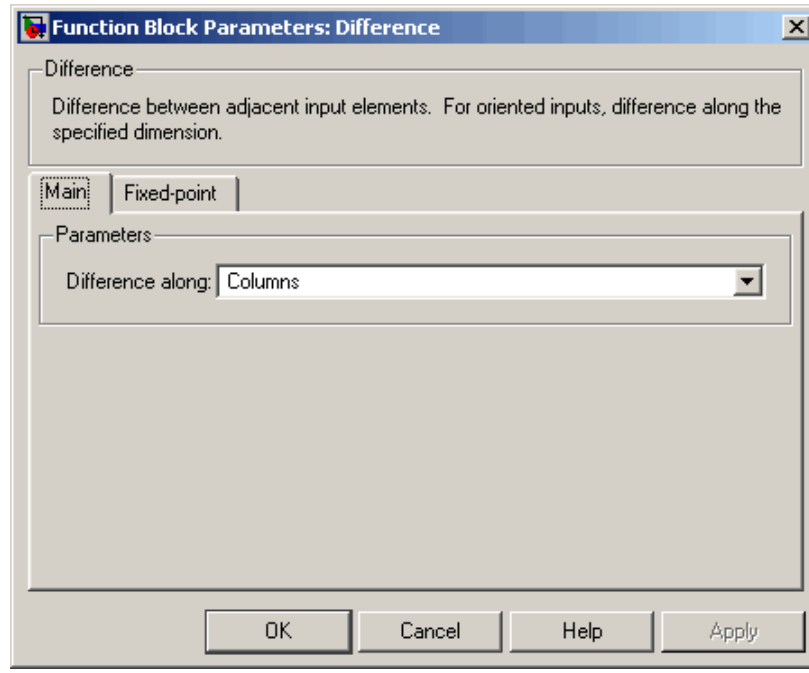
The following diagram shows the data types used within the Difference block for fixed-point signals.



You can set the accumulator and output data types in the block dialog as discussed in “Dialog Box” on page 2-325.

Dialog Box

The Main pane of the Difference block appears as follows.



Difference along

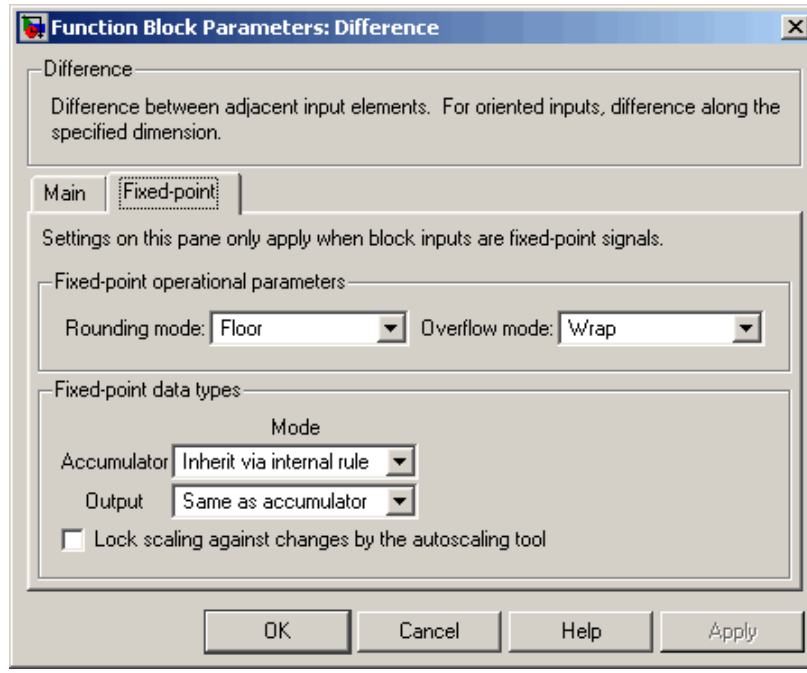
Specify whether the block performs columnwise differencing, rowwise differencing, or differencing along a specified dimension.

Dimension

Specify the one-based dimension along which to compute element-to-element differences.

This parameter is only visible when you select Specified dimension for the Difference along parameter.

The Fixed-point pane of the Difference block appears as follows.



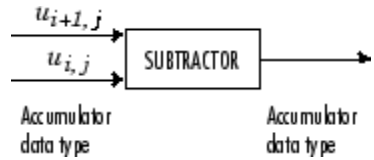
Roundin g mode

Select th e rounding mode for Þxed-point operations.

Over flow m ode

Select th e overßow mode for Þxed-point operations.

Accumula tor



Use this parameter to specify how you would like to designate the accumulator word and fraction lengths:

- When you select Inherit via internal rule , the accumulator word length and fraction length are calculated automatically. For informati on about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and fraction lengt h of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as accumulator , these characteristics match those of the accumulator.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any Þxed-point scaling you specify in this block mask f rom being overridden by the autoscaling tool in the Fixed-Point Tool.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Cumulative Sum — Signal Processing Blockset

diff — MATLAB

Purpose

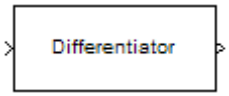
Design differentiator filter

Library

Filtering / Filter Design Toolbox

dspfdesign

Description



This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See “Differentiator Filter Design Dialog Box — Main Pane” in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset product.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point

Port	Supported Data Types
	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Purpose

Filter each channel of input over time using static or time-varying digital filter implementations

Library

Filtering / Filter Designs

dsparch4

Description

Note Use this block to efficiently implement a floating-point or fixed-point filter for which you know the coefficients, or that is already defined in a Signal Processing Toolbox dfilt object or a Filter Design Toolbox dfilt object. The following Signal Processing Blockset blocks also implement digital filters, but serve slightly different purposes:

- Digital Filter Design — Use to design, analyze, and then efficiently implement floating-point filters. This block provides the same filter implementation as the Digital Filter block for floating-point signals.
- Filter Realization Wizard — Use to implement floating-point or fixed-point filters built from Sum, Gain, and Unit Delay blocks. You can either design the filter using block filter design and analysis parameters, or import the coefficients of a filter that you designed elsewhere.

The Digital Filter block independently filters each channel of the input signal with a specified digital IIR or FIR filter. The block can implement static *fil*ters with fixed coefficients, as well as time-varying *fil*ters with coefficients that change over time. You can tune the coefficients of a static filter during simulation.

This block filters each channel of the input signal independently over time. The output frame status and dimensions are always the same as those of the input signal that is filtered. When inputs are frame based, the block treats each column as an independent channel; the block filters each column. When inputs are sample based, the block treats each element of the input as an individual channel.

The outputs of this block numerically match the outputs of the Digital Filter Design block and of the dfilt function in Signal Processing Toolbox software or Filter Design Toolbox software.

Note The Digital Filter block has direct feedthrough, so if you connect the output of this block back to its input you get an algebraic loop. For more information on direct feed through and algebraic loops, see "Algebraic Loops" in the S imulink documentation.

Sections of This Reference Page

- "CoefÞcient Source" on page 2-332
- "Supported Filter Structures" on page 2-333
- "Specifying Initial Conditions" on page 2-335
- "State Logging" on page 2-338
- "Fixed-Point Data Types" on page 2-339
- "Dialog Box" on page 2-339
- "Filter Structure Diagrams" on page 2-354
- "Supported Data Types" on page 2-390
- "See Also" on page 2-390

Coefficient Source

The Digital Filter block can operate in three different modes. Select the mode in the Coef fi cient source group box. If you select

- Dialog parameters , you enter information about the Þlter such as structure and coef Þcients in the block mask.
- Input port(s) , you enter the Þlter structure in the block mask, and the Þlter coef Þcients come in through one or more block ports. This mode is useful for specifying time-varying Þlters.

- Discrete-time filter object (DFILT), you specify the filter using a dfilt object from the Signal Processing Toolbox product or the Filter Design Toolbox product.

Supported Filter Structures

When you select Discrete-time filter object (DFILT), the following dfilt structures are supported:

- dfilt.df1
- dfilt.df1t
- dfilt.df2
- dfilt.df2t
- dfilt.df1sos
- dfilt.df1tsos
- dfilt.df2sos
- dfilt.df2tsos
- dfilt.dffir
- dfilt.dffirt
- dfilt.dfsymfir
- dfilt.dfasymfir
- dfilt.latticear
- dfilt.latticemamin

When you select Dialog parameters or Input port(s), the list of filter structures offered in the Filter structure parameter depends on whether you set the Transfer function type to IIR (poles & zeros), IIR (all poles), or FIR (all zeros), as summarized in the following table.

Note Each structure listed in the table below supports both fixed-point and floating-point signals.

The table also shows the vector or matrix of filter coefficients you must provide for each filter structure. For more information on how to specify filter coefficients for various filter structures, see “Specifying Static Filters” and “Specifying Time-Varying Filters”.

Filter Structures and Filter Coefficients

Transfer Function Type	Supported Filter Structures	Filter Coefficient Specification
IIR (poles & zeros)	Direct form I Direct form I transposed Direct form II Direct form II transposed	• Numerator coefficients vector [b0, b1, b2, ..., bn] • Denominator coefficients vector [a0, a1, a2, ..., am]
	Biquadratic direct form I (SOS) Biquadratic direct form I transposed (SOS) Biquadratic direct form II (SOS) Biquadratic direct form II transposed (SOS)	• M-by-6 second-order section (SOS) matrix. • Scale values See “Specifying the SOS Matrix (Biquadratic Filter Coefficients)”.
IIR (all poles)	Direct form Direct form transposed	Denominator coefficients vector [a0, a1, a2, ..., am]
	Lattice AR	Reflection coefficients vector [k1, k2, ..., kn]

Filter Structures and Filter Coefficients (Continued)

Transfer Function Type	Supported Filter Structures	Filter Coefficient Specification
FIR (all zeros)	Direct form Direct form symmetric Direct form antisymmetric Direct form transposed	Numerator coefficients vector [b0, b1, b2, ..., bn]
	Lattice MA	Reflection coefficients vector [k1, k2, ..., kn]

Specifying Initial Conditions

In Dialog parameters and Input port(s) modes, the block initializes the internal filter states to zero by default, which is equivalent to assuming past inputs and outputs are zero. You can optionally use the Initial conditions parameter to specify nonzero initial conditions for the filter delays.

To determine the number of initial condition values you must specify, and how to specify them, see the following table on Valid Initial Conditions and Number of Delay Elements (Filter States) on page 2-337. The Initial conditions parameter can take one of four forms as described in the following table.

Valid Initial Conditions

Initial Condition	Examples	Description
Scalar	5 Each delay element for each channel is set to 5.	The block initializes all delay elements in the filter to the scalar value.

Valid Initial Conditions (Continued)

Initial Condition	Examples	Description
Vector (for applying the same delay elements to each channel)	For a filter with two delay elements: [d_1 d_2] The delay elements for all channels are d1 and d2.	Each vector element specifies a unique initial condition for a corresponding delay element. The block applies the same vector of initial conditions to each channel of the input signal. The vector length must equal the number of delay elements in the filter (specified in the table Number of Delay Elements (Filter States) on page 2-337).
Vector or matrix (for applying different delay elements to each channel)	For a 3-channel input signal and a filter with two delay elements: [d_1 d_2 D_1 D_2 d_1 d_2] or $\begin{bmatrix} d_1 & D_1 & d_1 \\ d_2 & D_2 & d_2 \end{bmatrix}$ • The delay elements for channel 1 are d_1 and d_2. • The delay elements for channel 2 are D_1 and D_2. • The delay elements for channel 3 are $d_1$and d_2.	Each vector or matrix element specifies a unique initial condition for a corresponding delay element in a corresponding channel: • The vector length must be equal to the product of the number of input channels and the number of delay elements in the filter (specified in the table Number of Delay Elements (Filter States) on page 2-337). • The matrix must have the same number of rows as the number of delay elements in the filter (specified in the table Number of Delay Elements (Filter States) on page 2-337), and must have one column for each channel of the input signal.
Empty matrix	[ ] Each delay element for each channel is set to 0.	The empty matrix, [] , is equivalent to setting the Initial conditions parameter to the scalar value 0.

The number of delay elements (Þlter states) per input channel depends on the Þlter structure, as indicated in the following table.

Number of Delay Elements (Filter States)

Filter Structure	Number of Delay Elements per Channel
Direct form Direct form transposed Direct form symmetric Direct form antisymmetric	#_of_filter_coeffs-1
Direct form I Direct form I transposed	• #_of_zeros-1 • #_of_poles-1
Direct form II Direct form II transposed	max(#_of_zeros, #_of_poles)-1
Biquadratic direct form I (SOS) Biquadratic direct form I transposed (SOS) Biquadratic direct form II (SOS) Biquadratic direct form II transposed (SOS)	2 * #_of_filter_sections
Lattice AR Lattice MA	#_of_reflection_coeffs

State Logging

Simulink enables you to log the states in your model to the MATLAB workspace. The following table indicates which filter structures of the Digital Filter block support the Simulink state logging feature. See “States” in the Simulink User’s Guide documentation for more information.

Transfer Function Type	Filter Structure	State Logging Supported
IIR (poles & zeros)	Direct form I	No
	Direct form I transposed	Yes
	Direct form II	No
	Direct form II transposed	Yes
	Biquadratic direct form I (SOS)	Yes
	Biquadratic direct form I transposed (SOS)	Yes
	Biquadratic direct form II (SOS)	Yes
	Biquadratic direct form II transposed (SOS)	Yes
IIR (all poles)	Direct form	No
	Direct form transposed	Yes
	Lattice AR	Yes
FIR (all zeros)	Direct form	No
	Direct form symmetric	No
	Direct form antisymmetric	No
	Direct form transposed	Yes
	Lattice MA	Yes

Fixed-Point Data Types

All structures supported by the Digital Filter block support fixed-point data types. You can specify intermediate fixed-point data types for quantities such as the coefficients, accumulator, and product output for each filter structure. See "Filter Structure Diagrams" on page 2-354 for diagrams depicting the use of these intermediate fixed-point data types in each filter structure.

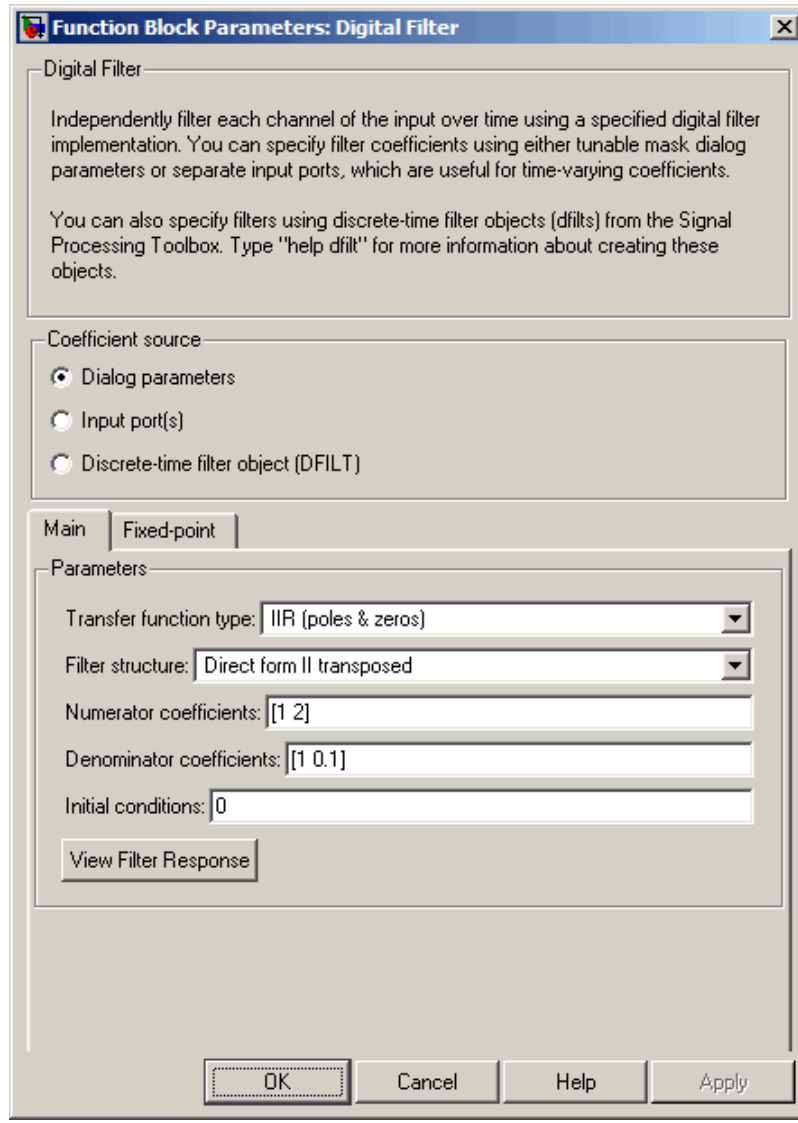
Dialog Box

Different items appear on the Digital Filter block dialog depending on whether you select Dialog parameters, Input port(s), or Discrete-time filter object (DFILT) in the Coefficient source group box. See the following sections for details:

- "Specify Filter Characteristics in Dialog and/or Through Input Ports" on page 2-339
- "Specify Discrete-Time Filter Object" on page 2-350

Specify Filter Characteristics in Dialog and/or Through Input Ports

The Main pane of the Digital Filter block dialog appears as follows when Dialog parameters is specified in the Coefficient source group box. The parameters below can appear when Dialog parameters or Input port(s) is selected, as noted.


Function Block Parameters: Digital Filter
Digital Filter
Independently filter each channel of the input over time using a specified digital filter implementation. You can specify filter coefficients using either tunable mask dialog parameters or separate input ports, which are useful for time-varying coefficients.
You can also specify filters using discrete-time filter objects (dfilts) from the Signal Processing Toolbox. Type "help dfilt" for more information about creating these objects.
Coefficient source
Dialog parameters
Input port(s)
Discrete-time filter object (DFILT)
Main
Fixed-point
Parameters
Transfer function type: IIR (poles & zeros)
Filter structure: Direct form II transposed
Numerator coefficients: [1 2]
Denominator coefficients: [1 0.1]
Initial conditions: 0
View Filter Response
OK
Cancel
Help
Apply

Transfer function type

Select the type of transfer function of the filter; IIR (poles & zeros), IIR (all poles), or FIR (all zeros). See "Supported Filter Structures" on page 2-333 for more information.

Filter structure

Select the filter structure. The selection of available structures varies depending the setting of the Transfer function type parameter. See "Supported Filter Structures" on page 2-333 for more information.

Numerator coefficients

Specify the vector of numerator coefficients of the filter's transfer function.

This parameter is only visible when Dialog parameters is selected and when the selected filter structure lends itself to specification with numerator coefficients. Tunable.

Denominator coefficients

Specify the vector of denominator coefficients of the filter's transfer function.

This parameter is only visible when Dialog parameters is selected and when the selected filter structure lends itself to specification with denominator coefficients. Tunable.

Reflection coefficients

Specify the vector of reflection coefficients of the filter's transfer function.

This parameter is only visible when Dialog parameters is selected and when the selected filter structure lends itself to specification with reflection coefficients. Tunable.

SOS matrix (Mx6)

Specify an M-by-6 SOS matrix containing coefficients of a second-order section (SOS) filter, where M is the number of sections. You can use the ss2sos and tf2sos functions from

Signal Processing Toolbox software to check whether your SOS matrix is valid. For more on the requirements of the SOS matrix, see “Specifying the SOS Matrix (Biquadratic Filter Coefficients)”.

This parameter is only visible when Dialog parameters is selected and when the selected filter structure is biquadratic. Tunable.

Scale values

Specify the scale values to be applied before and after each section of a biquadratic filter.

- If you specify a scalar, that value is applied before the first filter section. The rest of the scale values are set to 1.
- You can also specify a vector with M + 1 elements, assigning a different value to each scale. See “Filter Structure Diagrams” on page 2-354 for diagrams depicting the use of scale values in biquadratic filter structures.

This parameter is only visible when Dialog parameters is selected and when the selected filter structure is biquadratic. Tunable.

First denominator coefficient = 1, remove a0 term in the structure

Select this parameter to reduce the number of computations the block must make to produce the output by omitting the $1 / a_0$ term in the filter structure. The block output is invalid if you select this parameter when the first denominator filter coefficient is not always 1 for your time-varying filter.

This parameter is only enabled when the Input port(s) is selected and when the selected filter structure lends itself to this specification. See “Removing the a0 Term in the Filter Structure” for a diagram and details.

Coefficient update rate

Specify how often the block updates time-varying filters; once per sample or once per frame. This parameter only affects the output when the input signal is frame based.

This parameter is only visible when the Input port(s) is selected and when the selected filter structure lends itself to this specification. For more information, see “Specifying Time-Varying Filters”.

Initial conditions

Specify the initial conditions of the filter states. To learn how to specify initial conditions, see “Specifying Initial Conditions” on page 2-335.

Initial conditions on zeros side

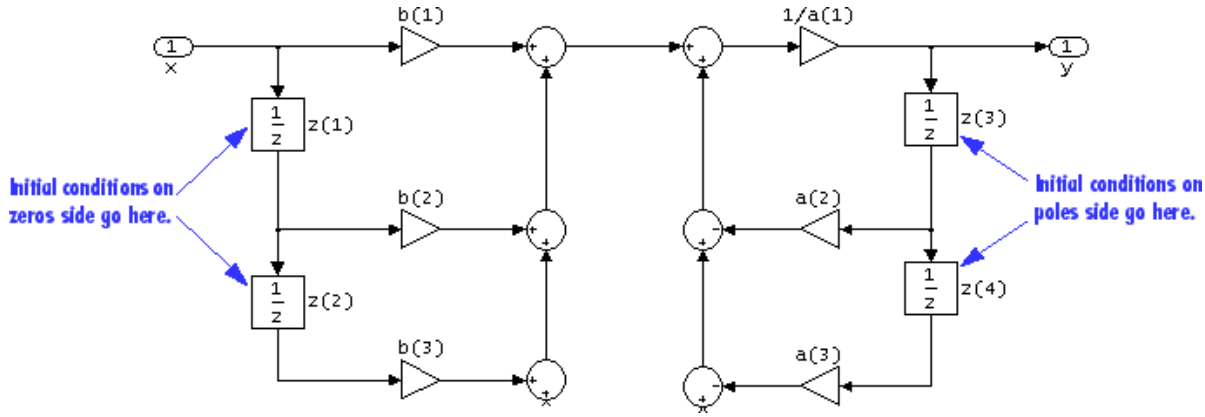
(Not shown in dialog above.) Specify the initial conditions for the filter states on the side of the filter structure with the zeros (b_0, b_1,b_2, ...); see the diagram below.

This parameter is enabled only when the filter has both poles and zeros, and when you select a structure such as direct form I, which has separate filter states corresponding to the poles (a_k) and zeros (b_k). To learn how to specify initial conditions, see “Specifying Initial Conditions” on page 2-335.

Initial conditions on poles side

(Not shown in dialog above). Specify the initial conditions for the filter states on the side of the filter structure with the poles (a_0, a_1,a_2, ...); see the diagram below.

This parameter is enabled only when the filter has both poles and zeros, and when you select a structure such as direct form I, which has separate filter states corresponding to the poles (a_k) and zeros (b_k). To learn how to specify initial conditions, see “Specifying Initial Conditions” on page 2-335.

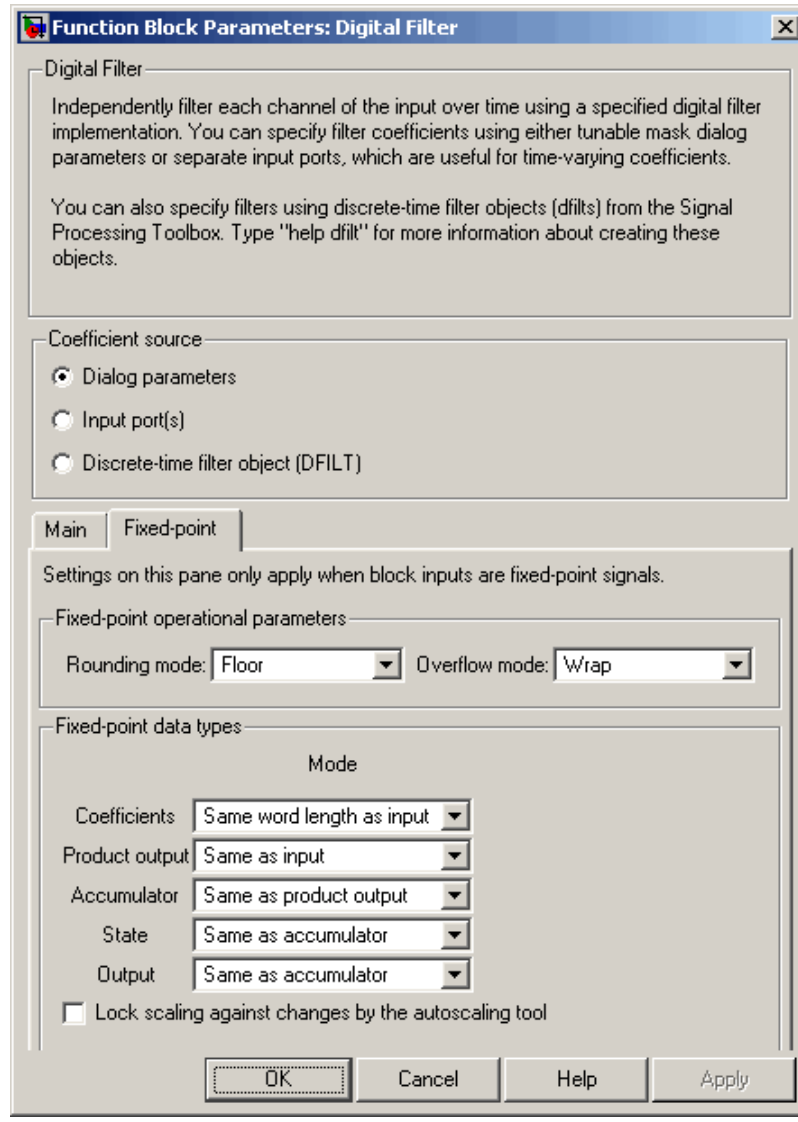


View filter response

This button opens the Filter Visualization Tool (fvtool) from the Signal Processing Toolbox product and displays the filter response of the filter defined by the block. For more information on FVTool, see the Signal Processing Toolbox documentation.

Note This button is only available when the Filter Design Toolbox product is installed. If you specify a filter in the Filter parameter, you must apply the filter by clicking the Apply button before using the View filter response button.

The Fixed point pane of the Digital Filter block dialog appears as follows when Dialog parameters is specified in the Coefficient source group box. The parameters below can appear when Dialog parameters or Input port(s) is selected, depending on the filter structure and whether the coefficients are being entered via ports or on the block mask.


Function Block Parameters: Digital Filter
Digital Filter
Independently filter each channel of the input over time using a specified digital filter implementation. You can specify filter coefficients using either tunable mask dialog parameters or separate input ports, which are useful for time-varying coefficients.
You can also specify filters using discrete-time filter objects (dfilts) from the Signal Processing Toolbox. Type "help dfilt" for more information about creating these objects.
Coefficient source
Dialog parameters
Input port(s)
Discrete-time filter object (DFILT)
Main
Fixed-point
Settings on this pane only apply when block inputs are fixed-point signals.
Fixed-point operational parameters
Rounding mode: Floor
Overflow mode: Wrap
Fixed-point data types
Mode
Coefficients
Same word length as input
Product output
Same as input
Accumulator
Same as product output
State
Same as accumulator
Output
Same as accumulator
Lock scaling against changes by the autoscaling tool
OK
Cancel
Help
Apply

Rounding mode

Select the rounding mode for fixed-point operations. The filter coefficients do not obey this parameter; they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations. The filter coefficients do not obey this parameter; they are always saturated.

Section I/O

Choose how you specify the word length and the fraction length of the fixed-point data type going into and coming out of each section of a biquadratic filter. See "Filter Structure Diagrams" on page 2-354 for illustrations depicting the use of the section I/O data type in this block.

This parameter is only visible when the selected filter structure is biquadratic:

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word and fraction lengths of the section input and output, in bits.
- When you select Slope and bias scaling, you can enter the word lengths, in bits, and the slopes of the section input and output. This block requires power-of-two slope and a bias of zero.

Tap sum

Choose how you specify the word length and the fraction length of the tap sum data type of a direct form symmetric or direct form antisymmetric filter. See "Filter Structure Diagrams" on page 2-354 for illustrations depicting the use of the tap sum data type in this block.

This parameter is only visible when the selected filter structure is either Direct form symmetric or Direct form antisymmetric:

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the tap sum accumulator, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the tap sum accumulator. This block requires power-of-two slope and a bias of zero.

Multiplicand

Choose how you specify the word length and the fraction length of the multiplicand data type of a direct form I transposed or biquadratic direct form I transposed filter. See "Filter Structure Diagrams" on page 2-354 for illustrations depicting the use of the multiplicand data type in this block.

This parameter is only visible when the selected filter structure is either Direct form I transposed or Biquad direct form I transposed (SOS):

- When you select Same as output, these characteristics match those of the output to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the multiplicand data type, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the multiplicand data type. This block requires power-of-two slope and a bias of zero.

Coefficients

Choose how you specify the word length and the fraction length of the filter coefficients (numerator and/or denominator). See "Filter Structure Diagrams" on page 2-354 for illustrations depicting the use of the coefficient data types in this block:

- When you select Same word length as input, the word length of the filter coefficients match that of the input to the

block. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.

- When you select Specify word length, you can enter the word length of the coefficients, in bits. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select Binary point scaling, you can enter the word length and the fraction length of the coefficients, in bits. If applicable, you can enter separate fraction lengths for the numerator and denominator coefficients.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the coefficients. If applicable, you can enter separate slopes for the numerator and denominator coefficients. This block requires power-of-two slope and a bias of zero.
- The filter coefficients do not obey the Rounding mode and the Overflow mode parameters; they are always saturated and rounded to Nearest.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Filter Structure Diagrams” on page 2-354 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.

- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths. See “Filter Structure Diagrams” on page 2-354 and “Multiplication Data Types” for illustrations depicting the use of the accumulator data type in this block:

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Same as product output, these characteristics match those of the product output.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

State

Use this parameter to specify how you would like to designate the state word and fraction lengths. See “Filter Structure Diagrams” on page 2-354 for illustrations depicting the use of the state data type in this block.

This parameter is not visible for direct form and direct form I filter structures.

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Same as accumulator, these characteristics match those of the accumulator.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.

- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

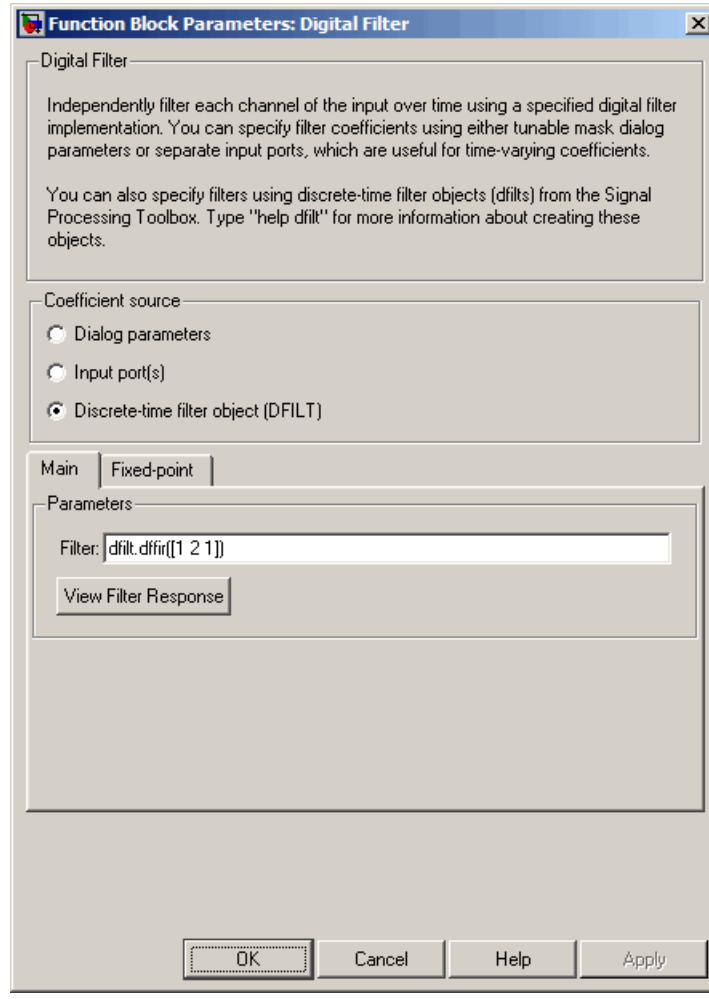
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Same as accumulator, these characteristics match those of the accumulator.
- When you select Binary point scaling, you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool.

Specify Discrete-Time Filter Object

The Main pane of the Digital Filter block dialog appears as follows when Discrete-time filter object (DFILT) is specified in the Coefficient source group box.



Filter

Specify the discrete-time filter object (dfilt) that you would like the block to implement. You can do this in one of three ways:

- You can fully specify the dfilt object in the block mask, as shown in the default value.
- You can enter the variable name of a dfilt object that is deÞned in any workspace.
- You can enter a variable name for a dfilt object that is not yet deÞned.

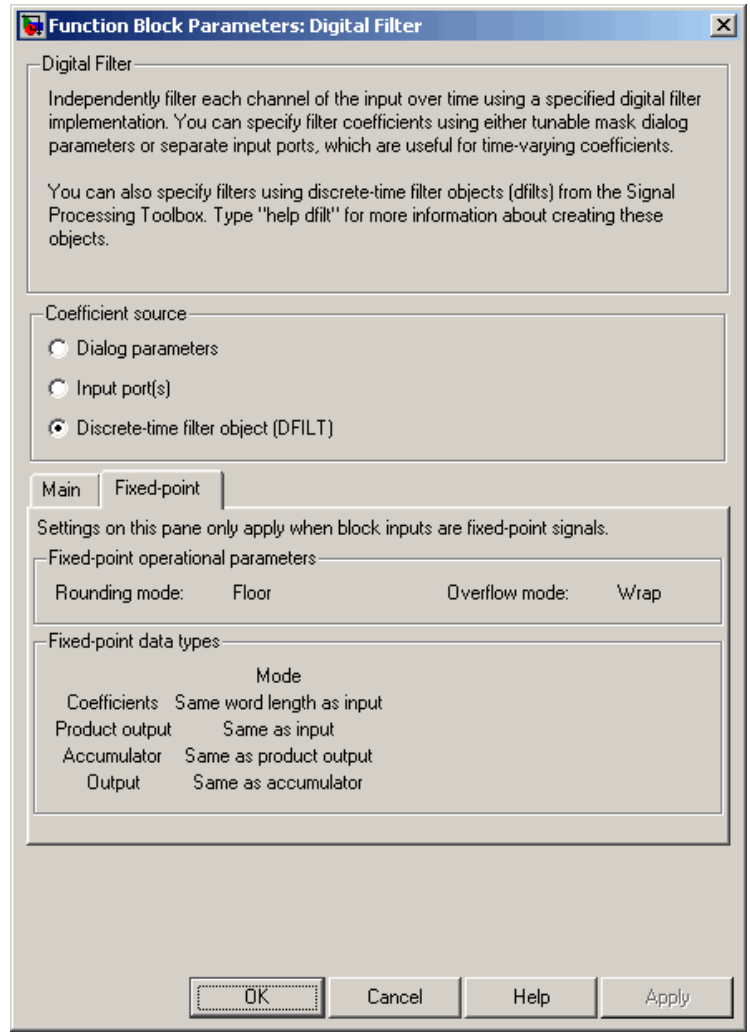
For more information on creating dfilt objects, see the dfilt function reference page in the Signal Processing Toolbox documentation or the Filter Design Toolbox documentation.

View filter response

This button opens the Filter Visualization Tool (fvtool) from the Signal Processing Toolbox product and displays the Þlter response of the dfilt object speciÞed in the Filter parameter. For more information on FVTool, see the Signal Processing Toolbox documentation.

Note This button is only available when the Filter Design Toolbox product is installed. If you specify a Þlter in the Filter parameter, you must apply the Þlter by clicking the Apply button before using the View filter response button.

The Fixed-point pane of the Digital Filter block dialog appears as follows when Discrete-time filter object (DFILT) is speciÞed in the Coefficient source group box.



The fixed-point settings of the filter object specified on the Main pane are displayed on the Fixed-point pane. You cannot change these settings directly on the block mask. To change the fixed-point settings you must edit the filter object directly.

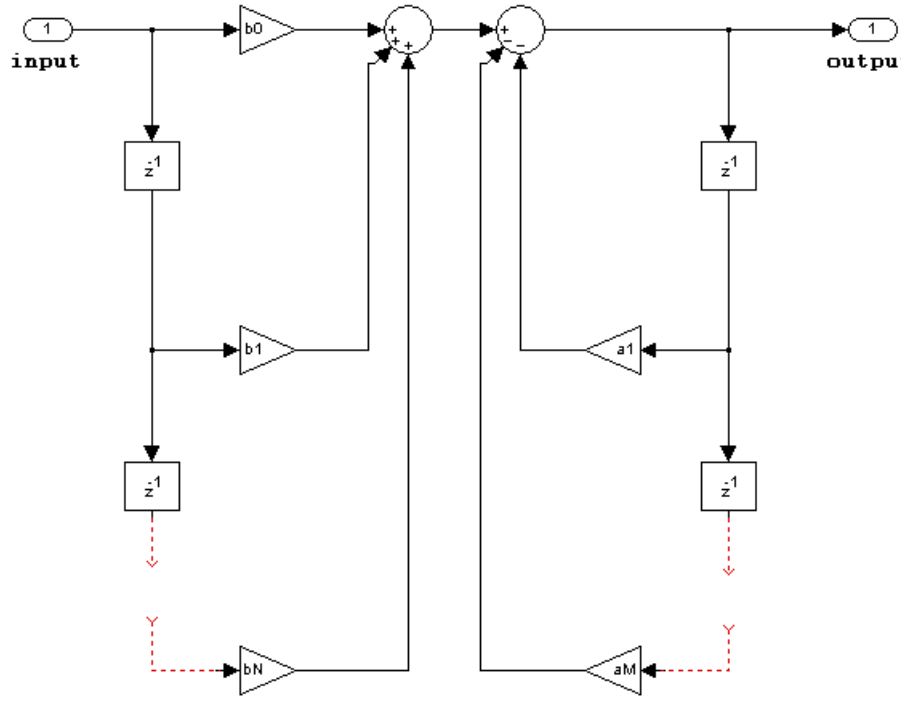
For more information on discrete-time filter objects, see the dfilt function reference page in the Signal Processing Toolbox documentation or the Filter Design Toolbox documentation.

Filter Structure Diagrams

The diagrams in the following sections show the filter structures supported by the Digital Filter block. They also show the data types used in the filter structures for fixed-point signals. You can set the coefficient, output, accumulator, product output, and state data types shown in these diagrams in the block dialog. This is discussed in "Dialog Box" on page 2-339.

- "IIR direct form I" on page 2-355
- "IIR direct form I transposed" on page 2-357
- "IIR direct form II" on page 2-360
- "IIR direct form II transposed" on page 2-362
- "IIR biquadratic direct form I" on page 2-365
- "IIR biquadratic direct form I transposed" on page 2-368
- "IIR biquadratic direct form II" on page 2-371
- "IIR biquadratic direct form II transposed" on page 2-373
- "IIR (all poles) direct form" on page 2-376
- "IIR (all poles) direct form transposed" on page 2-378
- "IIR (all poles) direct form lattice AR" on page 2-380
- "FIR (all zeros) direct form" on page 2-381
- "FIR (all zeros) direct form symmetric" on page 2-383
- "FIR (all zeros) direct form antisymmetric" on page 2-385
- "FIR (all zeros) direct form transposed" on page 2-387
- "FIR (all zeros) lattice MA" on page 2-389

IIR direct form I

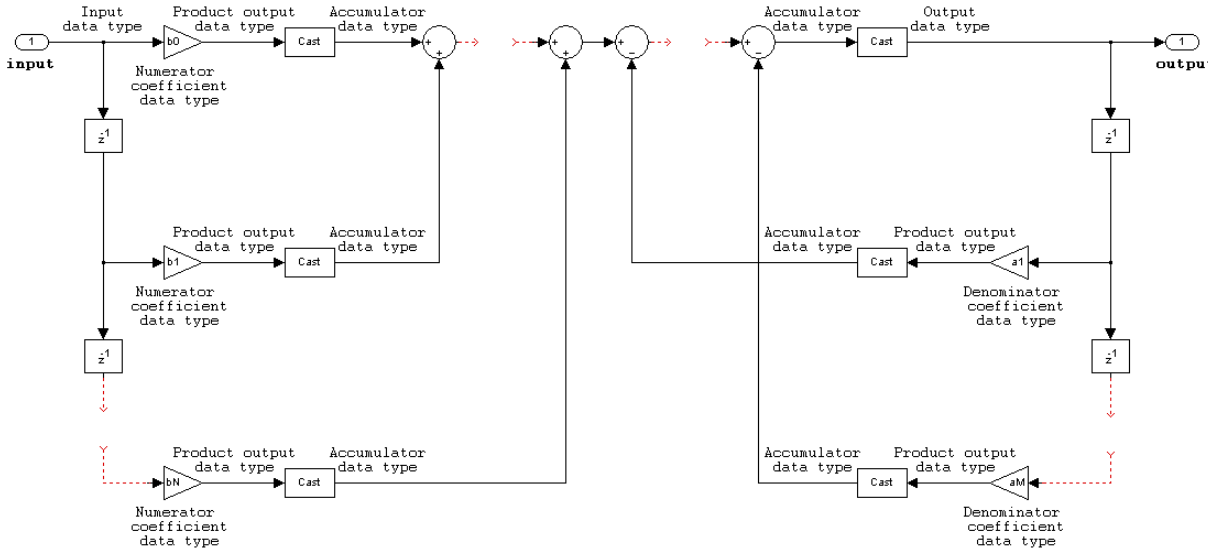


The following constraints are applicable when processing a fixed-point signal with this filter structure:

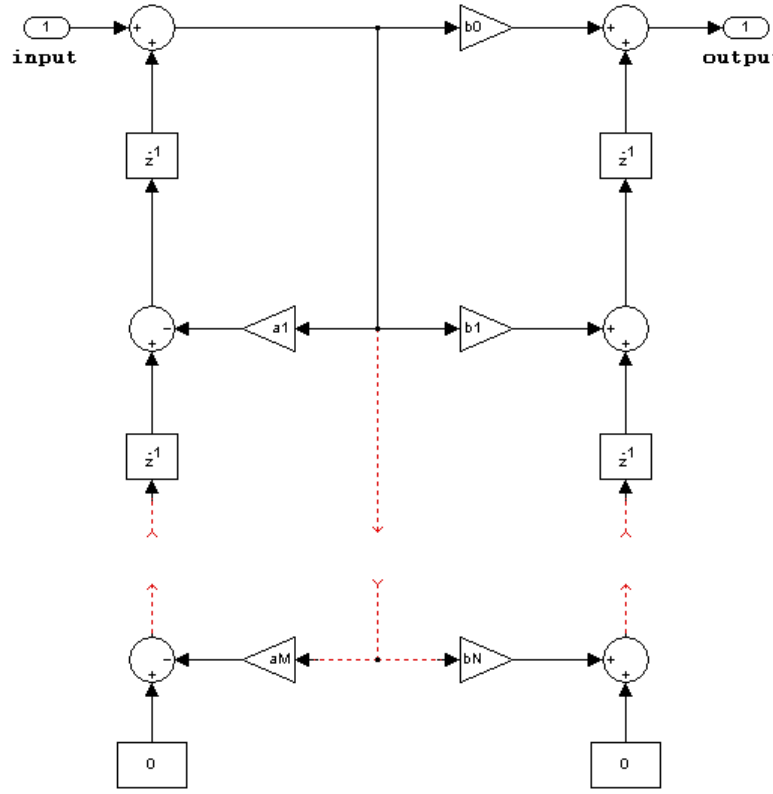
- Inputs can be real or complex.
- Numerator and denominator coefficients can be real or complex.
- Numerator and denominator coefficients must be the same complexity as each other.
  - When the numerator and denominator coefficients are specified via input ports and have different complexities from each other, you get an error.

- When the numerator and denominator coefficients are specified in the dialog and have different complexities from each other, the block does not error, but instead processes the filter as if two sets of complex coefficients are provided. The coefficient set that is real-valued is treated as if it is a complex vector with zero-valued imaginary parts.

- Numerator and denominator coefficients must have the same word length. They can have different fraction lengths.
- The State data type cannot be specified on the block mask for this structure, because the input and output states have the same data types as the input and output buffers.



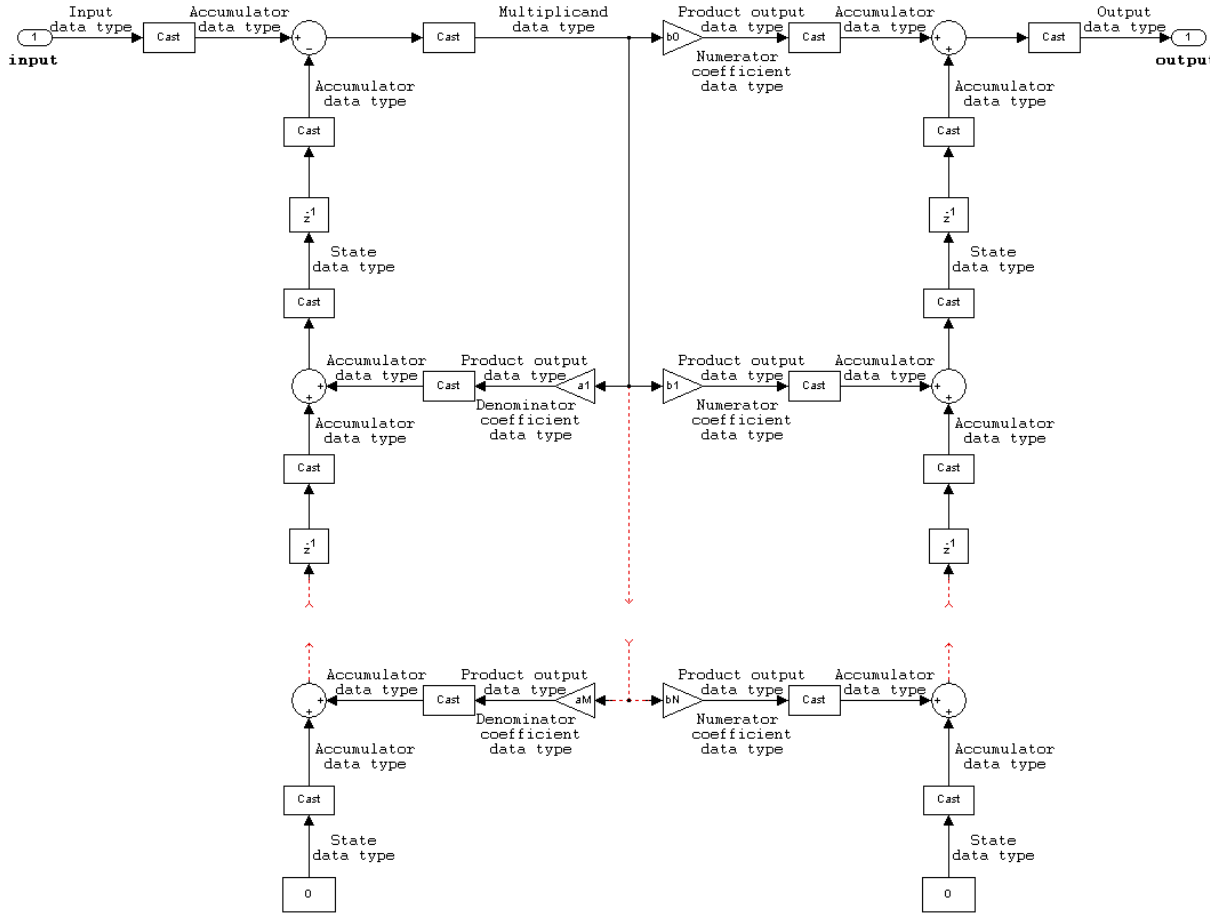
IIR direct form I transposed



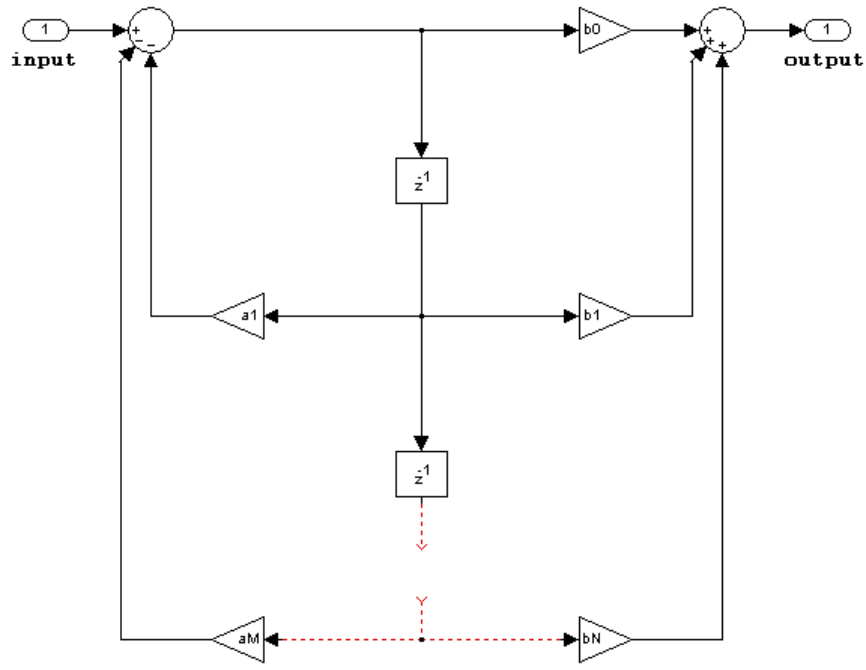
The following constraints are applicable when processing a fixed-point signal with this filter structure:

- Inputs can be real or complex.
- Numerator and denominator coefficients can be real or complex.
- Numerator and denominator coefficients must be the same complexity as each other.

  - When the numerator and denominator coefficients are specified via input ports and have different complexities from each other, you get an error.
  - When the numerator and denominator coefficients are specified in the dialog and have different complexities from each other, the block does not error, but instead processes the filter as if two sets of complex coefficients are provided. The coefficient set that is real-valued is treated as if it is a complex vector with zero-valued imaginary parts.
- States are complex when either the input or the coefficients are complex.
- Numerator and denominator coefficients must have the same word length. They can have different fraction lengths.


Input
data type
Accumulator
data type
Multiplicand
data type
Product output
data type
Accumulator
data type
Output
data type
input
output
Cast
Numerator
coefficient
data type
State
data type
Denominator
coefficient
data type
bO
b1
bN
a1
aM
0

IIR direct form II

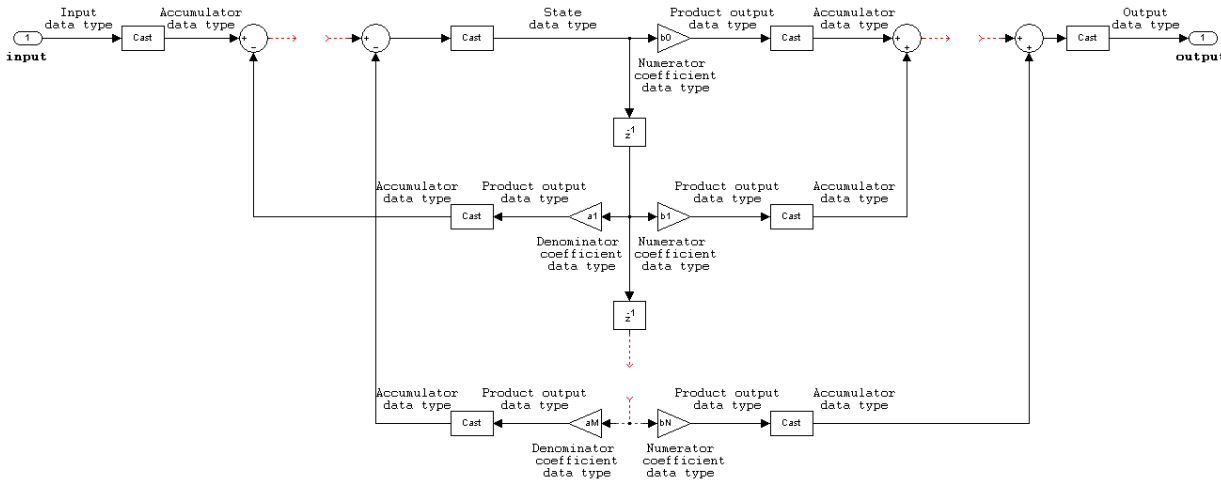


The following constraints are applicable when processing a fixed-point signal with this filter structure:

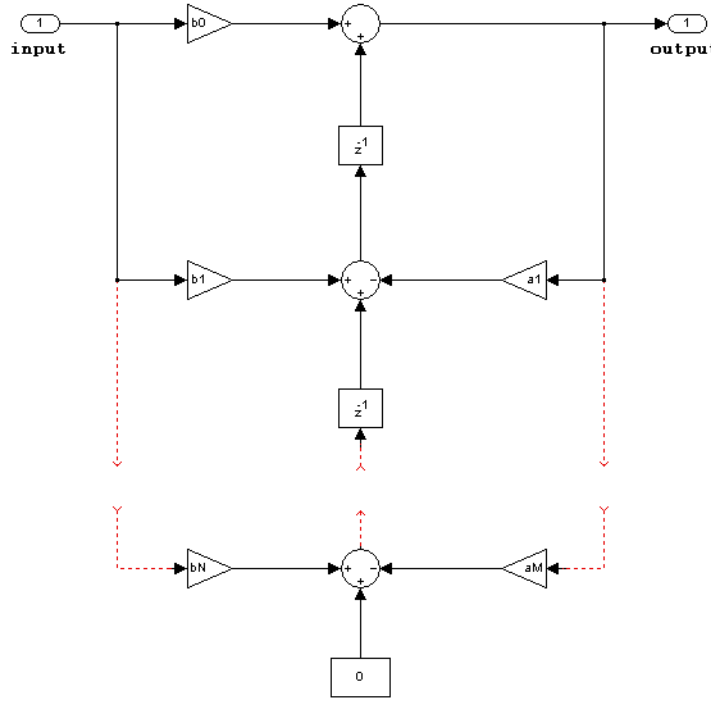
- Inputs can be real or complex.
- Numerator and denominator coefficients can be real or complex.
- Numerator and denominator coefficients must be the same complexity as each other.
  - When the numerator and denominator coefficients are specified via input ports and have different complexities from each other, you get an error.

- When the numerator and denominator coefficients are specified in the dialog and have different complexities from each other, the block does not error, but instead processes the filter as if two sets of complex coefficients are provided. The coefficient set that is real-valued is treated as if it is a complex vector with zero-valued imaginary parts.

- States are complex when either the inputs or the coefficients are complex.
- Numerator and denominator coefficients must have the same word length. They can have different fraction lengths.



IIR direct form II transposed

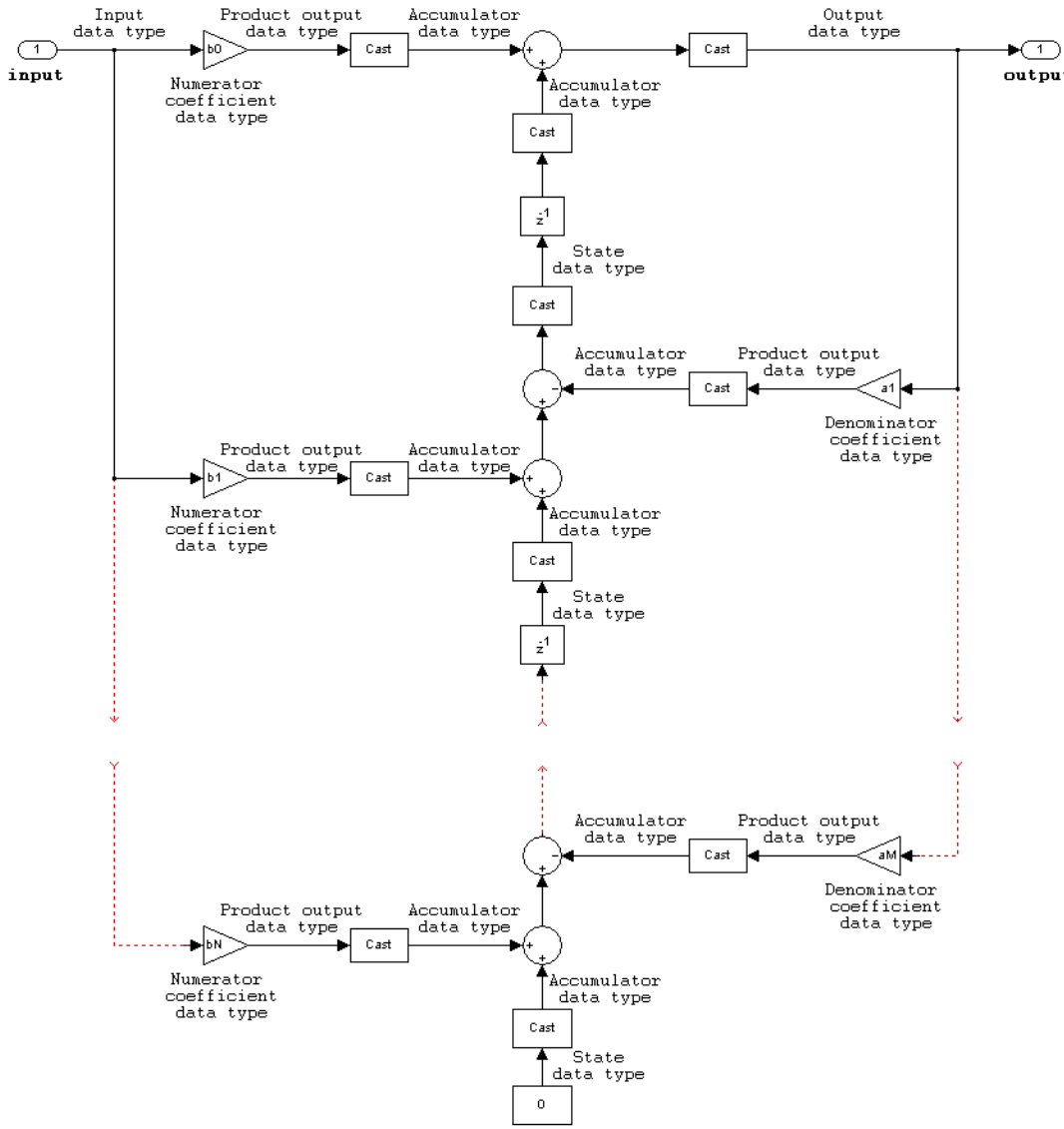


The following constraints are applicable when processing a fixed-point signal with this filter structure:

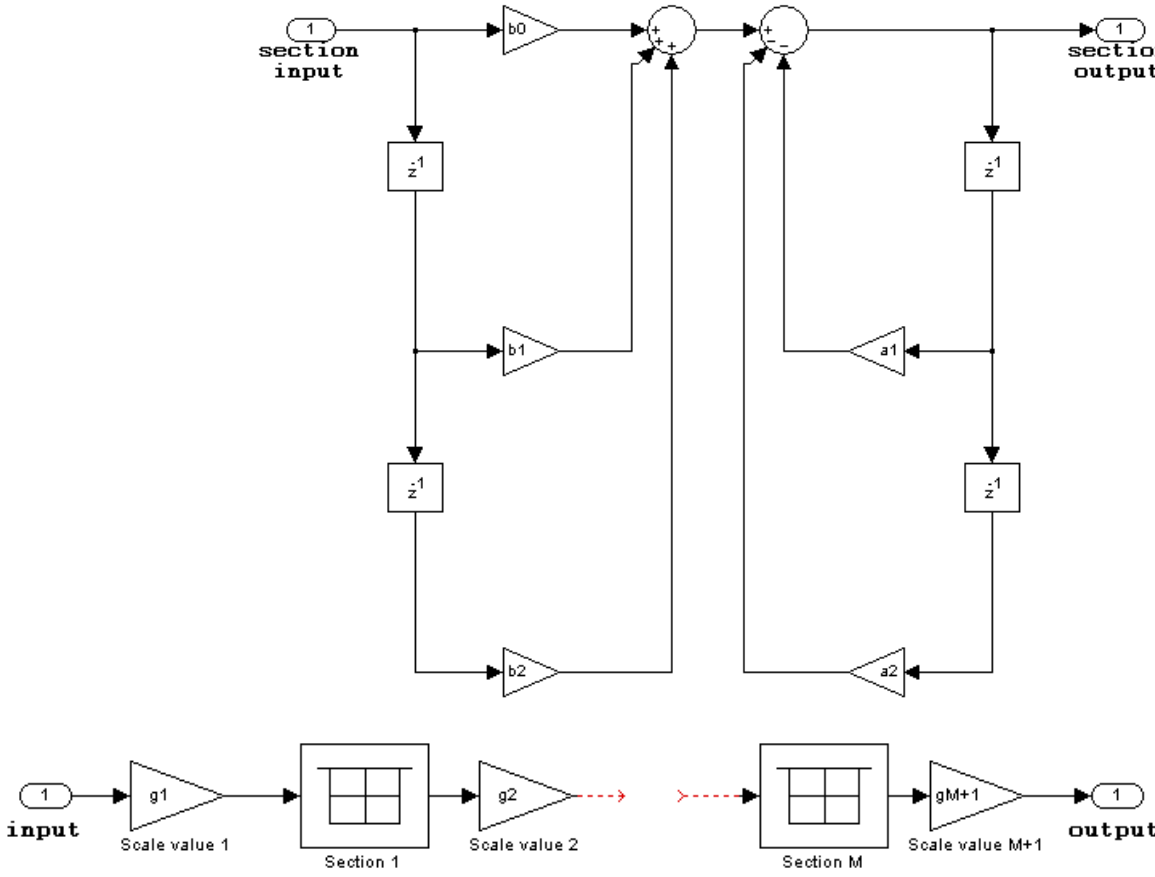
- Inputs can be real or complex.
- Numerator and denominator coefficients can be real or complex.
- Numerator and denominator coefficients must be the same complexity as each other.
  - When the numerator and denominator coefficients are specified via input ports and have different complexities from each other, you get an error.

  - When the numerator and denominator coefficients are specified in the dialog and have different complexities from each other, the block does not error, but instead processes the filter as if two sets of complex coefficients are provided. The coefficient set that is real-valued is treated as if it is a complex vector with zero-valued imaginary parts.

- States are complex when either the inputs or the coefficients are complex.
- Numerator and denominator coefficients must have the same word length. They can have different fraction lengths.


1
input
Input
data type
b0
Numerator
coefficient
data type
Product output
data type
Cast
Accumulator
data type
Cast
Output
data type
1
output
Accumulator
data type
Cast
z-1
State
data type
Cast
Accumulator
data type
Cast
Product output
data type
a1
Denominator
coefficient
data type
b1
Numerator
coefficient
data type
Product output
data type
Cast
Accumulator
data type
Accumulator
data type
Cast
State
data type
z-1
aM
Denominator
coefficient
data type
bN
Numerator
coefficient
data type
0

IIR biquadratic direct form I

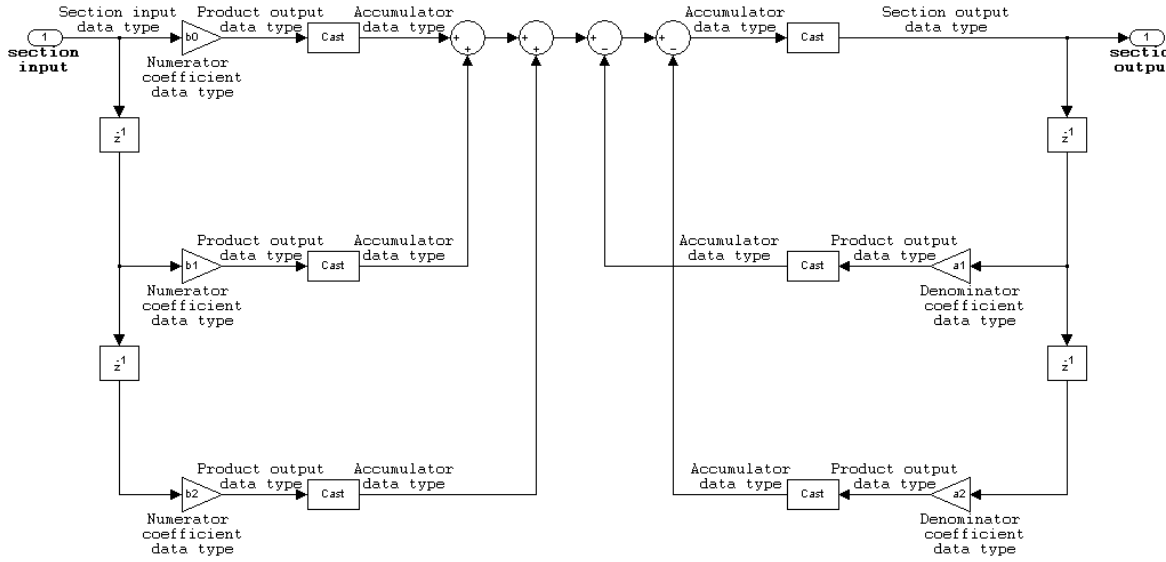


The following constraints are applicable when processing a fixed-point signal with this filter structure:

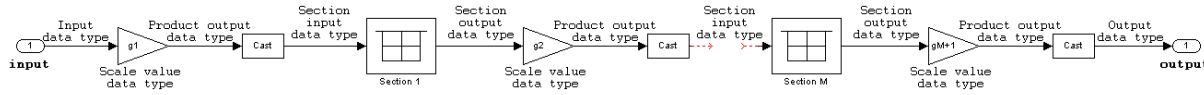
- Inputs and coefficients can be real or complex.
- Numerator and denominator coefficients can be real or complex.

- Specify the coefficients by a M-by-6 matrix in the block mask. You cannot specify coefficients by input ports for this filter structure.
- When the a0 element of any row is not equal to one, that row is normalized by a0 prior to filtering.
- States are complex when either the inputs or the coefficients are complex.
- You cannot specify the state data type on the block mask for this structure, because the input and output states have the same data types as the input.
- Scale values must have the same complexity as the coefficient SOS matrix.
- The scale value parameter must be a scalar or a vector of length M+1, where M is the number of sections.
- The Section I/O parameter determines the data type for the section input and output data types. The section input and stage output data type must have the same word length but can have different fraction lengths.

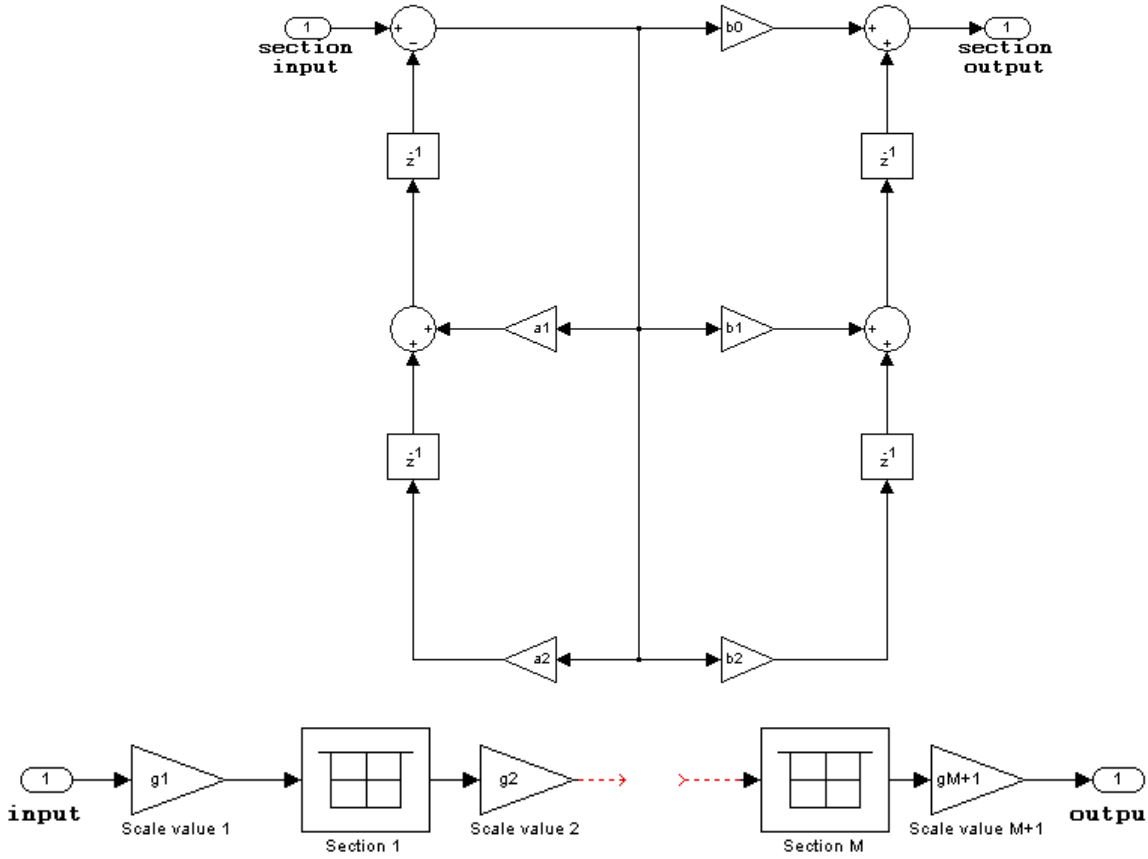
The following diagram shows the data types for one section of the filter.



The following diagram shows the data types between filter sections.



IIR biquadratic direct form I transposed

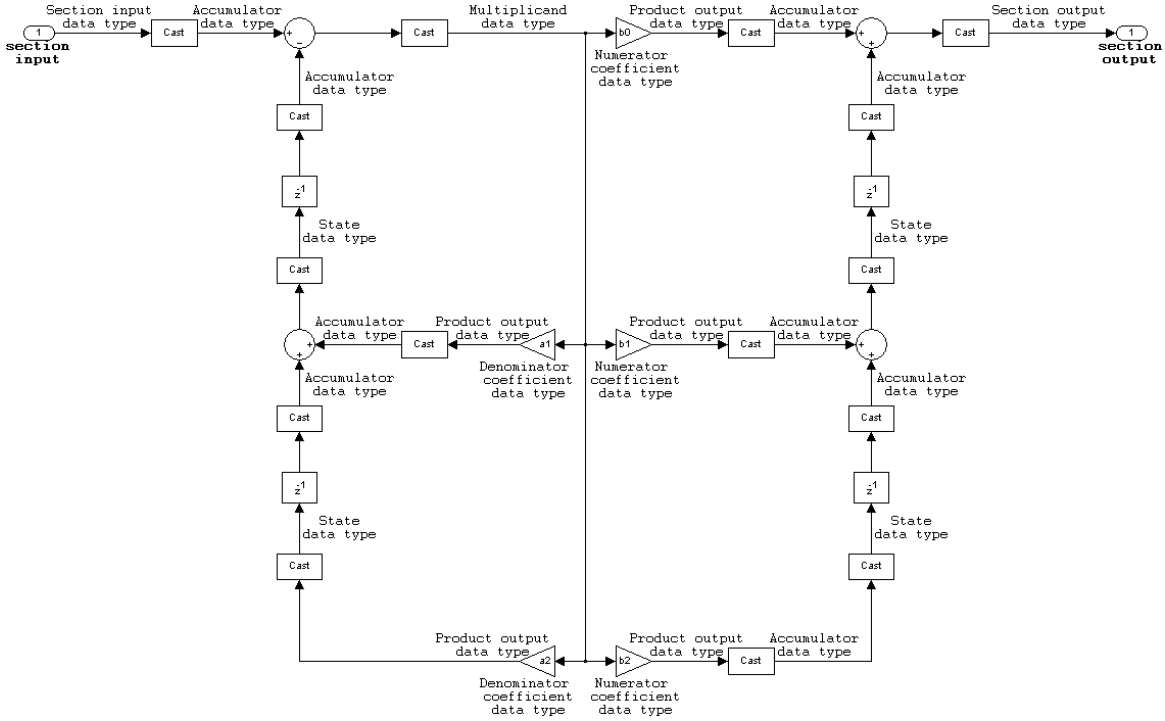


The following constraints are applicable when processing a fixed-point signal with this filter structure:

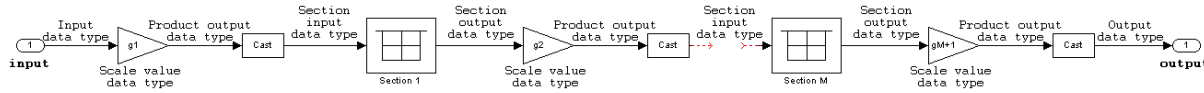
- Inputs and coefficients can be real or complex.
- Numerator and denominator coefficients can be real or complex.

- Specify the coefficients by a M-by-6 matrix in the block mask. You cannot specify coefficients by input ports for this filter structure.
- When the a0 element of any row is not equal to one, that row is normalized by a0 prior to filtering.
- States are complex when either the inputs or the coefficients are complex.
- Scale values must have the same complexity as the coefficient SOS matrix.
- The scale value parameter must be a scalar or a vector of length M+1, where M is the number of sections.
- The Section I/O parameter determines the data type for the section input and output data types. The section input and section output data type must have the same word length but can have different fraction lengths.

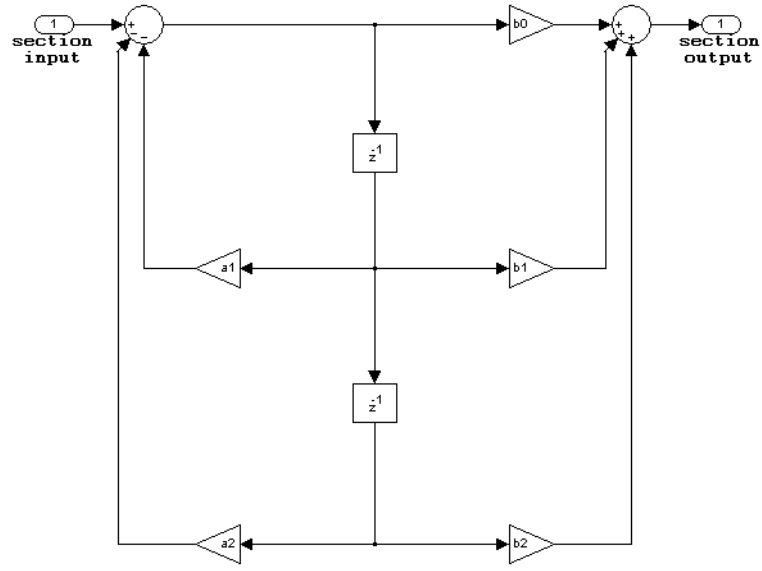
The following diagram shows the data types for one section of the Þlter.

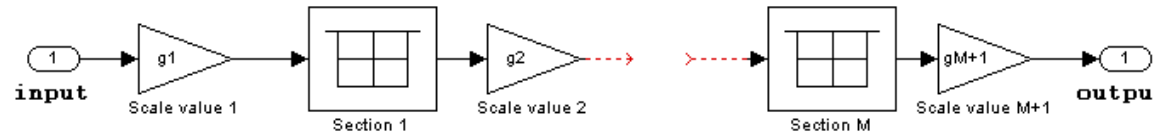


The following diagram shows the data types between filter sections.



IIR biquadratic direct form II



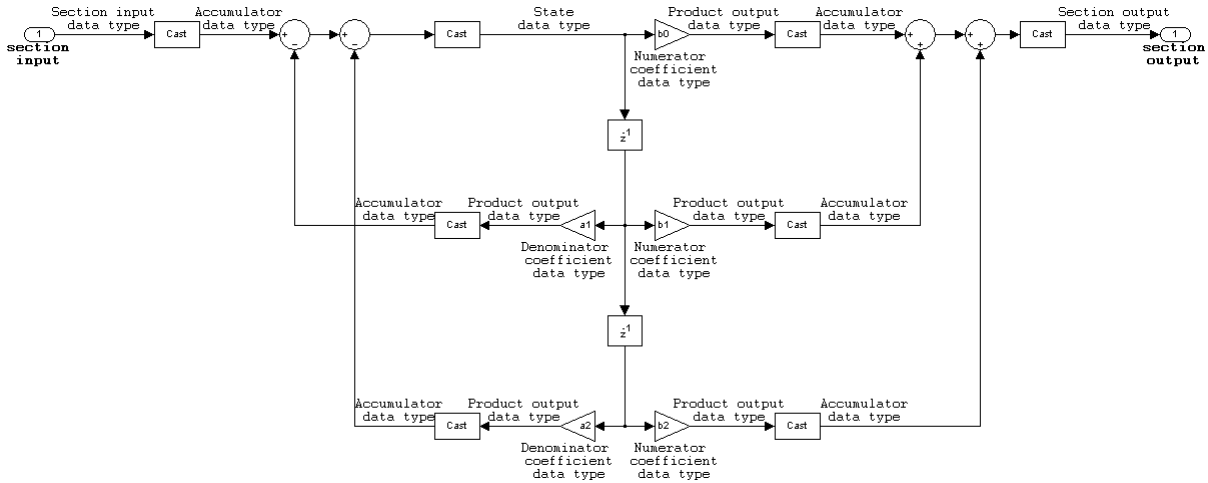


The following constraints are applicable when processing a fixed-point signal with this filter structure:

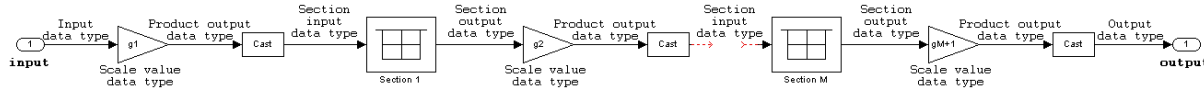
- Inputs and coefficients can be real or complex.
- Numerator and denominator coefficients can be real or complex.

- Specify the coefficients by a M-by-6 matrix in the block mask. You cannot specify coefficients by input ports for this filter structure.
- When the a0 element of any row is not equal to one, that row is normalized by a0 prior to filtering.
- States are complex when either the inputs or the coefficients are complex.
- Scale values must have the same complexity as the coefficient SOS matrix.
- The scale value parameter must be a scalar or a vector of length M+1, where M is the number of sections.
- The Section I/O parameter determines the data type for the section input and output data types. The section input and section output data type must have the same word length but can have different fraction lengths.

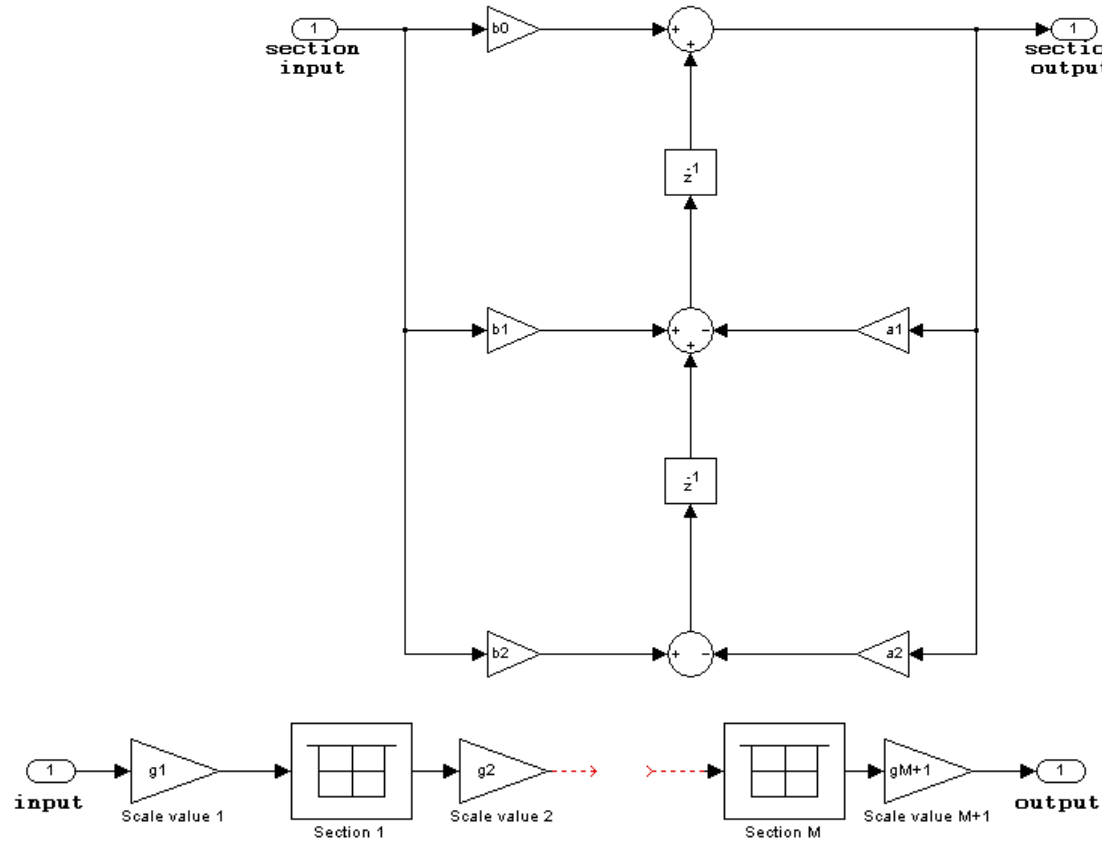
The following diagram shows the data types for one section of the filter.



The following diagram shows the data types between filter sections.



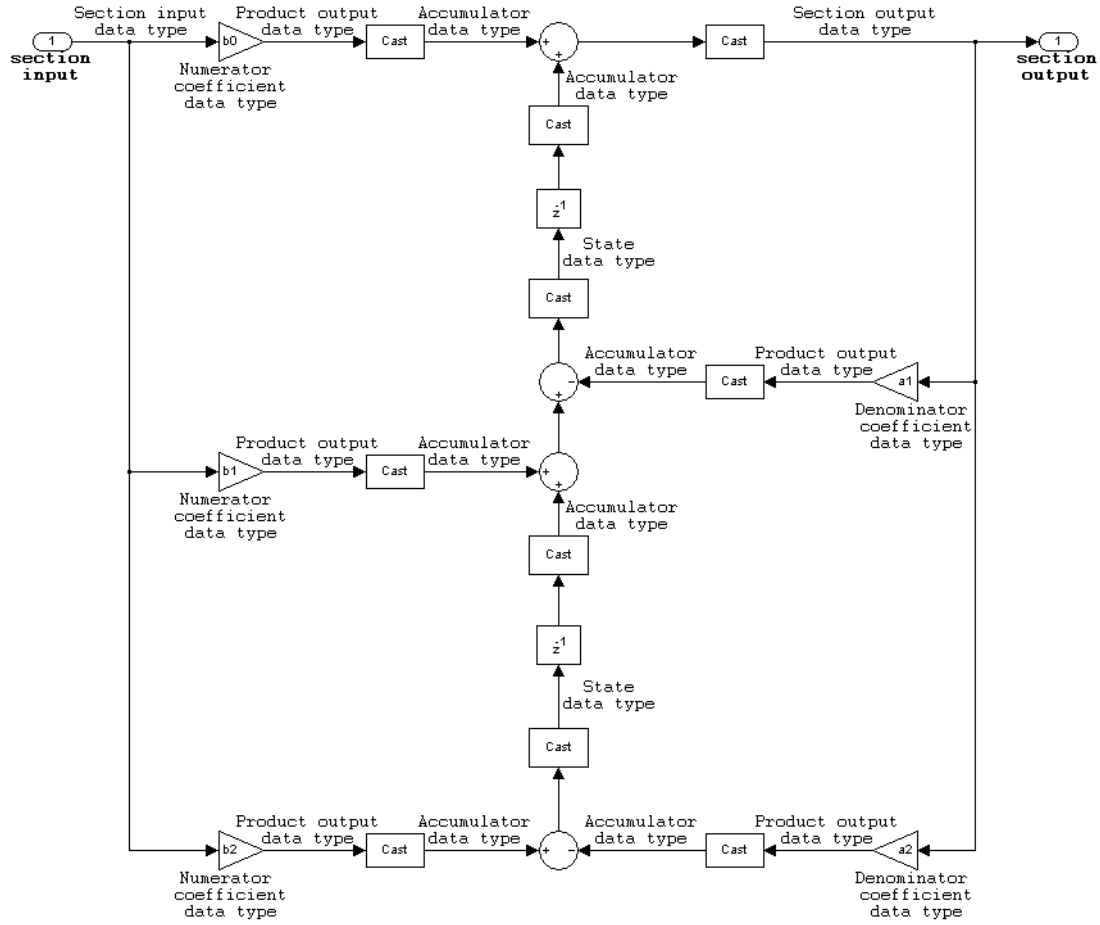
IIR biquadratic direct form II transposed



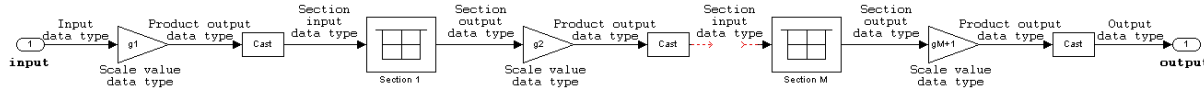
The following constraints are applicable when processing a fixed-point signal with this filter structure:

- Inputs and coefficients can be real or complex.
- Numerator and denominator coefficients can be real or complex.
- Specify the coefficients by a M-by-6 matrix in the block mask. You cannot specify coefficients by input ports for this filter structure.
- When the a0 element of any row is not equal to one, that row is normalized by a0 prior to filtering.
- States are complex when either the inputs or the coefficients are complex.
- Scale values must have the same complexity as the coefficient SOS matrix.
- The scale value parameter must be a scalar or a vector of length M+1, where M is the number of sections.
- The Section I/O parameter determines the data type for the section input and output data types. The section input and section output data type must have the same word length but can have different fraction lengths.

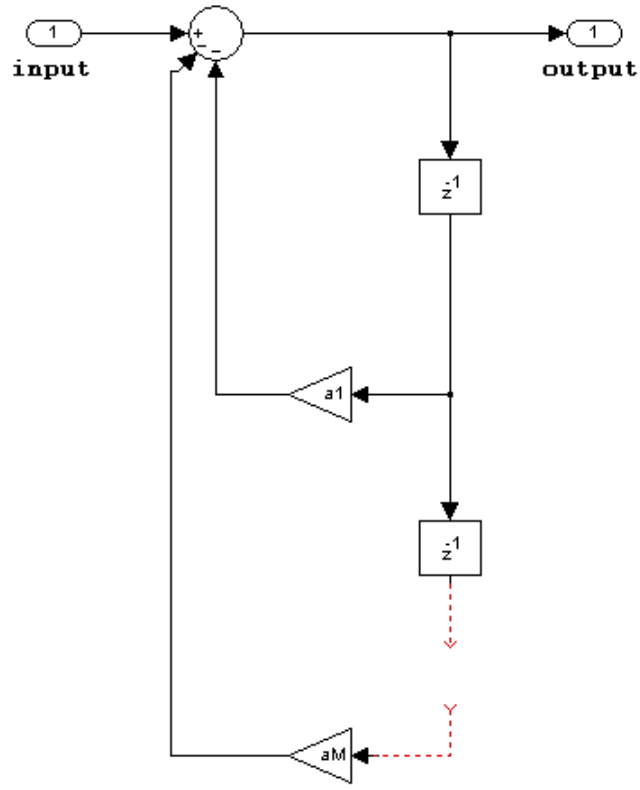
The following diagram shows the data types for one section of the filter.



The following diagram shows the data types between filter sections.

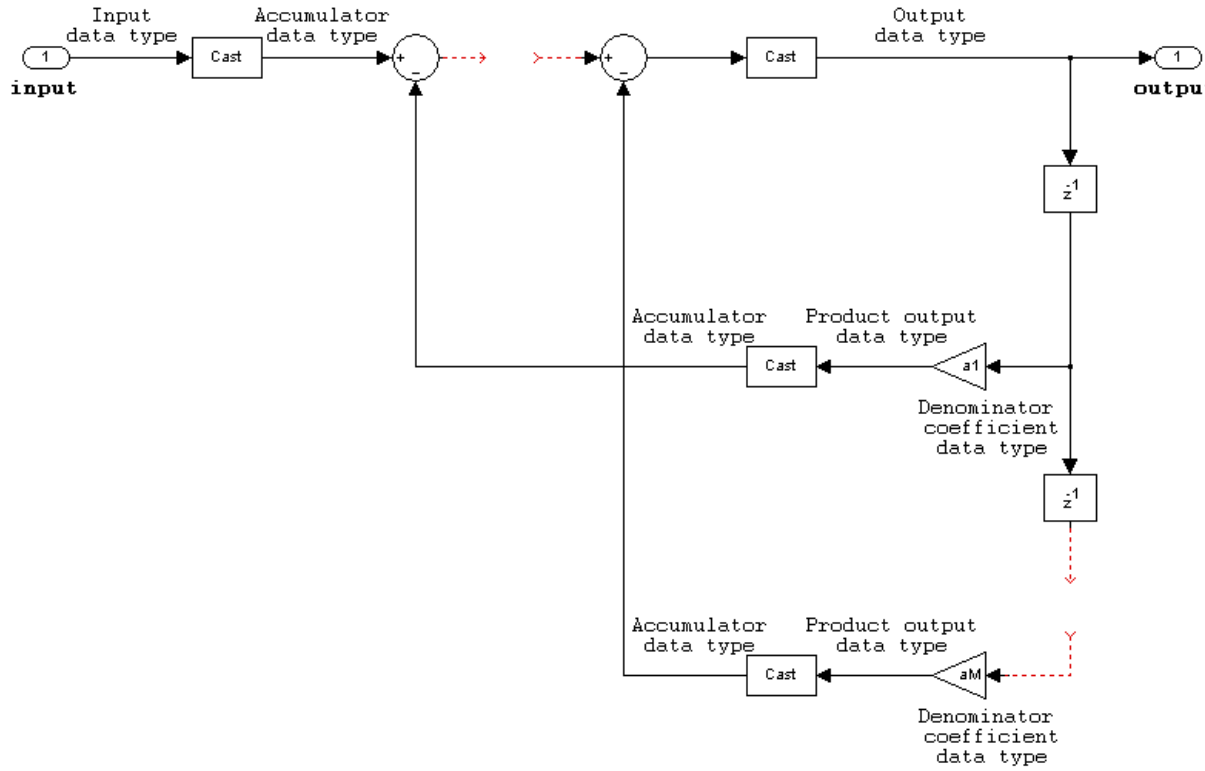


IIR (all poles) direct form

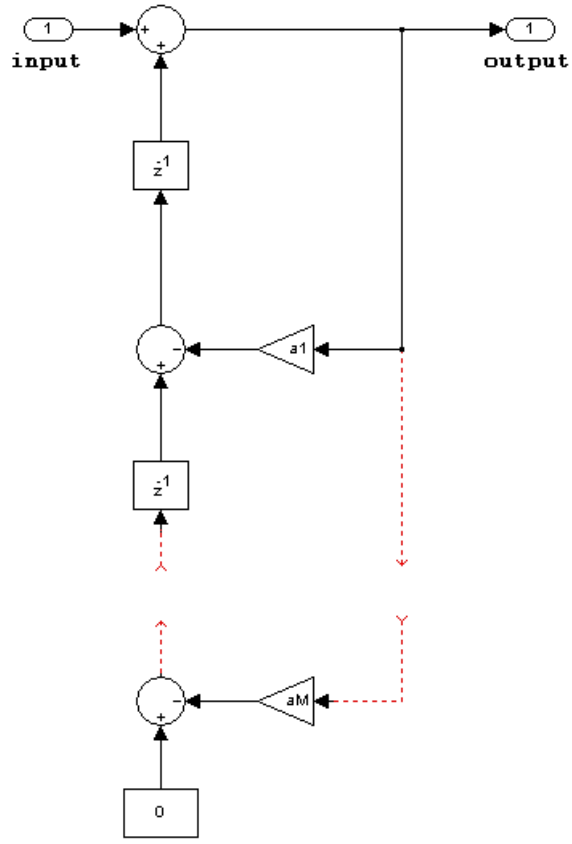


The following constraints are applicable when processing a fixed-point signal with this filter structure:

- Inputs and coefficients can be real or complex.
- Denominator coefficients can be real or complex.
- You cannot specify the state data type on the block mask for this structure, because the input and output states have the same data types as the input.

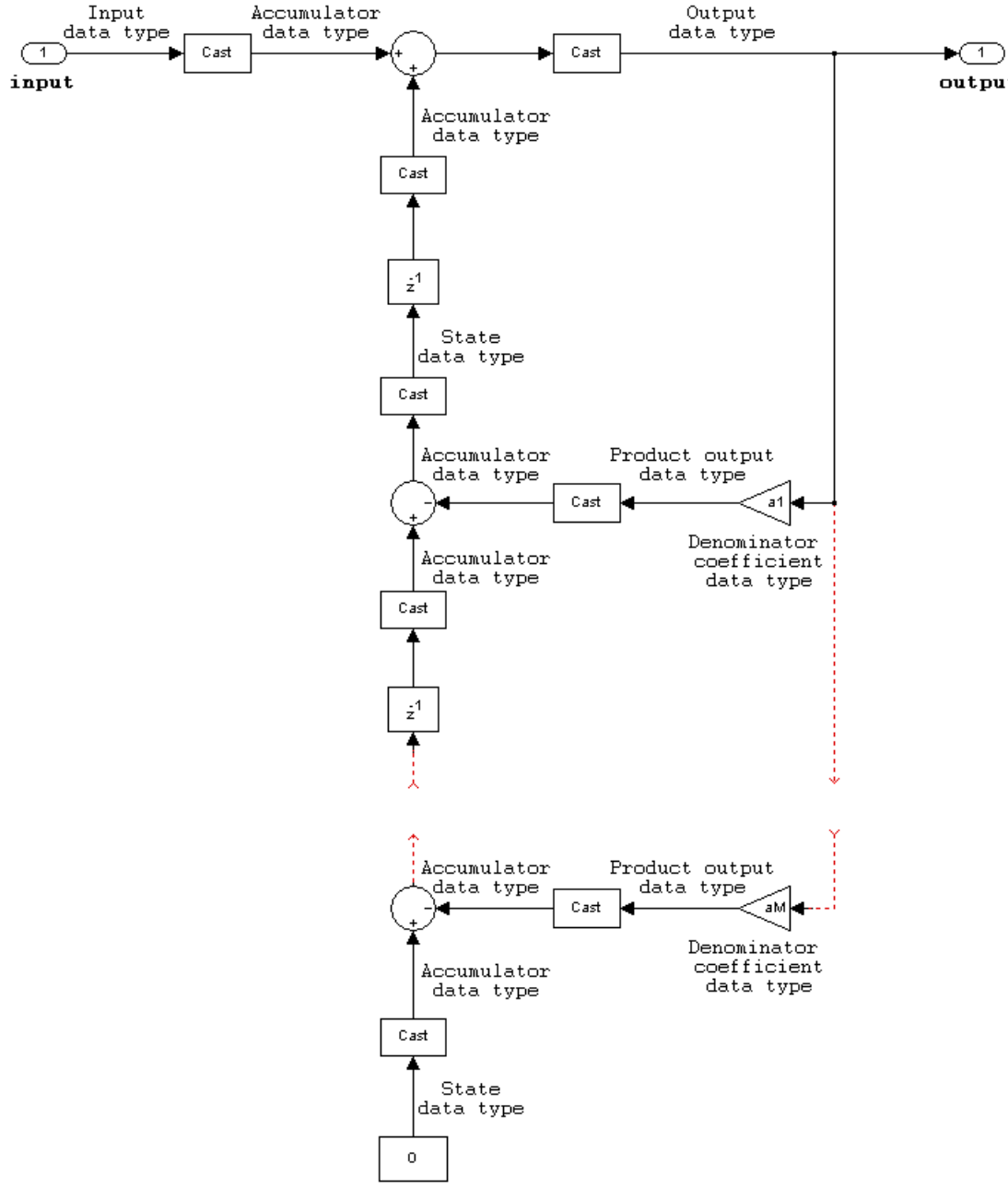

Input
data type
Accumulator
data type
1
input
Cast
Cast
Output
data type
1
output
z-1
Accumulator
data type
Product output
data type
Cast
a1
Denominator
coefficient
data type
z-1
Accumulator
data type
Product output
data type
Cast
aM
Denominator
coefficient
data type

IIR (all poles) direct form transposed

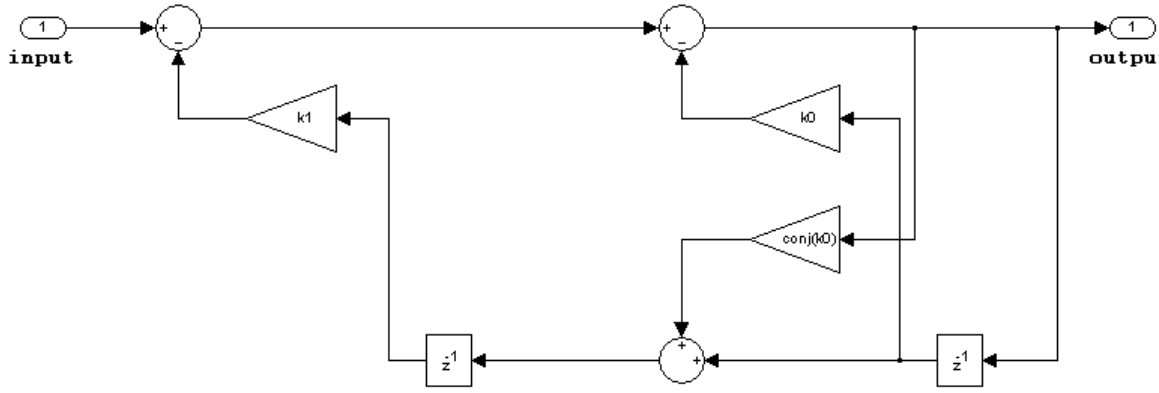


The following constraints are applicable when processing a fixed-point signal with this filter structure:

- Inputs and coefficients can be real or complex.
- Denominator coefficients can be real or complex.

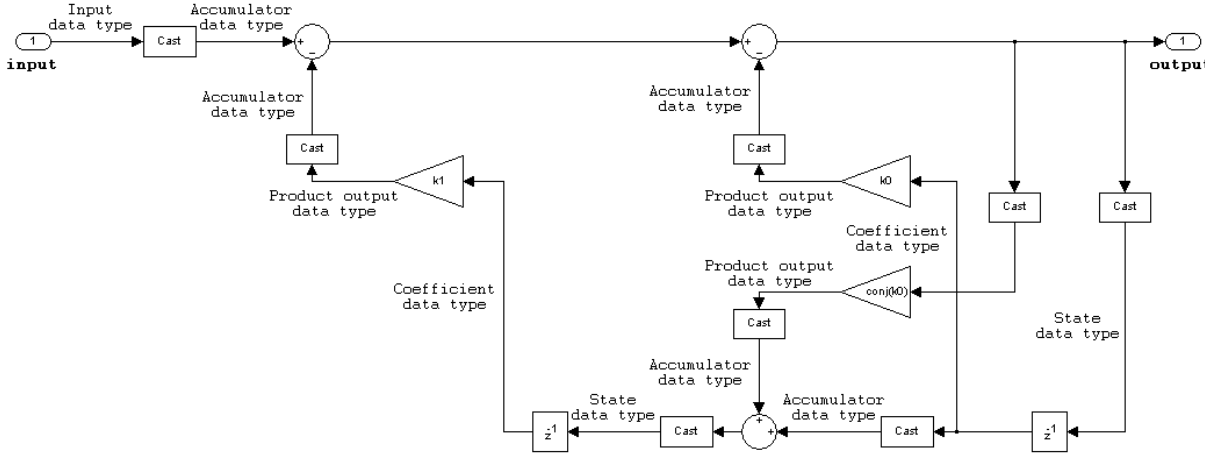

Input
data type
Accumulator
data type
Output
data type
1
input
Cast
Cast
1
output
Accumulator
data type
Cast
z-1
State
data type
Cast
Accumulator
data type
Product output
data type
Cast
a1
Denominator
coefficient
data type
Accumulator
data type
Cast
z-1
Accumulator
data type
Product output
data type
Cast
aM
Denominator
coefficient
data type
Accumulator
data type
Cast
State
data type
0

IIR (all poles) direct form lattice AR

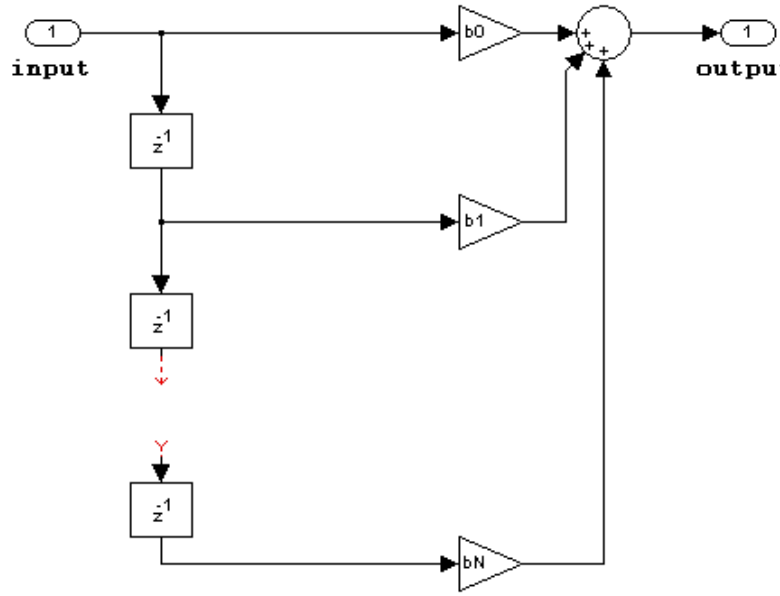


The following constraints are applicable when processing a fixed-point signal with this filter structure:

- Inputs and coefficients can be real or complex.
- Coefficients can be real or complex.

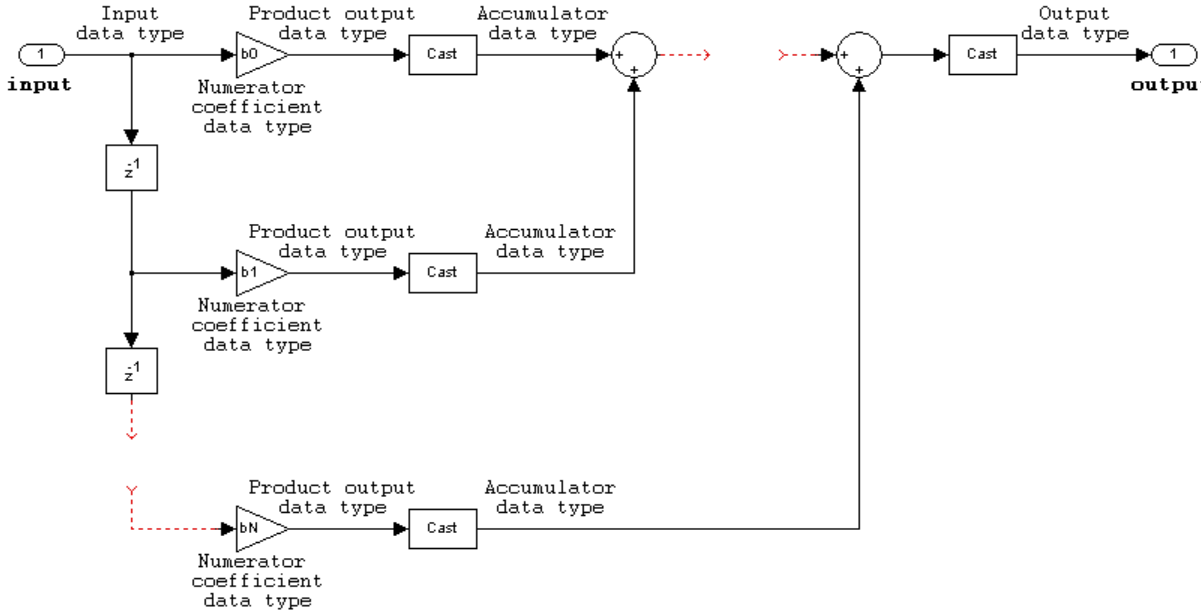


FIR (all zeros) direct form

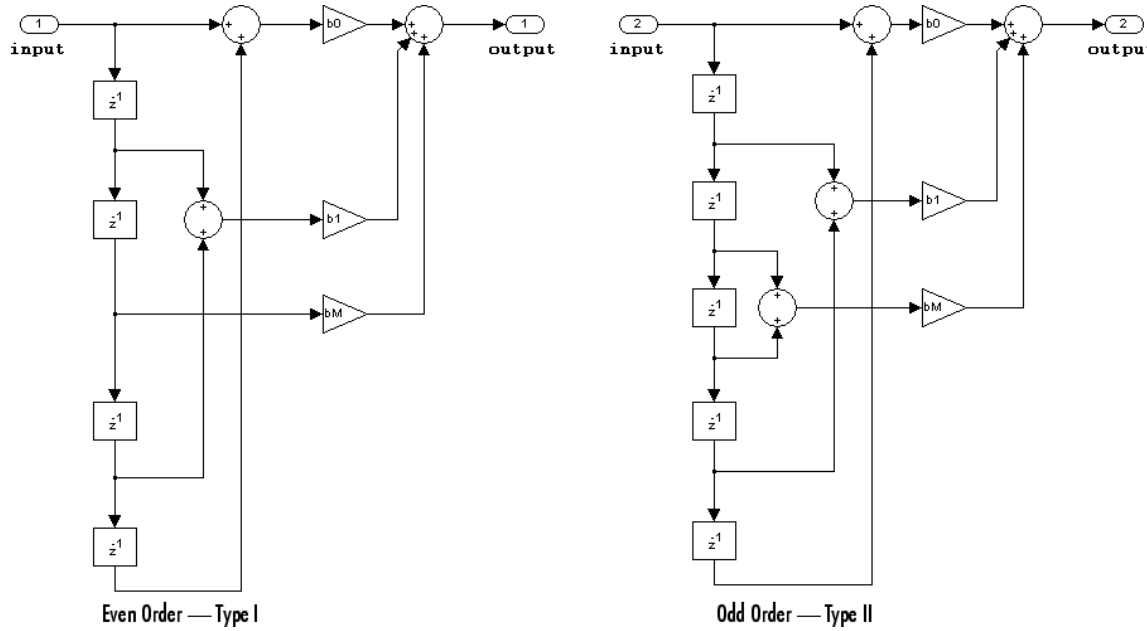


The following constraints are applicable when processing a fixed-point signal with this filter structure:

- Inputs can be real or complex.
- Numerator coefficients can be real or complex.
- You cannot specify the state data type on the block mask for this structure, because the input and output states have the same data types as the input.

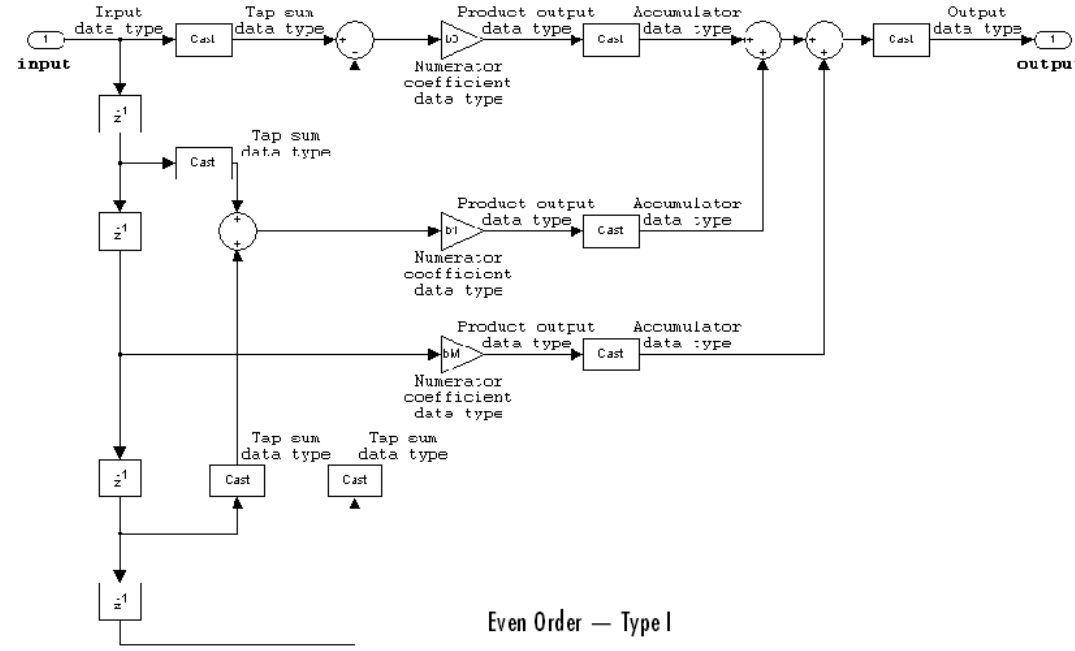

Input
data type
Product output
data type
Accumulator
data type
Output
data type
1
input
b0
Cast
Cast
1
output
Numerator
coefficient
data type
z-1
Product output
data type
Accumulator
data type
b1
Cast
Numerator
coefficient
data type
z-1
Product output
data type
Accumulator
data type
bN
Cast
Numerator
coefficient
data type

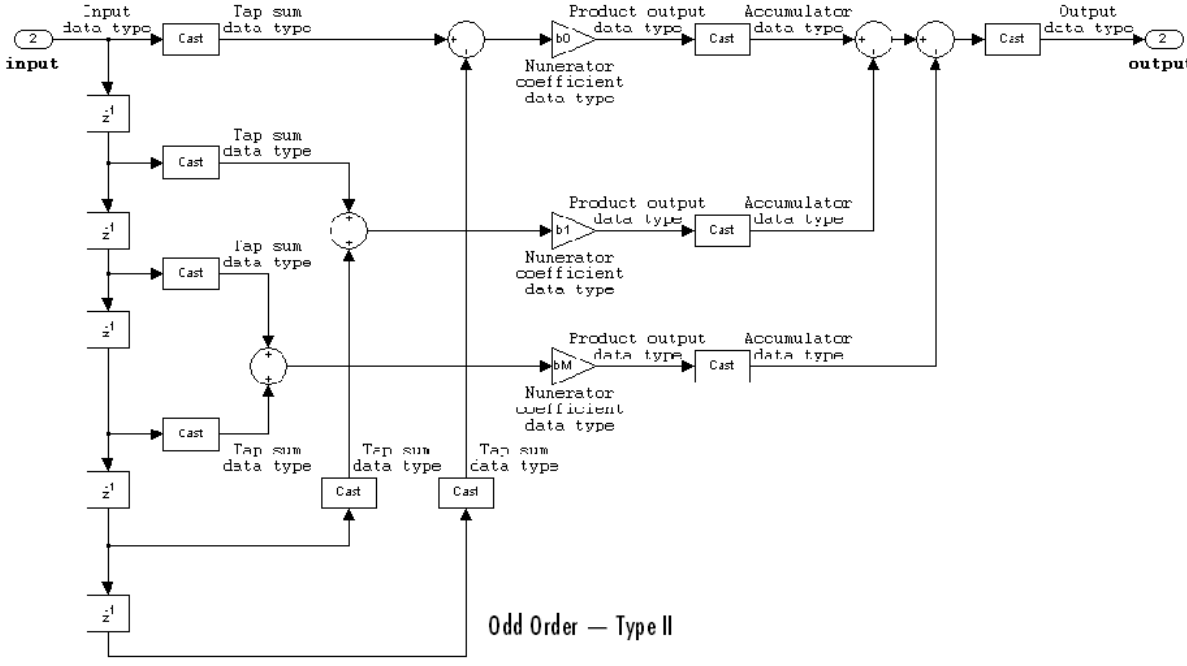
FIR (all zeros) direct form symmetric



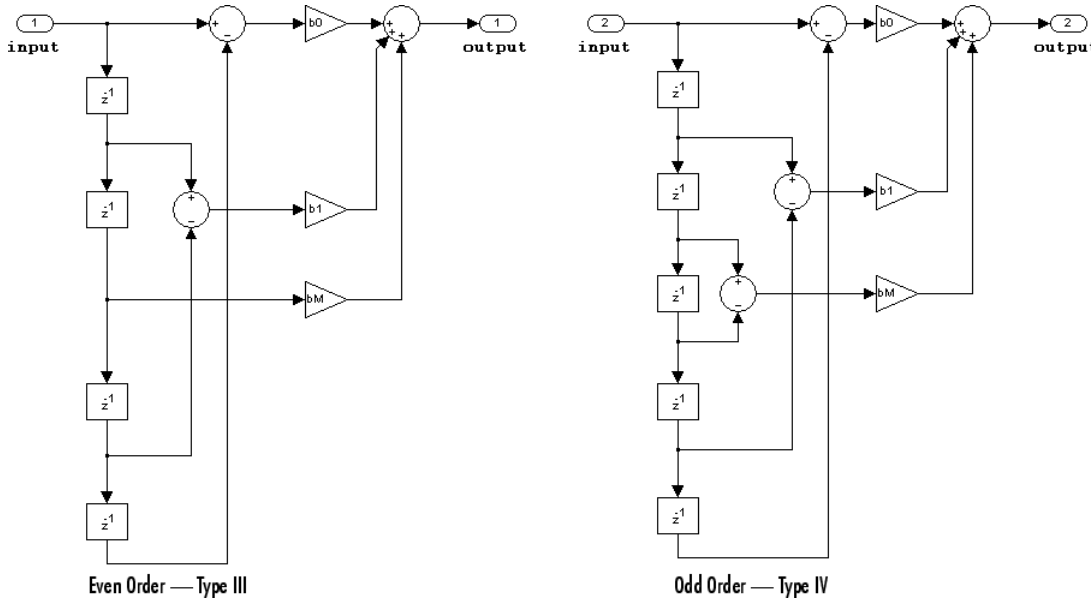
The following constraints are applicable when processing a fixed-point signal with this filter structure:

- Inputs can be real or complex.
- Numerator coefficients can be real or complex.
- You cannot specify the state data type on the block mask for this structure, because the input and output states have the same data types as the input.
- It is assumed that the filter coefficients are symmetric. Only the first half of the coefficients are used for filtering.
- The Tap Sum parameter determines the data type the filter uses when it sums the inputs prior to multiplication by the coefficients.


Input
data type
Tap sum
data type
Product output
data type
Accumulator
data type
Output
data type
Cast
input
output
Numerator
coefficient
data type
Even Order — Type I

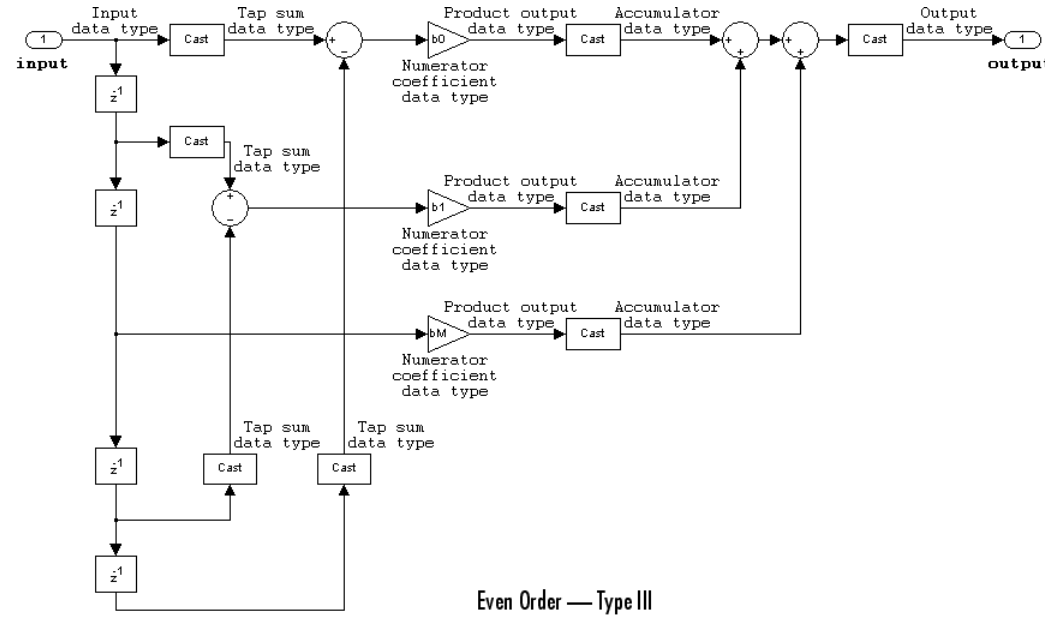

Input
data type
Tap sum
data type
Product output
data type
Accumulator
data type
Output
data type
Cast
input
output
Numerator
coefficient
data type
Odd Order — Type II

FIR (all zeros) direct form antisymmetric

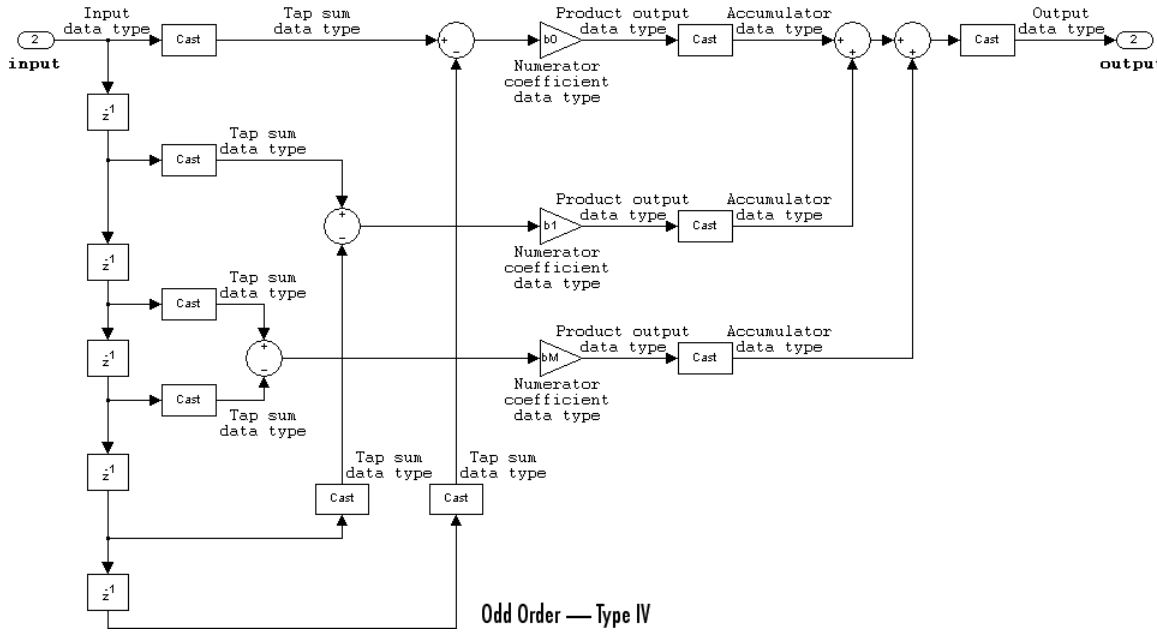


The following constraints are applicable when processing a fixed-point signal with this filter structure:

- Inputs can be real or complex.
- Numerator coefficients can be real or complex.
- You cannot specify the state data type on the block mask for this structure, because the input and output states have the same data types as the input.
- It is assumed that the filter coefficients are antisymmetric. Only the first half of the coefficients are used for filtering.
- The Tap Sum parameter determines the data type the filter uses when it sums the inputs prior to multiplication by the coefficients.

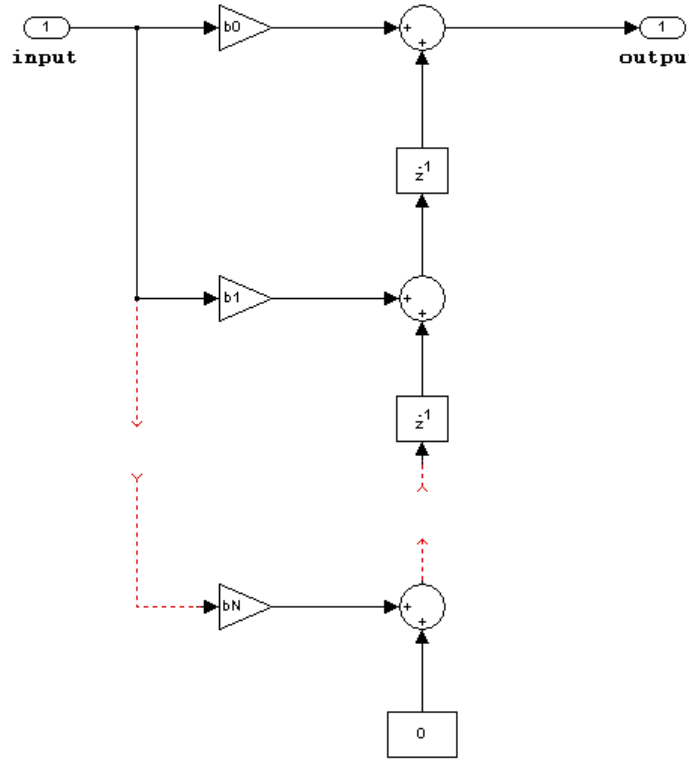


Even Order — Type III



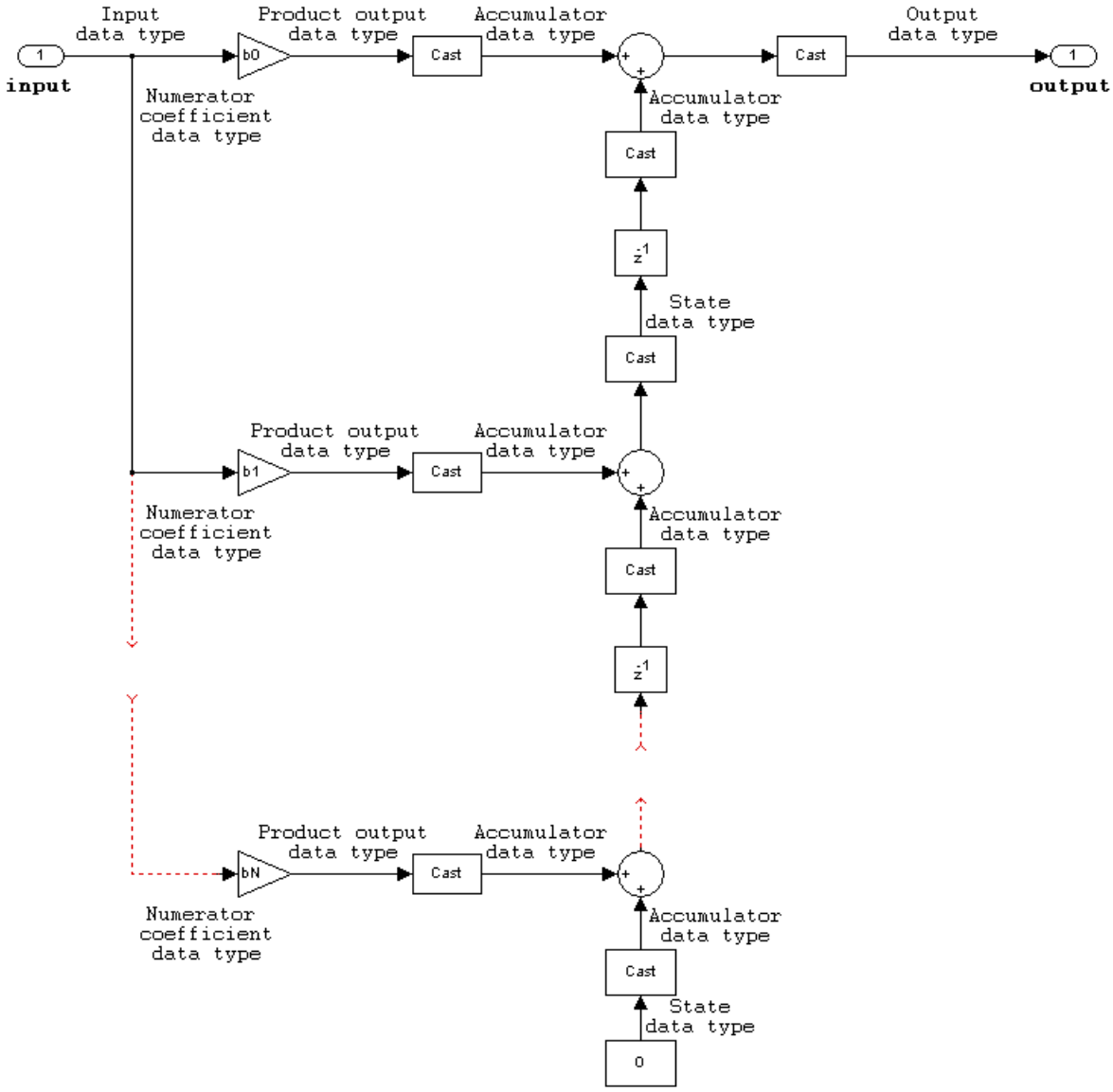
Odd Order — Type IV

FIR (all zeros) direct form transposed

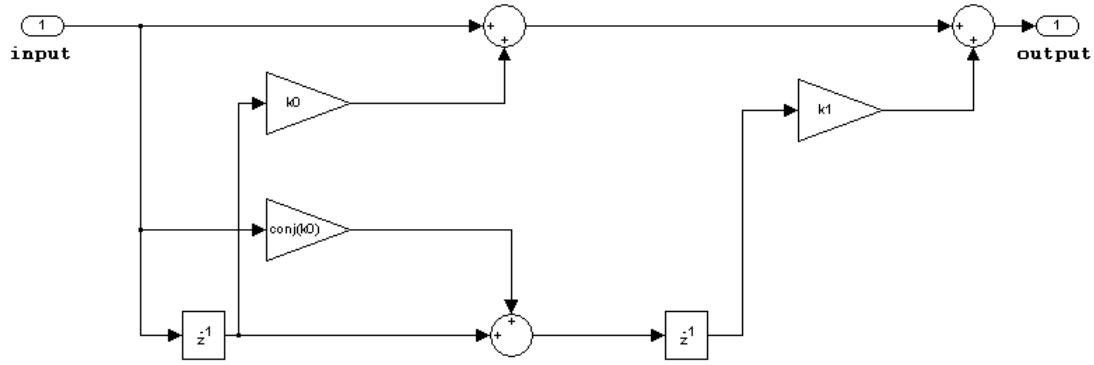


The following constraints are applicable when processing a fixed-point signal with this filter structure:

- Inputs can be real or complex.
- Coefficients can be real or complex.
- States are complex when either the inputs or the coefficients are complex.

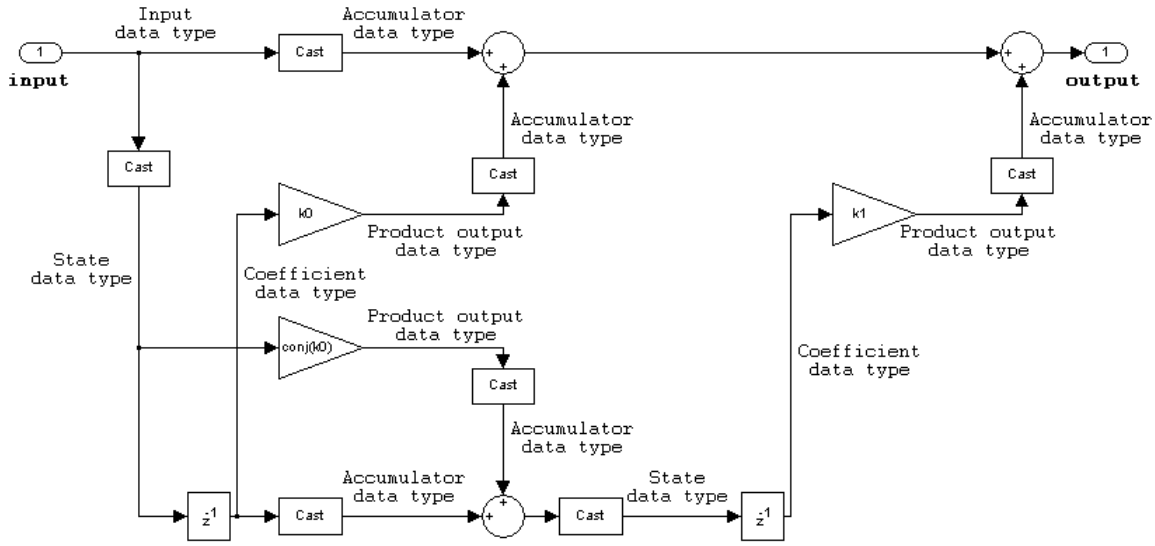

Input
data type
Product output
data type
Accumulator
data type
Output
data type
1
input
b0
Cast
Cast
1
output
Numerator
coefficient
data type
Accumulator
data type
Cast
z-1
State
data type
Cast
Product output
data type
Accumulator
data type
b1
Cast
Numerator
coefficient
data type
Accumulator
data type
Cast
z-1
Product output
data type
Accumulator
data type
bN
Cast
Numerator
coefficient
data type
Accumulator
data type
Cast
State
data type
0

FIR (all zeros) lattice MA



The following constraints are applicable when processing a fixed-point signal with this filter structure:

- Inputs and coefficients can be real or complex.
- Coefficients can be real or complex.



Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Fixed point (signed only)
- 8-, 16-, and 32-bit signed integers

See Also

Digital Filter Design	Signal Processing Blockset
Filter Realization Wizard	Signal Processing Blockset
dfilt	Signal Processing Toolbox
fdatool	Signal Processing Toolbox
fvtool	Signal Processing Toolbox
sptool	Signal Processing Toolbox

Purpose

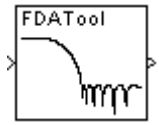
Design and implement digital FIR and IIR filters

Library

Filtering / Filter Designs

dsparch4

Description



Note Use this block to design, analyze, and then efficiently implement floating-point filters. The following blocks also implement digital filters, but serve slightly different purposes:

- Digital Filter — Use to efficiently implement floating-point or fixed-point filters that you have already designed. This block provides the same exact filter implementation as the Digital Filter Design block.
- Filter Realization Wizard — Use to implement floating-point or fixed-point filters built from Sum, Gain, and Unit Delay blocks. You can either design the filter within this block, or import the coefficients of a filter that you designed elsewhere.

The Digital Filter Design block implements a digital FIR or IIR filter that you design using the Filter Design and Analysis Tool (fdatool) GUI. This block provides the same exact filter implementation as the Digital Filter block.

The block applies the specified filter to each channel of a discrete-time input signal, and outputs the result. The outputs of the block numerically match the outputs of the Digital Filter block, the MATLAB filter function, and the Filter Design Toolbox filter function.

The sampling frequency, Fs, that you specify in the FDATool GUI should be identical to the sampling frequency of the Digital Filter Design block's input block. When the sampling frequencies of these blocks do not match, the Digital Filter Design block returns a warning message and inherits the sampling frequency of the input block.

Valid Inputs and Corresponding Outputs

The block accepts inputs that are sample-based or frame-based vectors and matrices. The block filters each input channel independently over time, where

- Each column of a frame-based vector or matrix is an independent channel.
- Each element of a sample-based vector or matrix is an independent channel.

The output has the same dimensions and frame status as the input.

Designing the Filter

Double-click the Digital Filter Design block to open FDATool. Use FDATool to design or import a digital FIR or IIR filter. To learn how to design filters with this block and FDATool, see the following topics:

- "Digital Filter Design Block"
- fdatool reference page in the Signal Processing Toolbox documentation

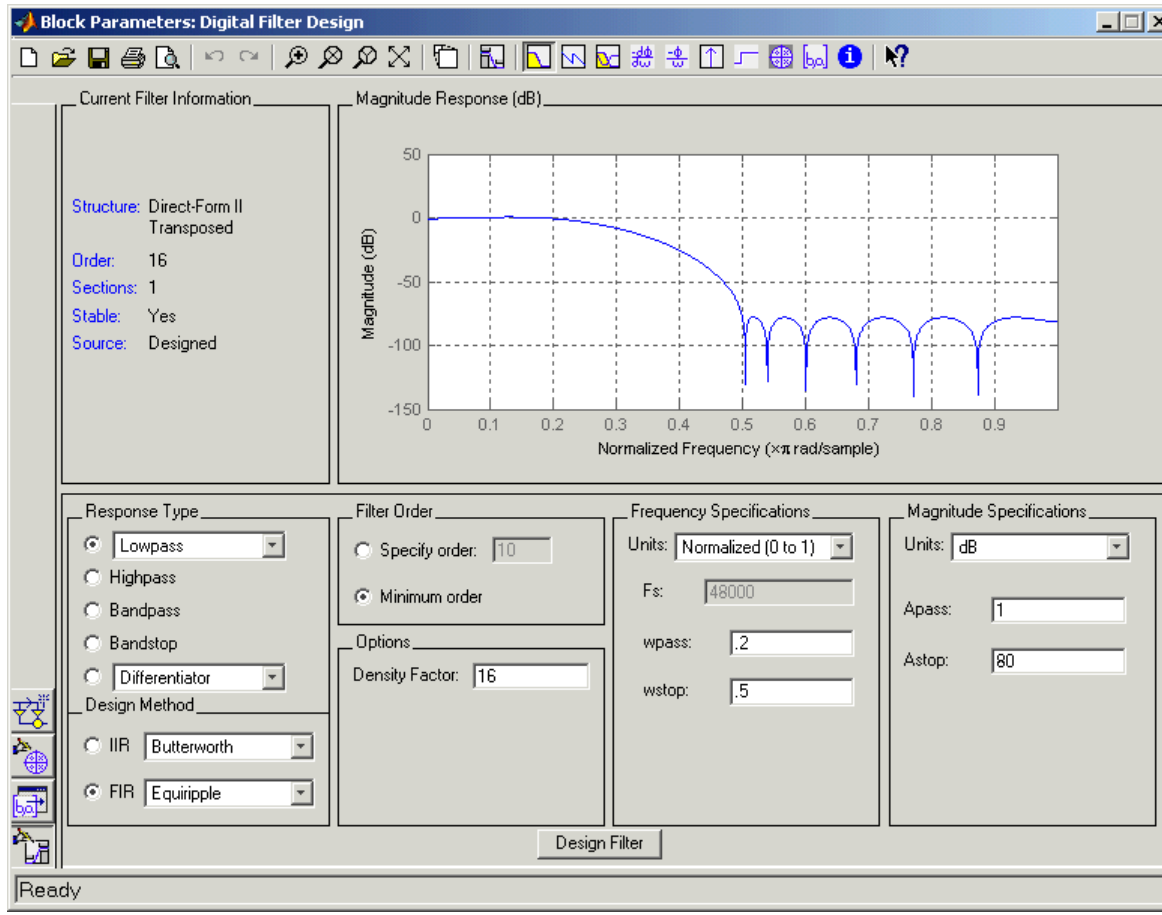
Tuning the Filter During Simulation

You can tune the filter specifications in FDATool during simulations as long as your changes do not modify the filter length or filter order. The block's filter updates as soon as you apply any filter changes in FDATool.

Examples

See the "Digital Filter Design Block" section in the Signal Processing Blockset User's Guide.

Dialog Box



The FDATool GUI Opened from the Digital Filter Design Block

To get the Transform Filter button , install the Filter Design Toolbox product. To get the Targets menu, install the Target Support Package™ TC6 product.

To learn how to use the FDATool GUI, see “Designing the Filter” on page 2-392.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Analog Filter Design	Signal Processing Blockset
Window Function	Signal Processing Blockset
fdatool	Signal Processing Toolbox
filter	Signal Processing Toolbox
fvtool	Signal Processing Toolbox
sptool	Signal Processing Toolbox
filter	Filter Design Toolbox

To learn how to use this block and FDATool, see the following:

- “Filters”
- “Digital Filter Design Block”
- fdatool reference page in the Signal Processing Toolbox documentation

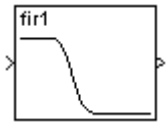
Purpose

Design and implement a variety of FIR filters

Library

dspobslib

Description



Note The Digital FIR Filter Design block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the Digital Filter block.

The Digital FIR Filter Design block designs a discrete-time (digital) FIR filter in one of several different band configurations using a window method. Most of these filters are designed using the Signal Processing Toolbox fir1 function, and are real with linear phase response. The block applies the filter to a discrete-time input using the Direct-Form II Transpose Filter (Obsolete) block.

An M-by-N sample-based matrix input is treated as M*N independent channels, and an M-by-N frame-based matrix input is treated as N independent channels. In both cases, the block filters each channel independently over time, and the output has the same size and frame status as the input.

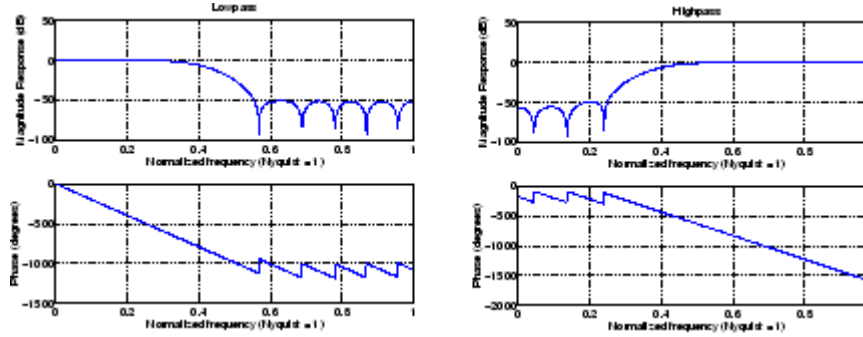
For complete details on the classical FIR filter design algorithm, see the description of the fir1 and fir2 functions in the Signal Processing Toolbox documentation.

Band Configurations

The band configuration for the filter is set from the Filter type pop-up menu. The band configuration parameters below this pop-up menu adapt appropriately to match the Filter type selection.

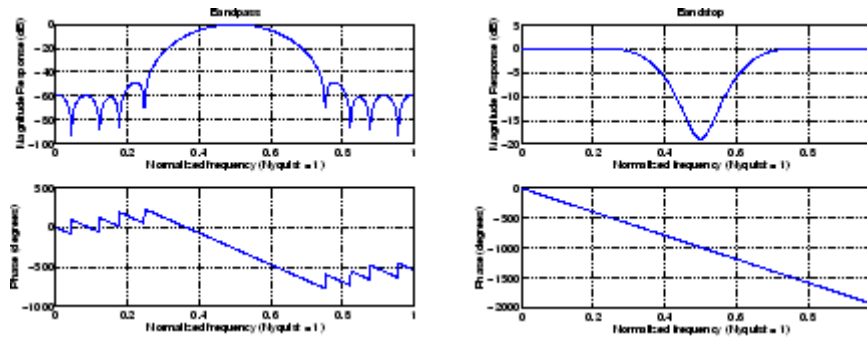
- Lowpass and Highpass

  In lowpass and highpass configurations, the Filter order and Cutoff frequency parameters specify the filter design. Frequencies are normalized to half the sample frequency. The figure below shows the frequency response of the default order-22 filter with cutoff at 0.4.



- Bandpass and Bandstop

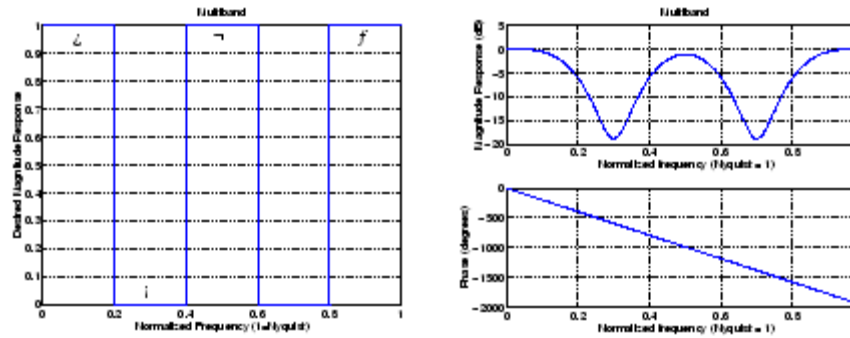
  In bandpass and bandstop con Þgurations, the Filter order , Lower cutoff frequency , and Upper cutoff frequency parameters specify the Þlter design. Frequencies are normalized to half the sample frequency, and the actual Þlter order is twice the Filter order parameter value. The Þgure below shows the frequency response of the default order-22 Þlter with lower cutoff at 0.4, and upper cutoff at 0.6.



- Multiband

  In the multiband con Þguration, the Filter order , Cutoff frequency vector , and Gain in the **Þrst band** parameters specify the Þlter design. The Cutoff frequency vector contains frequency points in the range 0 to 1, where 1 corresponds to half th e sample frequency. Frequency points must appear in ascending order. The Gain in the **Þrst band** parameter speci Þes the gain in the Þrst band: 0 indicates a stopband, and 1 indicates a passband. Additional bands alternate between passband and stopband. The Þgure below shows the frequency response of the default order-22 Þlter with Þve bands, the Þrst a passband.

  Cutoff frequency vector = [0.2 0.4 0.6 0.8]



- Arbitrary shape

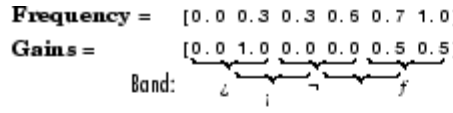
  In the arbitrary shape con Þguration, the Filter order , Frequency vector , and Gains at these frequencies parameters specify the Þlter design. The Frequency vector , fn , contains frequency points in the range 0 to 1 (inclusive) in ascending order, where 1 corresponds to half the sample frequency. The Gains at these frequencies parameter, mn, is a vector containing the desired magnitude response at the corresponding points in the Frequency vector . (Note that the speciÞcations for the Arbitrary shape conÞguration are similar to those for the Yule-Walker IIR Filter Design block. Arbitrary-shape

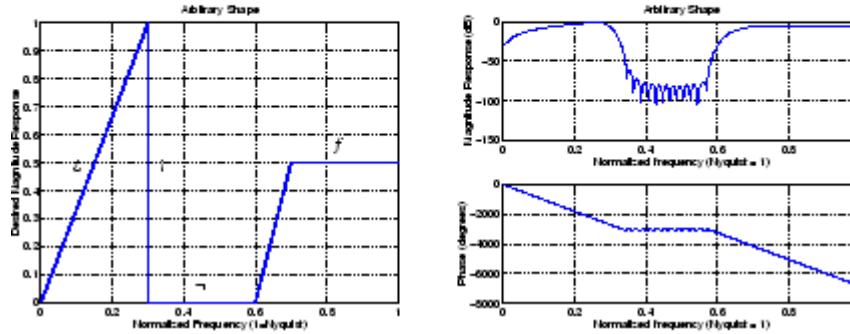
Þlters are designed using the Signal Processing Toolbox fir2 function.)

The desired magnitude response of the design can be displayed by typing

```
plot(fn,mn)
```

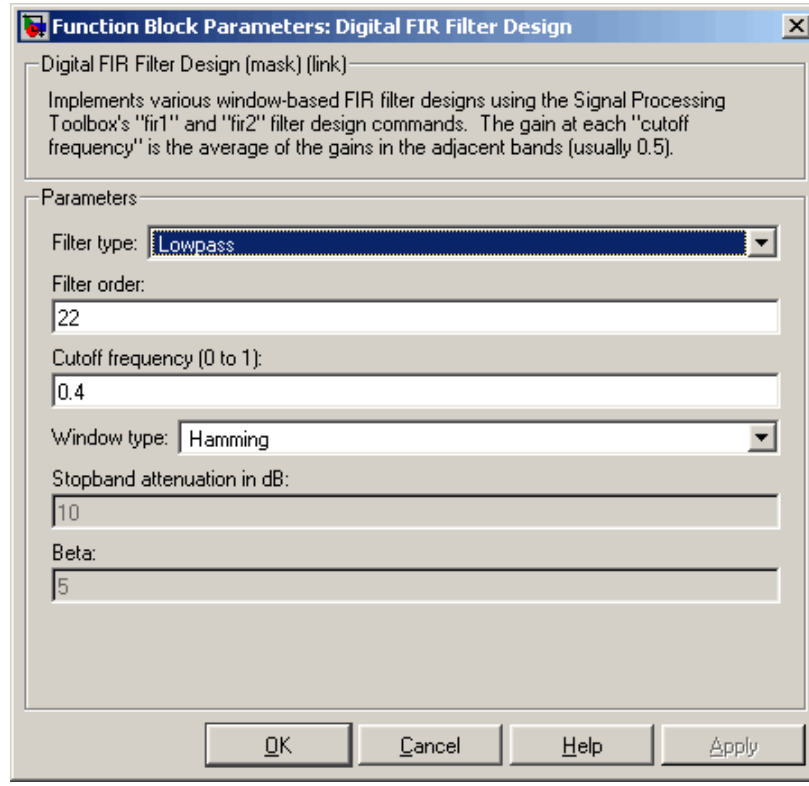
Duplicate frequencies can be use d to specify a step in the response (such as band 2 below). The Þgure shows an order-100 Þlter with Þve bands.





The Window type parameter allows you to select from a variety of different windows. See the Window Function block reference for a complete description of the available options.

Dialog Box



The parameters displayed in the dialog box vary for different design/band combinations. Only some of the parameters listed below are visible in the dialog box at any one time.

Filter type

The type of filter to design: Lowpass, Highpass, Bandpass, Bandstop, Multiband, or Arbitrary Shape. Tunable.

Filter order

The order of the filter. The filter length is one more than this value. For the Bandpass and Bandstop configurations, the order of the final filter is twice this value.

Cutoff frequency

The normalized cutoff frequency for the Highpass and Lowpass filter configurations. A value of 1 specifies half the sample frequency. Tunable.

Lower cutoff frequency

The lower passband or stopband frequency for the Bandpass and Bandstop filter configurations. A value of 1 specifies half the sample frequency. Tunable.

Upper cutoff frequency

The upper passband or stopband frequency for the Bandpass and Bandstop filter configurations. A value of 1 specifies half the sample frequency. Tunable.

Cutoff frequency vector

A vector of ascending frequency points defining the cutoff edges for the Multiband filter. A value of 1 specifies half the sample frequency. Tunable.

Gain in the first band

The gain in the first band of the Multiband filter: 0 specifies a stopband, 1 specifies a passband. Additional bands alternate between passband and stopband. Tunable.

Frequency vector

A vector of ascending frequency points defining the frequency bands of the Arbitrary shape filter. The frequency range is 0 to 1 including the endpoints, where 1 corresponds to half the sample frequency. Tunable.

Gains at these frequencies

A vector containing the desired magnitude response for the Arbitrary shape filter at the corresponding points in the Frequency vector . Tunable.

Window type

The type of window to apply. See the Window Function block reference. Tunable.

Stopband ripple

The level (dB) of stopband ripple, R_s, for the Chebyshev window. Tunable.

Beta

The Kaiser window ß parameter. Increasing Beta widens the mainlobe and decreases the amplitude of the window sidelobes in the window's frequency magnitude response. Tunable.

References

Antoniou, A. Digital Filters: Analysis, Design, and Applications . 2nd ed. New York, NY: McGraw-Hill, 1993.

Oppenheim, A. V. and R. W. Schafer. Discrete-Time Signal Processing . Englewood Cliffs, NJ: Prentice Hall, 1989.

Proakis, J. and D. Manolakis. Digital Signal Processing. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

Digital FIR Raised Cosine Filter Design (Obsolete)

Purpose

Design and implement raised cosine FIR filter

Library

dspobslib

Description

Note The Digital FIR Raised Cosine Filter Design block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the Digital Filter block.

The Digital FIR Raised Cosine Filter Design block uses the Signal Processing Toolbox firrcos function to design a lowpass, linear-phase, digital FIR filter with a raised cosine transition band. The block applies the filter to a discrete-time input using the Direct-Form II Transpose Filter (Obsolete) block.

An M-by-N sample-based matrix input is treated as M*N independent channels, and an M-by-N frame-based matrix input is treated as N independent channels. In both cases, the block filters each channel independently over time, and the output has the same size and frame status as the input.

The frequency response of the raised cosine filter is

$$H(f) = \begin{cases} \dfrac{1}{2f_{n0}} & 0 \le |f| \le (1-R)f_{n0} \\ \left[\dfrac{1+\cos\dfrac{\pi}{2Rf_{n0}}|f|-(1-R)f_{n0}}{4f_{n0}}\right] & (1-R)f_{n0} \le |f| \le (1+R)f_{n0} \\ 0 & (1+R)f_{n0} \le |f| \le 1 \end{cases}$$

where H(f) is the magnitude response at frequency f, f_{n0} is the normalized cutoff frequency (-6 dB) specified by the Upper cutoff frequency parameter, and R is a rolloff factor in the range [0,1] determining the passband-to-stopband transition width.

The Square-root raised cosine filter option designs a filter with magnitude response $\sqrt{H(f)}$. This is useful when the filter is part of a pair of matched filters.

When the Design method parameter is set to Rolloff factor, the secondary Rolloff factor parameter is enabled, and R can be directly specified. When Design method is set to Transition bandwidth, the secondary Transition bandwidth parameter is enabled, and the transition region bandwidth, Δf, can be specified in place of R. The transition region is centered on f_{n0} and must be sufficiently narrow to satisfy

$$0 < \left(f_{n0} \pm \frac{\Delta f}{2} \right) < 1$$

The Upper cutoff frequency and Transition bandwidth parameter values are normalized to half the sample frequency.

The Window type parameter allows you to apply a variety of different windows to the raised cosine filter. See the Window Function block reference for a complete description of the available options.

Algorithm

The filter output is computed by convolving the input with a truncated, delayed, windowed version of the filter's impulse response. The impulse response for the raised cosine filter is

$$h(kT_s) = \frac{1}{F_s} \operatorname{sinc}(2\pi k T_s f_{n0}) \frac{\cos(2\pi R k T_s f_{n0})}{1 - (4 R k T_s f_{n0})^2} \quad -\infty < k < \infty$$

which has limits

$$h(0) = \frac{1}{F_s}$$

and

$$h\left(\pm\frac{1}{4Rf_{n0}}\right) = \frac{R}{2F_s}\sin\left(\frac{\pi}{2R}\right)$$

The impulse response for the square-root raised cosine filter is

$$h(kT_s) = \frac{4R\cos((1+R)2\pi kT_s f_{n0}) + \dfrac{\sin((1-R)2\pi kT_s f_{n0})}{8RkT_s f_{n0}}}{\pi F_s\sqrt{\dfrac{1}{2f_{n0}}}((8RkT_s f_{n0})^2 - 1)} \quad -\infty < k < \infty$$

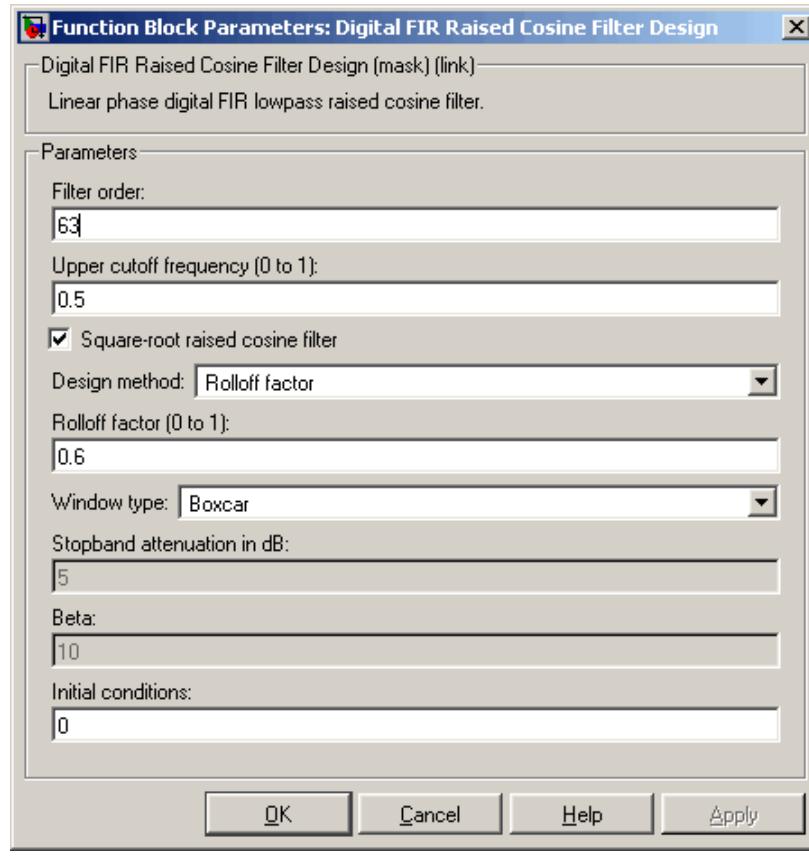
which has limits

$$h(0) = (-4R - \pi + \pi R)\frac{\sqrt{2f_{n0}}}{\pi F_s}$$

and

$$\left(\pm\frac{1}{8Rf_{n0}}\right) = \frac{\sqrt{2f_{n0}}}{2\pi F_s}\left[\pi(1+R)\sin\left(\frac{\pi(1+R)}{4R}\right) - 4R\sin\left(\frac{\pi(R-1)}{4R}\right) + \pi(R-1)\cos\left(\frac{\pi(R-1)}{4R}\right)\right]$$

Dialog Box



Filter order

The order of the filter. The filter length is one more than this value.

Upper cutoff frequency

The normalized cutoff frequency, f_{n0}. A value of 1 specifies half the sample frequency. Tunable.

Square-root raised cosine filter
: Selects the square-root filter option, which designs a filter with magnitude response $\sqrt{H(f)}$. Tunable.

Design method
: The method used to design the transition region of the filter, Rolloff factor or Transition bandwidth. Tunable.

Rolloff factor
: The rolloff factor, R, enabled when Rolloff factor is selected in the Design method parameter. Tunable.

Transition bandwidth
: The transition bandwidth, Δf, enabled when Transition bandwidth is selected in the Design method parameter. Tunable.

Window type
: The type of window to apply. See the Window Function block reference. Tunable.

Stopband attenuation in dB
: The level (dB) of stopband attenuation, R_s, for the Chebyshev window. Tunable.

Beta
: The Kaiser window β parameter. Increasing β widens the mainlobe and decreases the amplitude of the window sidelobes in the window's frequency magnitude response. Tunable.

Initial conditions
: The filter's initial conditions, a scalar, vector, or matrix. See the Direct-Form II Transpose Filter (Obsolete) block reference for complete syntax information.

References

Proakis, J. G. Digital Communications. Third ed. New York, NY: McGraw-Hill, 1995.

Proakis, J. G. and M. Salehi. Contemporary Communication Systems Using MATLAB. Boston, MA: PWS Publishing, 1998.

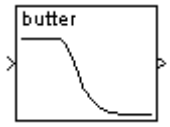
Purpose

Design and implement IIR filter

Library

dspobslib

Description



Note The Digital IIR Filter Design block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the Digital Filter block.

The Digital IIR Filter Design block designs a discrete-time (digital) IIR filter in a lowpass, highpass, bandpass, or bandstop configuration, and applies it to the input using the Direct-Form II Transpose Filter (Obsolete) block.

An M-by-N sample-based matrix input is treated as M*N independent channels, and an M-by-N frame-based matrix input is treated as N independent channels. In both cases, the block filters each channel independently over time, and the output has the same size and frame status as the input.

The Design method parameter allows you to specify Butterworth, Chebyshev type I, Chebyshev type II, and elliptic filter designs. Note that for the bandpass and bandstop configurations, the actual filter length is twice the Filter order parameter value.

Filter Design	Description
Butterworth	The magnitude response of a Butterworth filter is maximally flat in the passband and monotonic overall.
Chebyshev type I	The magnitude response of a Chebyshev type I filter is equiripple in the passband and monotonic in the stopband.

Filter Design	Description
Chebyshev type II	The magnitude response of a Chebyshev type II filter is monotonic in the passband and equiripple in the stopband.
Elliptic	The magnitude response of an elliptic filter is equiripple in both the passband and the stopband.

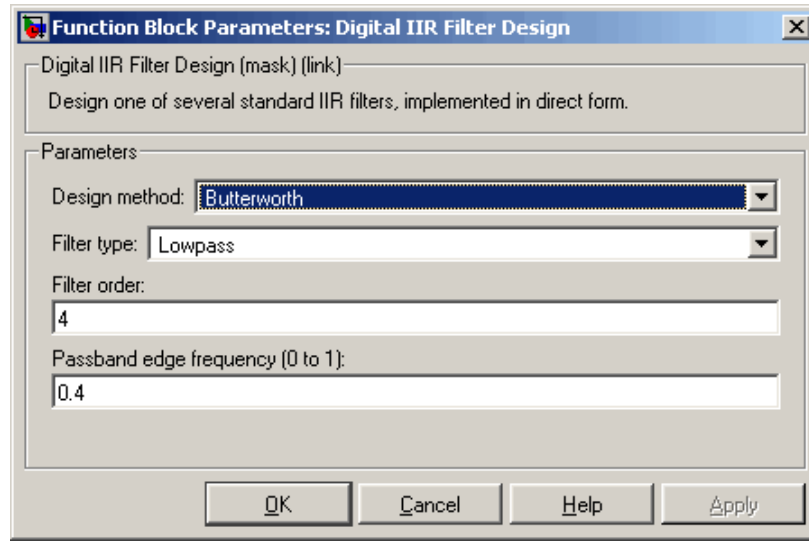
The design and band configuration of the filter are selected from the Design method and Filter type pop-up menus in the dialog box. For each combination of design method and band configuration, an appropriate set of secondary parameters is displayed.

The table below lists the available parameters for each design/band combination. For lowpass and highpass band configurations, these parameters include the passband edge frequency f_{np}, the stopband edge frequency f_{ns}, the passband ripple R_p, and the stopband attenuation R_s. For bandpass and bandstop configurations, the parameters include the lower and upper passband edge frequencies, f_{np1} and f_{np2}, the lower and upper stopband edge frequencies, f_{ns1} and f_{ns2}, the passband ripple R_p, and the stopband attenuation R_s. Frequency values are normalized to half the sample frequency, and ripple and attenuation values are in dB.

	Lowpass	Highpass	Bandpass	Bandstop
Butterworth	Order, f_{np}	Order, f_{np}	Order, f_{np1}, f_{np2}	Order, f_{np1}, f_{np2}
Chebyshev Type I	Order, f_{np}, R_p	Order, f_{np}, R_p	Order, f_{np1}, f_{np2}, R_p	Order, f_{np1}, f_{np2}, R_p
Chebyshev Type II	Order, f_{ns}, R_s	Order, f_{ns}, R_s	Order, f_{ns1}, f_{ns2}, R_s	Order, f_{ns1}, f_{ns2}, R_s
Elliptic	Order, f_{np}, R_p, R_s	Order, f_{np}, R_p, R_s	Order, f_{np1}, f_{np2}, R_p, R_s	Order, f_{np1}, f_{np2}, R_p, R_s

The digital filters are designed using Signal Processing Toolbox software's filter design commands butter , cheby1 , cheby2 , and ellip .

Dialog Box



The parameters displayed in the dialog box vary for different design/band combinations. Only some of the parameters listed below are visible in the dialog box at any one time.

Design method

The Þlter design method: Butterworth , Chebyshev type I , Chebyshev type II , or Elliptic . Tunable.

Filter type

The type of Þlter to design: Lowpass , Highpass , Bandpass , or Bandstop . Tunable.

Filter order

The order of the Þlter for lowpass and highpass con Þgurations. For bandpass and bandstop con Þgurations, the length of the Þnal Þlter is twice this value.

Passband edge frequency

The normalized passband edge frequency for the highpass and lowpass conÞgurations of the Butterworth, Chebyshev type I, and elliptic designs. Tunable.

Lower passband edge frequency
: The normalized lower passband frequency for the bandpass and bandstop configurations of the Butterworth, Chebyshev type I, and elliptic designs. Tunable.

Upper passband edge frequency
: The normalized upper passband frequency for the bandpass and bandstop configurations of the Butterworth, Chebyshev type I, or elliptic designs. Tunable.

Stopband edge frequency
: The normalized stopband edge frequency for the highpass and lowpass band configurations of the Chebyshev type II design. Tunable.

Lower stopband edge frequency
: The normalized lower stopband frequency for the bandpass and bandstop configurations of the Chebyshev type II design. Tunable.

Upper stopband edge frequency
: The normalized upper stopband frequency for the bandpass and bandstop filter configurations of the Chebyshev type II design. Tunable.

Passband ripple in dB
: The passband ripple, in dB, for the Chebyshev type I and elliptic designs. Tunable.

Stopband attenuation in dB
: The stopband attenuation, in dB, for the Chebyshev type II and elliptic designs. Tunable.

References

Antoniou, A. Digital Filters: Analysis, Design, and Applications . 2nd ed. New York, NY: McGraw-Hill, 1993.

Oppenheim, A. V. and R. W. Schafer. Discrete-Time Signal Processing . Englewood Cliffs, NJ: Prentice Hall, 1989.

Proakis, J. and D. Manolakis. Digital Signal Processing. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

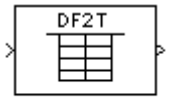
Purpose

Apply IIR filter to input

Library

dspobslib

Description



Direct-Form II
Transpose Filter

Note This block is now just an implementation of the Digital Filter block.

Discrete Impulse

Purpose

Generate discrete impulse

Library

Signal Processing Sources

dspsrcs4

Description

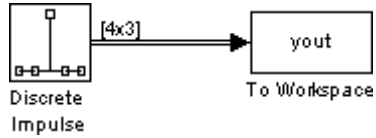
The Discrete Impulse block generates an impulse (the value 1) at output sample D+1, where D is specified by the Delay parameter ($D \geq 0$). All output samples preceding and following sample D+1 are zero.

When D is a length- N vector, the block generates an M-by-N matrix output representing N distinct channels, where frame size M is specified by the Samples per frame parameter. The impulse for the i th channel appears at sample D(i)+1 . For M=1, the output is sample based; otherwise, the output is frame based.

The Sample time parameter value, T_s, specifies the output signal sample period. The resulting frame period is $M*T_s$.

Examples

Construct the model below.



Configure the Discrete Impulse block to generate a frame-based three-channel output of type double , with impulses at samples 1, 4, and 6 of channels 1, 2, and 3, respectively. Use a sample period of 0.25 and a frame size of 4. The corresponding settings should be as follows:

- Delay = [0 3 5]
- Sample time = 0.25
- Samples per frame = 4
- Output data type = double

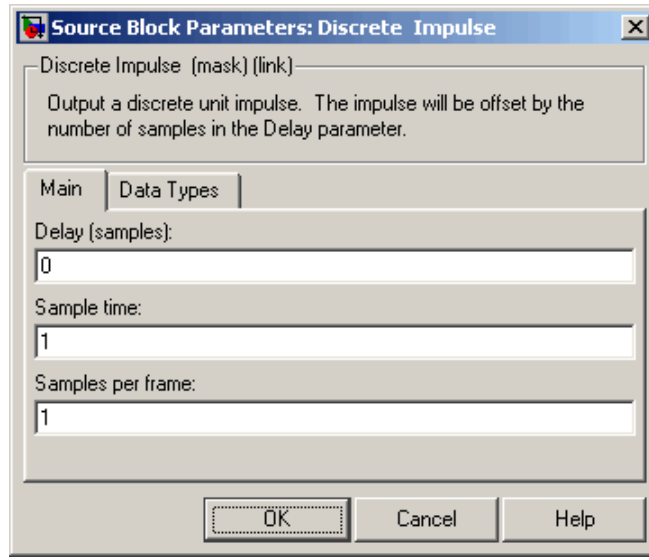
Run the model and look at the output, yout . The first few samples of each channel are shown below.

```
yout(1:10,:)
ans =
     1     0     0
     0     0     0
     0     0     0
     0     1     0
     0     0     0
     0     0     1
     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

The block generates an impulse at sample 1 of channel 1 (first column), at sample 4 of channel 2 (second column), and at sample 6 of channel 3 (third column).

Dialog Box

The Main pane of the Discrete Impulse block dialog appears as follows.

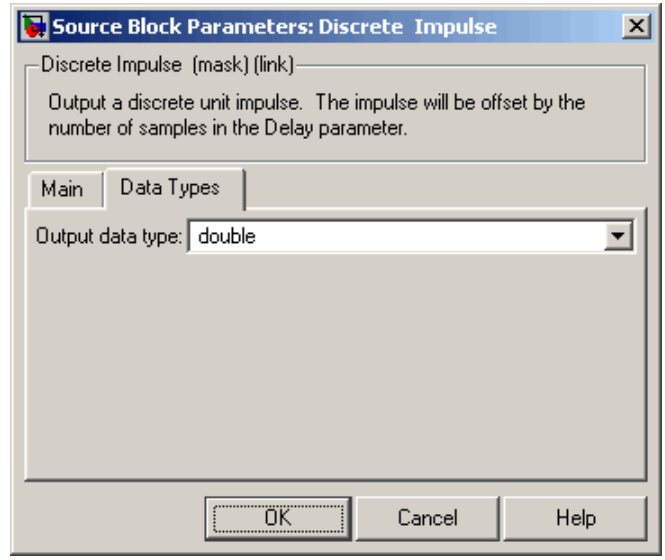


Delay
: The number of zero-valued output samples, D, preceding the impulse. A length- N vector specifies an N-channel output.

Sample time
: The sample period, T_s, of the output signal. The output frame period is $M*T_s$.

Samples per frame
: The number of samples, M, in each output frame. When the value of this parameter is 1, the block outputs a sample-based signal.

The Data Types pane of the Discrete Impulse block dialog appears as follows.



Output data type

Specify the output data type in one of the following ways:

- Choose one of the built-in data types from the list.
- Choose Fixed-point to specify the output data type and scaling in the Signed, Word length, Set fraction length in output to, and Fraction length parameters.
- Choose User-defined to specify the output data type and scaling in the User-defined data type, Set fraction length in output to, and Fraction length parameters.
- Choose Inherit via back propagation to set the output data type and scaling to match the next block downstream.

Signed

Select to output a signed fixed-point signal. Otherwise, the signal is unsigned. This parameter is only visible when you select Fixed-point for the Output data type parameter.

Word length

Specify the word length, in bits, of the fixed-point output data type. This parameter is only visible when you select Fixed-point for the Output data type parameter.

User-defined data type

Specify any built-in or fixed-point data type. You can specify fixed-point data types using the following Simulink Fixed Point functions: sfix, ufix, sint, uint, sfrac, and ufrac. This parameter is only visible when you select User-defined for the Output data type parameter.

Set fraction length in output to

Specify the scaling of the fixed-point output by either of the following two methods:

- Choose Best precision to have the output scaling automatically set such that the output signal has the best possible precision.
- Choose User-defined to specify the output scaling in the Fraction length parameter.

This parameter is only visible when you select Fixed-point for the Output data type parameter, or when you select User-defined and the specified output data type is a fixed-point data type.

Fraction length

For fixed-point output data types, specify the number of fractional bits, or bits to the right of the binary point. This parameter is only visible when you select Fixed-point or User-defined for the Output data type parameter and User-defined for the Set fraction length in output to parameter.

Supported Data Types

- Double-precision floating point
- Single-precision floating point
- Fixed point

- Boolean
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

See Also

Data Type Conversion	Simulink
Constant	Simulink
Multiphase Clock	Signal Processing Blockset
N-Sample Enable	Signal Processing Blockset
Signal From Workspace	Signal Processing Blockset
impz	Signal Processing Toolbox

Display

Purpose

Show value of input

Library

Signal Processing Sinks

dspsnks4

Description

The Display block is an implementation of the Simulink Display block. See Display for more information.

Purpose

Resample input at lower rate by deleting samples

Library

Signal Operations

dspsigops

Description

The Downsample block decreases the sampling rate of the input by deleting samples. When the input is a frame-based M_i-by-N matrix, the block resamples each channel of the input independently. When the input is a sample-based N-D array, the Downsample block resamples the input array across time. The resample rate is K times lower than the input sample rate, where K is an integer specified by the Downsample factor parameter. The Downsample block resamples the input by discarding K–1 consecutive samples following each sample that is passed through to the output.

The Sample offset parameter delays the output samples by an integer number of sample periods, D, where $0 \le D < (K-1)$, so that any of the K possible output phases can be selected. For example, when you downsample the sequence 1, 2, 3, ... by a factor of 4, you can select from the following four phases.

Input Sequence	Sample Offset, D	Output Sequence (K=4)
1,2,3,...	0	1,5,9,13,17,21,25,29,...
1,2,3,...	1	0,2,6,10,14,18,22,26,...
1,2,3,...	2	0,3,7,11,15,19,23,27,...
1,2,3,...	3	0,4,8,12,16,20,24,28,...

The initial zero in each of the latter three output sequences in the table is a result of the default zero Initial condition parameter setting for this example. See “Latency” on page 2-422 for more information on the Initial condition parameter.

This block supports triggered subsystems if, for Sample-based mode, you select Force single-rate and, for Frame-based mode, you select Maintain input frame rate.

Sample-Based Operation

When the input is a sample-based N-D array, the Downsample block resamples the input across time. The block downsamples the input array by discarding K–1 samples following each sample that it passes through to the output. The input and output sizes of the Downsample block are identical.

The Sample-based mode parameter determines how the block represents the new rate at the output. There are two available options:

- Allow multirate

  When you select Allow multirate, the sample period of the sample-based output is K times longer than the input sample period ($T_{so} = KT_{si}$). The block is therefore multirate.

- Force single rate

  When you select Force single rate, the block forces the output sample rate to match the input sample rate ($T_{so} = T_{si}$) by repeating every Kth input sample K times at the output. The block is therefore single rate. (The block's operation when you select Force single rate is similar to the operation of a Sample and Hold block with a repeating trigger event of period KT_{si}.)

The setting of the Frame-based mode parameter does not affect sample-based inputs.

Frame-Based Inputs

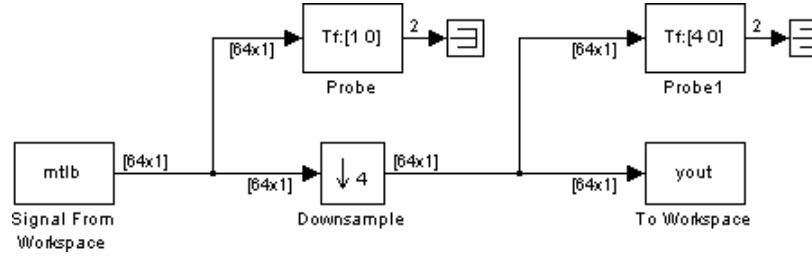
When the input is frame based, the block treats each of the N input columns as a frame containing M_i sequential time samples from an independent channel. The block downsamples each channel independently by discarding K–1 rows of the input matrix following each row that it passes through to the output.

The Frame-based mode parameter determines how the block adjusts the rate at the output to accommodate the reduced number of samples. There are two available options:

- Maintain input frame size

  The block generates the output at the slower (downsampled) rate by using a proportionally longer frame period at the output port than at the input port. For downsampling by a factor of K, the output frame period is K times longer than the input frame period ($T_{fo} = KT_{fi}$), but the input and output frame sizes are equal.

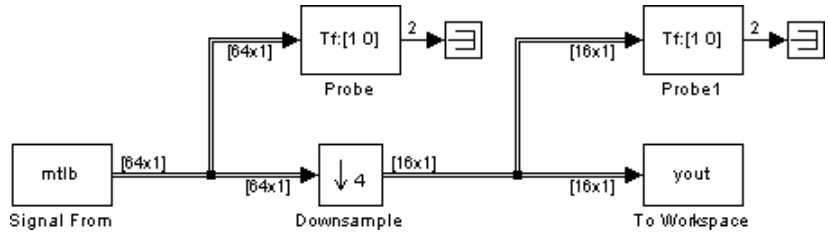
  The following model shows a single-channel input with a frame period of 1 second being downsampled by a factor of 4 to a frame period of 4 seconds. The input and output frame sizes are identical.



- Maintain input frame rate

  The block generates the output at the slower (downsampled) rate using a proportionally smaller frame size than the input. For downsampling by a factor of K, the output frame size is K times smaller than the input frame size ($M_o = M_i/K$), but the input and output frame rates are equal.

  The following model shows a single-channel input of frame size 64 being downsampled by a factor of 4 to a frame size of 16. The input and output frame rates are identical.



The setting of the Sample-based mode pop-up menu does not affect frame-based inputs.

Latency

The Downsample block has zero-tasking latency in the following cases:

- The Downsample factor parameter, K, is 1
- The block input is a frame-based signal, and the Maintain input frame rate parameter is selected
- The block input is a sample-based signal, and Sample offset parameter, D, is 0

Zero-tasking latency means that the block propagates input sample D+1 (received at t=0) as the first output sample, followed by input sample D+1+K, input sample D+1+2K, and so on. The Initial condition parameter value is not used.

In all other cases, the latency is nonzero:

- For sample-based input, the latency is one sample.
- For frame-based input, the latency is one frame.

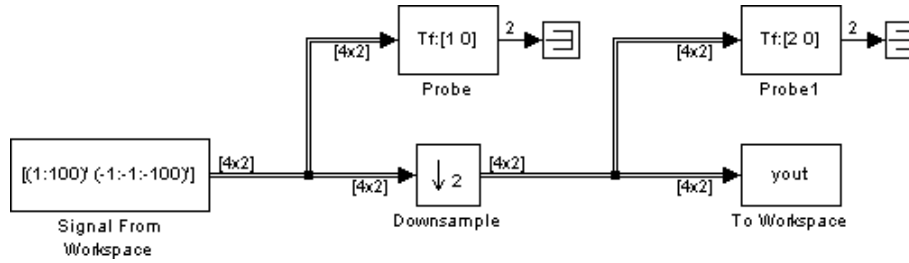
In all cases of one-sample latency, the initial condition for each channel appears as the first output sample. Input sample D+1 appears as the second output sample for each channel, followed by input sample D+1+K, input sample D+1+2K, and so on. The Initial condition parameter can be an array of the same size as the input, or a scalar to be applied to all signal channels.

In all cases of one-frame latency, the M_i rows of the initial condition matrix appear in sequence as the first M_i output rows. Input sample D+1 (i.e, row D+1 of the input matrix) appears in the output as sample M_i+1, followed by input sample D+1+K, input sample D+1+2K, and so on. The Initial condition value can be an M_i-by-N matrix containing one value for each channel, or a scalar to be repeated across all elements of the M_i-by-N matrix. See the following example for an illustration of this case.

Note For more information on latency and the Simulink tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Models with Multiple Sample Rates” in the Real-Time Workshop User’s Guide .

Examples

Construct this frame-based model.



Adjust the block parameters as follows:

- Configure the Signal From Workspace block to generate a two-channel signal with frame size of 4 and sample period of 0.25 second. This represents an output f rame period of 1 second (0.25*4). The first channel should contain the positive ramp signal 1, 2, ..., 100, and the second channel should contain the negative ramp signal –1, –2, ..., –100. The settings are:
  - Signal = [(1:100)' (-1:-1:-100)']
  - Sample time = 0.25

- Samples per frame = 4

- Configure the Downsample block to downsample the two-channel input by decreasing the output frame rate by a factor of 2 relative to the input frame rate. Set a sample offset of 1, and a 4-by-2 initial condition matrix of

$$\begin{bmatrix} 11 & -11 \\ 12 & -12 \\ 13 & -13 \\ 14 & -14 \end{bmatrix}$$

  - Downsample factor = 2
  - Sample offset = 1
  - Initial condition = [11 -11;12 -12;13 -13;14 -14]
  - Frame-based mode = Maintain input frame size

- Configure the Probe blocks by selecting only the Probe sample time check box.

This model is multirate because there are at least two distinct frame rates, as shown by the two Probe blocks. To run this model in the Simulink multitasking mode, open the Configuration Parameters dialog box. From the list on the left side of the dialog box, click Solver. From the Type list, select Fixed-step, and from the Solver list, select Discrete (no continuous states). From the Tasking mode for periodic sample times list, select MultiTasking. Additionally, set the Stop time parameter to 30.

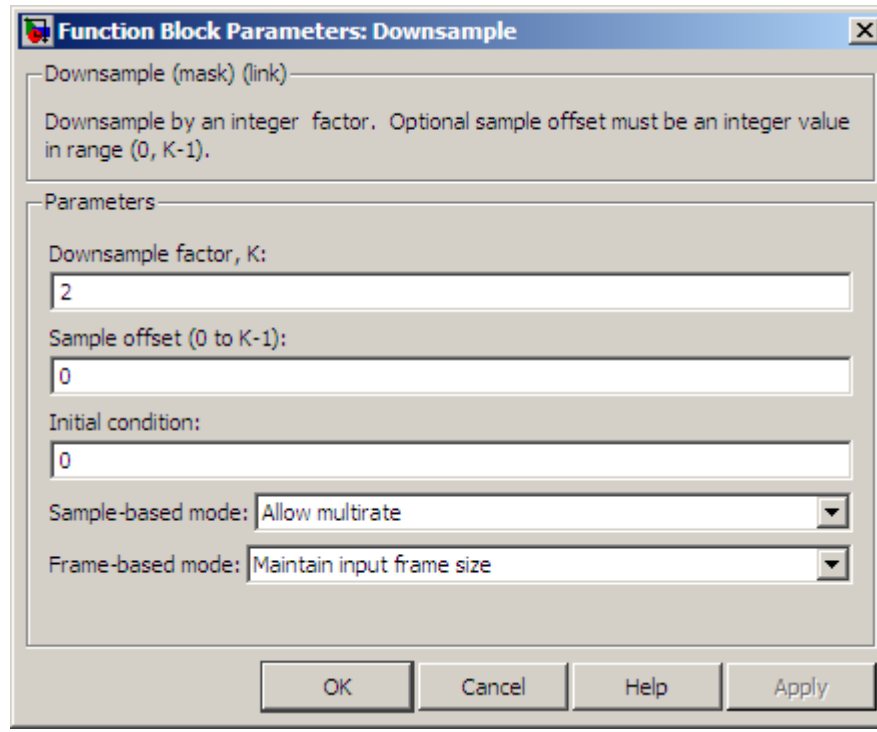
Run the model and look at the output, yout. The first few samples of each channel are shown:

```
yout =
    11    -11
    12    -12
    13    -13
    14    -14
```

```
 2    -2
 4    -4
 6    -6
 8    -8
10   -10
12   -12
14   -14
```

Since you ran this frame-based multirate model in multitasking mode, the first row of the initial condition matrix appears as the first output sample, followed by the other three initial condition rows. The second row of the first input matrix (row D+1, where D is the Sample offset) appears in the output as sample 5 (sample M_i+1, where M_i is the input frame size).

Dialog Box



Downsample factor
: The integer factor, K, by which to decrease the input sample rate.

Sample offset
: The sample offset, D, which must be an integer in the range [0, K–1].

Initial condition
: The value with which the block is initialized for cases of nonzero latency; a scalar value or an array of the same size as the input.

Sample-based mode
: The method by which to implement downsampling for sample-based inputs: Allow multirate (decrease the output sample rate), or Force single-rate (force the output sample

rate to match the input sample rate by repeating every Kth input sample K times at the output).

Frame-based mode

The method by which to implement downsampling for frame-based inputs: Maintain input frame size (decrease the frame rate), or Maintain input frame rate (decrease the frame size).

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

FIR Decimation	Signal Processing Blockset
FIR Rate Conversion	Signal Processing Blockset
Repeat	Signal Processing Blockset
Sample and Hold	Signal Processing Blockset
Upsample	Signal Processing Blockset

DSP Constant (Obsolete)

Purpose

Generate discrete- or continuous-time constant signal

Library

Signal Processing Sources

dspobslib

Description

Note The DSP Constant block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the Constant block.

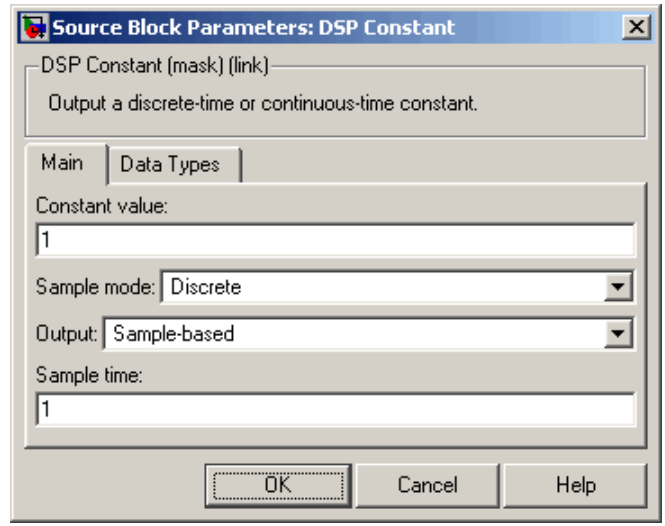
The DSP Constant block generates a signal whose value remains constant throughout the simulation. The Constant value parameter specifies the constant to output, and can be any valid MATLAB expression that evaluates to a scalar, vector, or matrix.

When Sample mode is set to Continuous, the output is a continuous-time signal. When Sample mode is set to Discrete, the Sample time parameter is visible, and the signal has the discrete output period specified by the Sample time parameter.

You can set the output signal to Frame-based, Sample-based, or Sample-based (interpret vectors as 1-D) with the Output parameter.

Dialog Box

The Main pane of the DSP Constant block dialog box appears as follows.



Opening this dialog box causes a running simulation to pause. See "Changing Source Block Parameters During Simulation" in the online Simulink documentation for details.

Constant value

Specify the constant to generate. This parameter is tunable; values entered here can be tuned, but their dimensions must remain fixed.

When you specify any data type information in this field, it is overridden by the value of the Output data type parameter in the Data Types pane, unless you select Inherit from 'Constant value'.

Sample mode

Specify the sample mode of the output, Discrete for a discrete-time signal or Continuous for a continuous-time signal.

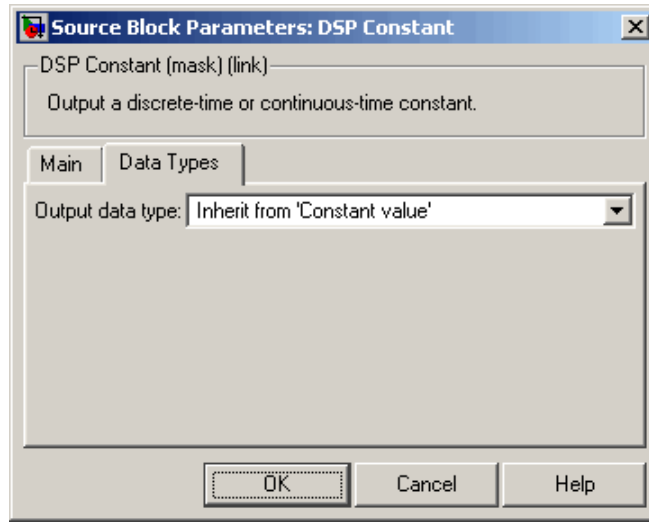
Output

Specify whether the output is Sample-based (interpret vectors as 1-D), Sample-based, or Frame-based. When you select Sample-based and the output is a vector, its dimension is constrained to match the Constant value dimension (row or column). When you select Sample-based (interpret vectors as 1-D), however, the output has no specified dimensionality.

Sample time

Specify the discrete sample period for sample-based outputs. When you select Frame-based for the Output parameter, this parameter is named Frame period, and is the discrete frame period for the frame-based output. This parameter is only visible when you select Discrete for the Sample mode parameter.

The Data Types pane of the DSP Constant block dialog box appears as follows.



Output data type

Specify the output data type in one of the following ways:

- Choose one of the built-in data types from the list.
- Choose Fixed-point to specify the output data type and scaling in the Signed, Word length, Set fraction length in output to, and Fraction length parameters.
- Choose User-defined to specify the output data type and scaling in the User-defined data type, Set fraction length in output to, and Fraction length parameters.
- Choose Inherit from 'Constant value' to set the output data type and scaling to match the values of the Constant value parameter in the Main pane.
- Choose Inherit via back propagation to set the output data type and scaling to match the following block.

The value of this parameter overrides any data type information specified in the Constant value parameter in the Main pane, except when you select Inherit from 'Constant value'.

Signed

Select to output a signed fixed-point signal. Otherwise, the signal is unsigned. This parameter is only visible when you select Fixed-point for the Output data type parameter.

Word length

Specify the word length, in bits, of the fixed-point output data type. This parameter is only visible when you select Fixed-point for the Output data type parameter.

User-defined data type

Specify any built-in or fixed-point data type. You can specify fixed-point data types using the following Simulink Fixed Point functions: sfix, ufix, sint, uint, sfrac, and ufrac. This parameter is only visible when you select User-defined for the Output data type parameter.

Set fraction length in output to

Specify the scaling of the fixed-point output by either of the following two methods:

- Choose Best precision to have the output scaling automatically set such that the output signal has the best possible precision.
- Choose User-defined to specify the output scaling in the Fraction length parameter.

This parameter is only visible when you select Fixed-point for the Output data type parameter, or when you select User-defined and the specified output data type is a fixed-point data type.

Fraction length

For fixed-point output data types, specify the number of fractional bits, or bits to the right of the binary point. This parameter is only visible when you select Fixed-point or User-defined for the Output data type parameter and User-defined for the Set fraction length in output to parameter.

Supported Data Types

- Double-precision floating point
- Single-precision floating point
- Fixed point
- Boolean
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

See Also

Constant

Signal From Workspace

Simulink

Signal Processing Blockset

Purpose

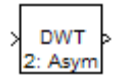
Compute discrete wavelet transform (DWT) of input

Library

Transforms

dspxfrm3

Description



Note The DWT block is the same as the Dyadic Analysis Filter Bank block in the Multirate Filters library, but with different default settings. See the Dyadic Analysis Filter Bank block reference page for more information on how to use the block.

The DWT block computes the discrete wavelet transform (DWT) of each column of a frame-based input. By default, the output is a sample-based vector or matrix with the same dimensions as the input. Each column of the output is the DWT of the corresponding input column.

You must install the Wavelet Toolbox™ product for the block to automatically design wavelet-based filters to compute the DWT. Otherwise, you must specify your own lowpass and highpass FIR filters by setting the Filter parameter to User defined.

For the same input, the DWT block and the Wavelet Toolbox function do not produce the same results. Because the blockset is designed for real-time implementation and the toolbox is designed for analysis, the products handle boundary conditions and filter states differently. To make the output of the dwt function and the DWT block match, complete the following steps:

1 For the dwt function, set the boundary condition to zero-padding by typing dwtmode('zpd') at the MATLAB command prompt.

2 To match the latency of the DWT block, which is implemented using FIR filters, add zeros to the input of the `dwt` function. The number of zeros you add must be equal to the half the filter length.

For detailed information about how to use this block, see the Dyadic Analysis Filter Bank block reference page.

Examples

See “Examples” on page 2-441 in the Dyadic Analysis Filter Bank block reference page.

See Also

Dyadic Analysis Filter Bank	Signal Processing Blockset
IDWT	Signal Processing Blockset
`dwt`	Wavelet Toolbox

Purpose

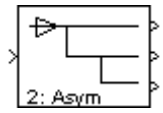
Decompose signals into subbands with smaller bandwidths and slower sample rates

Library

Filtering / Multirate Filters

dspmlti4

Description



Note This block decomposes frame-based signals with frame size a multiple of 2^n into either n+1 or 2^n subbands. To decompose sample-based signals or frame-based signals of different sizes, use the Two-Channel Analysis Subband Filter block. (You can connect multiple copies of the Two-Channel Analysis Subband Filter block to create a multilevel dyadic analysis filter bank.)

The Dyadic Analysis Filter Bank block decomposes a broadband signal into a collection of subbands with smaller bandwidths and slower sample rates. The block uses a series of highpass and lowpass FIR filters to repeatedly divide the input frequency range, as illustrated in the figure "n-Level Asymmetric Dyadic Analysis Filter Bank".

You can specify the filter bank's highpass and lowpass filters by providing vectors of filter coefficients. If you install the Wavelet Toolbox product, you can also specify wavelet-based filters by selecting a wavelet from the Filter parameter. You must set the filter bank structure to asymmetric or symmetric, and specify the number of levels in the filter bank.

Input Requirements

- Input can be a frame-based vector or frame-based matrix.
- The input frame size must be a multiple of 2^n, where n is the number of filter bank levels. For example, a frame size of 16 would be appropriate for a three-level tree (16 is a multiple of 2^3).
- The block always operates along the columns of the inputs.

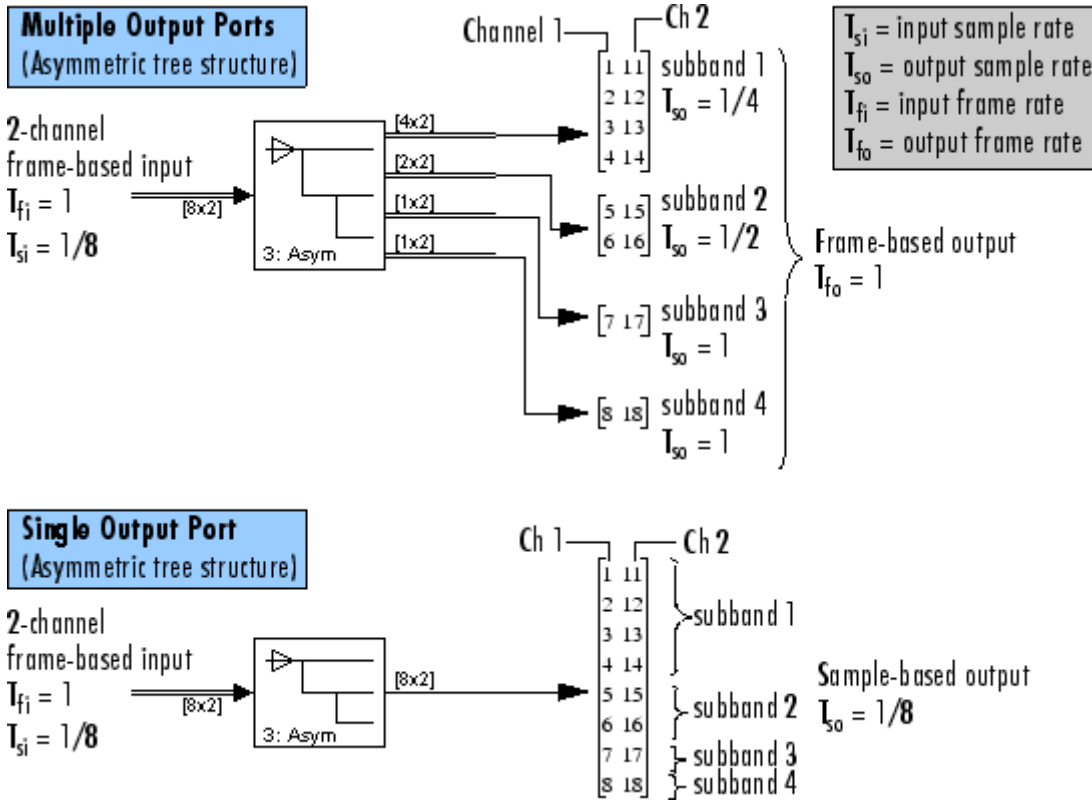
For an illustration of why the above input requirements exist, see the Þgure Outputs of a 3-Level Asymmetric Dyadic Analysis Filter Bank on page 2-437.

Output Characteristics

The output characteristics vary de pending on the block's parameter settings, as summarized in the following list and Þgure:

- Number of levels parameter set to n
- Tree structure parameter setting:
  - Asymmetric — Block produces n+1 output subbands
  - Symmetric — Block produces 2n output subbands
- Output parameter setting can be Multiple ports or Single port . The following Þgure illustrates the di fference between the two settings for a 3-level asymmetric dyadic analysis Þlter bank. For an explanation of the illustrated out put characteristics, see the table Output Characteristics for an n-Le vel Dyadic Analysis Filter Bank on page 2-438.

For more information about the Þlter bank levels and structures, see "Dyadic Analysis Filter Banks".



Outputs of a 3-Level Asymmetric Dyadic Analysis Filter Bank

The following table summarizes the di fferent output characteristics of the block when it is set to output from single or multiple ports.

Output Characteristics for an n-Level Dyadic Analysis Filter Bank

	Single Output Port	Multiple Output Ports
Output Description	Block concatenates all the subbands into one vector or matrix, and outputs the concatenated subbands from a single output port. Each output column contains subbands of the corresponding input channel.	Block outputs each subband from a separate output port. The topmost port outputs the subband with the highest frequencies. Each output column contains a subband for the corresponding input channel.
Output Frame Status	Sample-based	Frame-based
Output Frame Rate	Not applicable	Same as input frame rate (However, the output frame sizes can vary, so the output sample rates can vary.)
Output Dimensions (Frame Size)	Same number of rows and columns as the input.	The output has the same number of columns as the input. The number of output rows is the output frame size. For an input with frame size M_i output y_k has frame size $M_{o,k}$: • Symmetric — All outputs have the frame size, $M_i / 2^n$. • Asymmetric — The frame size of each output (except the last) is half that of the output from the previous level. The outputs from the last two output ports have the same frame size since they

Output Characteristics for an n-Level Dyadic Analysis Filter Bank (Continued)

	Single Output Port	Multiple Output Ports
		originate from the same level in the filter bank. $M_{o,k} = \begin{cases} M_i / 2^k & (1 \le k \le n) \\ M_i / 2^n & (k = n+1) \end{cases}$
Output Sample Rate	Same as input sample rate.	Though the outputs have the same frame rate as the input, they have different frame sizes than the input. Thus, the output sample rates, F_{so}, k, are different from the input sample rate, F_{si}: • Symmetric — All outputs have the sample rate $F_{si} / 2^n$. • Asymmetric — $F_{so,k} = \begin{cases} F_{si}/2^k & (1 \le k \le n) \\ F_{si}/2^n & (k = n+1) \end{cases}$

Filter Bank Filters

You must specify the highpass and lowpass filters in the filter bank by setting the Filter parameter to one of the following options:

- User defined — Allows you to explicitly specify the filters with two vectors of filter coefficients in the Lowpass FIR filter coefficients and Highpass FIR filter coefficients parameters. The block uses the same lowpass and highpass filters throughout the filter bank. The two filters should be halfband filters, where each filter passes the frequency band that the other filter stops.

- Wavelet such as Biorthogonal or Daubechies — The block uses the specified wavelet to construct the lowpass and highpass filters using the Wavelet Toolbox function wfilters. Depending on the wavelet, the block might enable either the Wavelet order or Filter order [synthesis / analysis] parameter. (The latter parameter allows you to specify different wavelet orders for the analysis and synthesis filter stages.) You must install the Wavelet Toolbox product to use wavelets.

Specifying Filters with the Filter Parameter and Related Parameters

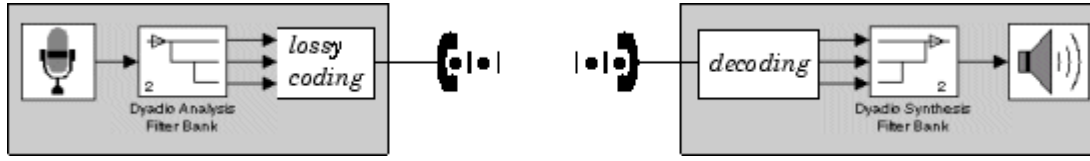
Filter	Sample Setting for Related Filter Specification Parameters	Corresponding Wavelet Function Syntax
User-defined	Filters based on Daubechies wavelets with wavelet order 3: • Highpass FIR filter coefficients = [-0.3327 0.8069 -0.4599 -0.1350 0.0854 0.0352] • Lowpass FIR filter coefficients = [0.0352 -0.0854 -0.1350 0.4599 0.8069 0.3327]	None
Haar	None	wfilters('haar')
Daubechies	Wavelet order = 4	wfilters('db4')
Symlets	Wavelet order = 3	wfilters('sym3')
Coiflets	Wavelet order = 1	wfilters('coif1')
Biorthogonal	Filter order [synthesis / analysis] = [3/1]	wfilters('bior3.1')
Reverse Biorthogonal	Filter order [synthesis / analysis] = [3/1]	wfilters('rbio3.1')
Discrete Meyer	None	wfilters('dmey')

Examples

Wavelets

The primary application for dyadic analysis filter banks and dyadic synthesis filter banks, is coding for data compression using wavelets.

At the transmitting end, the output of the dyadic analysis filter bank is fed to a lossy compression scheme, which typically assigns the number of bits for each filter bank output in proportion to the relative energy in that frequency band. This represents the more powerful signal components by a greater number of bits than the less powerful signal components.



At the receiving end, the transmission is decoded and fed to a dyadic synthesis filter bank to reconstruct the original signal. The filter coefficients of the complementary analysis and synthesis stages are designed to cancel aliasing introduced by the filtering and resampling.

Demos

See the Signal Processing Blockset Wavelet Reconstruction and Noise Reduction demo, which uses the Dyadic Analysis Filter Bank and Dyadic Synthesis Filter Bank blocks.

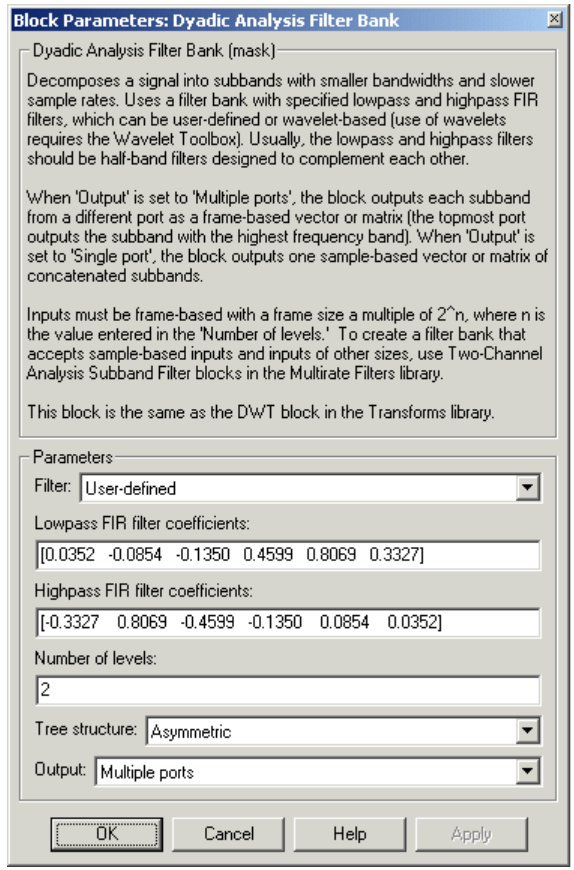
Note To see the version of the demos that use the Dyadic Analysis Filter Bank and Dyadic Synthesis Filter Bank blocks, click the Frame-Based Demo button in the demos.

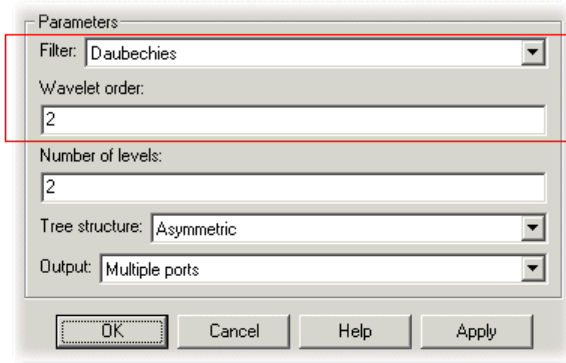
Open the demos using one of the following methods:

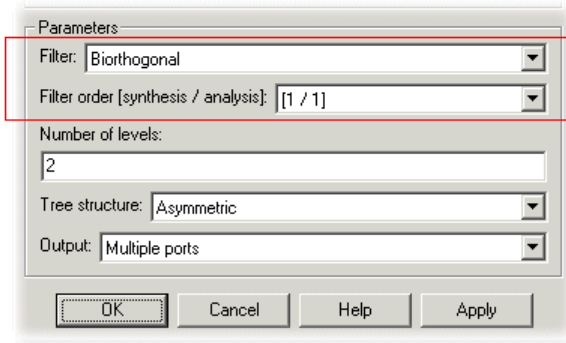
- Click the above link in the MATLAB Help browser (not in a Web browser).
- Type `demo blockset signal` at the MATLAB command line, and look in the `Miscellaneous` directory.

Dialog Box

The parameters displayed in the block dialog vary depending on the setting of the Filter parameter. Only some of the parameters described below are visible in the dialog box at any one time.







Filter

The type of filter used to determine the high- and low-pass FIR filters in the dyadic analysis filter bank:

Select User defined to explicitly specify the filter coefficients in the Lowpass FIR filter coefficients and Highpass FIR filter coefficients parameters.

Select a wavelet such as Biorthogonal or Daubechies to specify a wavelet-based filter. The block uses the Wavelet Toolbox function wfilters to construct the filters. Extra parameters such as Wavelet order or Filter order [synthesis / analysis] might become enabled. For a list of the supported wavelets, see Specifying Filters with the Filter Parameter and Related Parameters on page 2-440.

Lowpass FIR filter coefficients

A vector of filter coefficients (descending powers of z) that specifies coefficients used by all the lowpass filters in the filter bank. This parameter is enabled when you set Filter to User defined. The lowpass filter should be a half-band filter that passes the frequency band stopped by the filter specified in the Highpass FIR filter coefficients parameter. The default values of this parameter specify a filter based on a Daubechies wavelet with wavelet order 3.

Highpass FIR filter coefficients

A vector of filter coefficients (descending powers of z) that specifies coefficients used by all the highpass filters in the filter bank. This parameter is enabled when you set Filter to User defined. The highpass filter should be a half-band filter that passes the frequency band stopped by the filter specified in the Lowpass FIR filter coefficients parameter. The default values of this parameter specify a filter based on a Daubechies wavelet with wavelet order 3.

Wavelet order

The order of the wavelet selected in the Filter parameter. This parameter is enabled only when you set Filter to certain types of wavelets, as shown in the Specifying Filters with the Filter Parameter and Related Parameters table.

Filter order [synthesis / analysis]

The order of the wavelet for the synthesis and analysis filter stages. For example, when you set the Filter parameter to Biorthogonal and set the Filter order [synthesis / analysis] parameter to [2 / 6], the block calls the wfilters function with

input argument 'bior2.6'. This parameter is enabled only when you set Filter to certain types of wavelets, as shown in Specifying Filters with the Filter Parameter and Related Parameters on page 2-440.

Number of levels

The number of filter bank levels. An n-level asymmetric structure has n+1 outputs, and an n-level symmetric structure has 2^n outputs, as shown in the figures "n-Level Asymmetric Dyadic Analysis Filter Bank" and "n-Level Symmetric Dyadic Analysis Filter Bank". The block's icon displays the value of this parameter in the lower-left corner.

Tree structure

The structure of the filter bank: Asymmetric, or Symmetric. See the figures "n-Level Asymmetric Dyadic Analysis Filter Bank" and "n-Level Symmetric Dyadic Analysis Filter Bank".

Output

Set to Multiple ports to output each output subband on a separate port (the topmost port outputs the subband with the highest frequency band). Set to Single port to concatenate the subbands into one vector or matrix and output the concatenated subbands on a single port. For more information, see "Output Characteristics" on page 2-436.

References

Fliege, N. J. Multirate Digital Signal Processing: Multirate Systems, Filter Banks, Wavelets. West Sussex, England: John Wiley & Sons, 1994.

Strang, G. and T. Nguyen. Wavelets and Filter Banks. Wellesley, MA: Wellesley-Cambridge Press, 1996.

Vaidyanathan, P. P. Multirate Systems and Filter Banks. Englewood Cliffs, NJ: Prentice Hall, 1993.

Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

- Dyadic Synthesis Filter Bank
- Two-Channel Analysis Subband Filter
- Signal Processing Blockset
- Signal Processing Blockset

Purpose

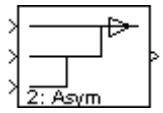
Reconstruct signals from subbands w ith smaller bandwidths and slower sample rates

Library

Filtering / Multirate Filters

dspmlti4

Description



Note This block always outputs frame-based signals, and its inputs must be of certain sizes. To get sample-based outputs or to use input subbands that do not Þt the criteria of this block, use the Two-Channel Synthesis Subband Filter block. (You can connect multiple copies of the Two-Channel Synthesis Subband F ilter block to create a multilevel dyadic synthesis Þlter bank.)

The Dyadic Synthesis Filter Bank block reconstructs a signal decomposed by the Dyadic Analysis F ilter Bank block. The block takes in subbands of a signal, and uses them to reconstruct the signal by using a series of highpass and lowpass FIR Þlters as illustrated in the Þgure “n-Level Asymmetric Dyad ic Synthesis Filter Bank”. The reconstructed signal has a wider bandwidth and faster sample rate than the input subbands.

You can specify the Þlter bank’s highpass and lowpass Þlters by providing vectors of Þlter coef Þcients. If you install the Wavelet Toolbox product, you can also specify wavelet-based Þlters by selecting a wavelet from the Filter parameter.

Note To use a dyadic synthesis filter bank to perfectly reconstruct the output of a dyadic analysis filter bank, the number of levels and tree structures of both filter banks must be the same. In addition, the filters in the synthesis filter bank must be designed to perfectly reconstruct the outputs of the analysis filter bank. Otherwise, the reconstruction is not perfect.

This block automatically computes wavelet-based perfect reconstruction filters when the wavelet selection in the Filter parameter of this block is the same as the Filter parameter setting of the corresponding Dyadic Analysis Filter Bank block. The use of wavelets requires the Wavelet Toolbox product. To learn how to design your own perfect reconstruction filters, see "References" on page 2-457.

Input Requirements

The inputs to this block are usually the outputs of a Dyadic Analysis Filter Bank block. Since the Dyadic Analysis Filter Bank block can output from either a single port or multiple ports, the Dyadic Synthesis Filter Bank block accepts inputs to either a single port or multiple ports.

The Input parameter sets whether the block accepts inputs from a single port or multiple ports, and thus determines the input requirements, as summarized in the following lists and figure.

Note Any output of a Dyadic Analysis Filter Bank block whose parameter settings match the corresponding settings of this block is a valid input to this block. For example, the setting of the Dyadic Analysis Filter Bank block parameter, Output, must be the same as this block's Input parameter (Single port or Multiple ports).

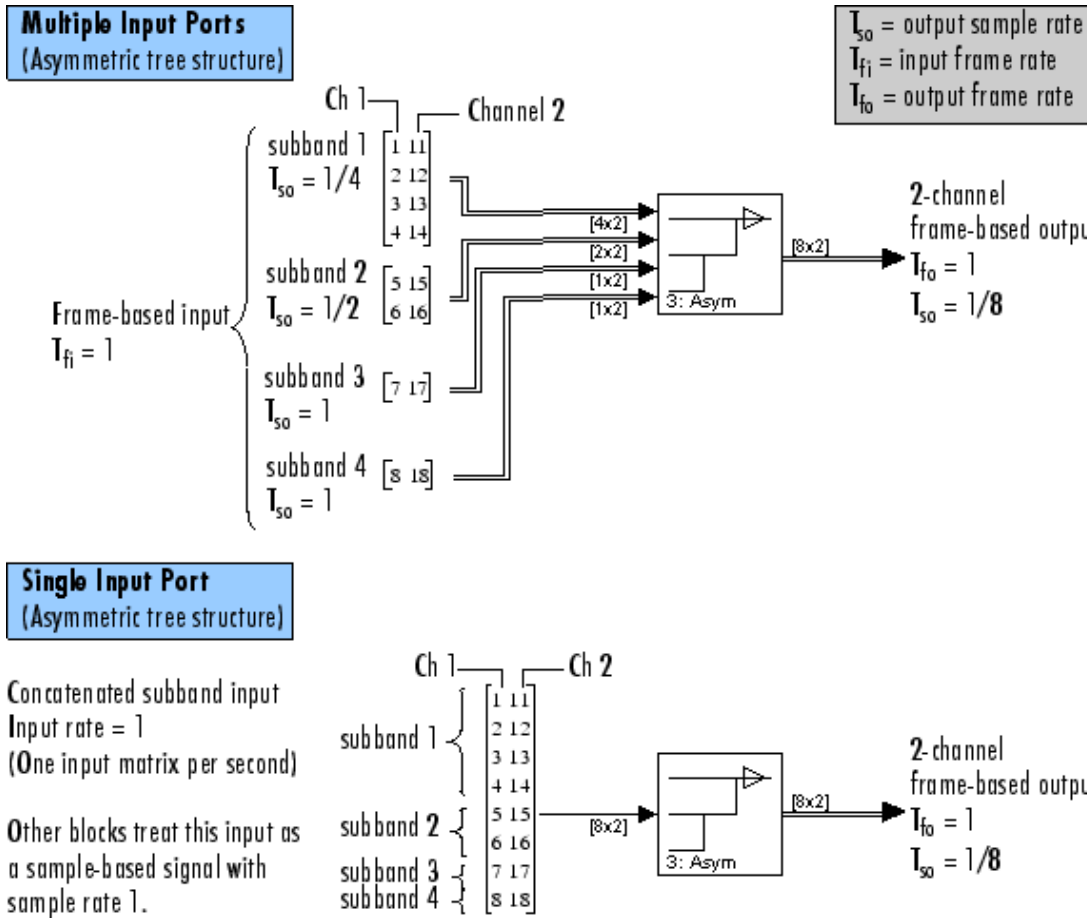
Valid Inputs for Input Set to Single Port

- Inputs must be sample-based vectors or sample-based matrices of concatenated subbands.

- Each input column contains the subb ands for an independent signal.
- Upper input rows contain the high-frequency subbands, and the lower rows contain the low-frequency subbands.

Valid Inputs for Input Set to Multiple Ports

- Inputs must be a frame-based vect or or frame-based matrix for each subband, each of which is input to a separate input port.
- The columns of each input contains a subband for an independent signal.
- The input to the topmost input port is the subband containing the highest frequencies, and the inp ut to the bottommost port is the subband containing the lowest frequencies.



Valid Inputs to a 3-Level Asymmetric Dyadic Synthesis Filter Bank

For general information about the Þlter banks, see “Dyadic Synthesis Filter Banks”.

Output Characteristics

The following table summarizes the output characteristics for both frame-based inputs, and concatenated subband inputs. For an

illustration of why the output ch aracteristics exist, see the Þgure Valid Inputs to a 3-Level Asymmetric Dyad ic Synthesis Filter Bank on page 2-450.

	Frame-Based Inputs (Input = Multiple ports)	Concatenated Subband Inputs (Input = Single port)
Output Frame Status	Outputs are always frame based regardless of the input frame status. Each output column is an independent channel, reconstructed from the corresponding channel in the inputs.	
Output Frame Rate	Same as the input frame rate.	Same as the input rate (the rate of the concatenated subband inputs).
Output Frame Dimensions	• The output has the same number of columns as the inputs. • The number of output rows depends on the tree structure of the Þlter bank: - Asymmetric — The number of output rows is twice the number of rows in the input to the topmost input port. - Symmetric — The number of output rows is the product of the number of input ports and the number of rows in an input to any input port.	The output has the same number of rows and columns as the input.

For general information about the Þlter banks, see “Dyadic Synthesis Filter Banks”.

Filter Bank Filters

You must specify the highpass and lowpass Þlters in the Þlter bank by setting the Filter parameter to one of the following options:

- User defined — Allows you to explicitly specify the filters with two vectors of filter coefficients in the Lowpass FIR filter coefficients and Highpass FIR filter coefficients parameters. The block uses the same lowpass and highpass filters throughout the filter bank. The two filters should be halfband filters, where each filter passes the frequency band that the other filter stops. To use this block to perfectly reconstruct a signal decomposed by a Dyadic Analysis Filter Bank block, the filters in this block must be designed to perfectly reconstruct the outputs of the analysis filter bank. To learn how to design your own perfect reconstruction filters, see "References" on page 2-457.
- Wavelet such as Biorthogonal or Daubechies — The block uses the specified wavelet to construct the lowpass and highpass filters using the Wavelet Toolbox function wfilters. Depending on the wavelet, the block might enable either the Wavelet order or Filter order [synthesis / analysis] parameter. (The latter parameter allows you to specify different wavelet orders for the analysis and synthesis filter stages.) To use this block to reconstruct a signal decomposed by a Dyadic Analysis Filter Bank block, you must set both blocks to use the same wavelets with the same order. You must install the Wavelet Toolbox product to use wavelets.

Specifying Filters with the Filter Parameter and Related Parameters

Filter	Sample Setting for Related Filter Specification Parameters	Corresponding Wavelet Function Syntax
User-defined	Filters based on Daubechies wavelets with wavelet order 3: • Lowpass FIR filter coefficients = [0.0352 -0.0854 -0.1350 0.4599 0.8069 0.3327] • Highpass FIR filter coefficients =	None

Specifying Filters with the Filter Parameter and Related Parameters (Continued)

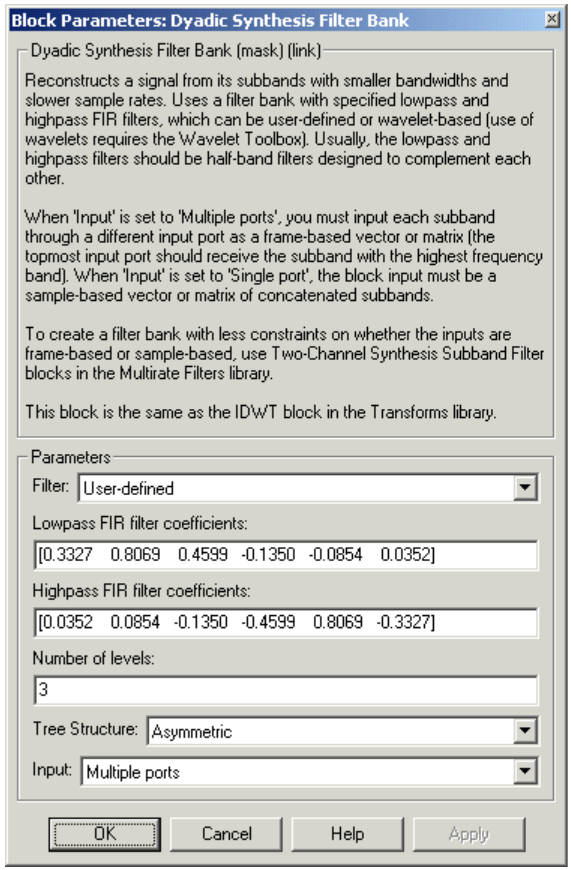
Filter	Sample Setting for Related Filter Specification Parameters	Corresponding Wavelet Function Syntax
	[-0.3327 0.8069 -0.4599 -0.1350 0.0854 0.0352]	
Haar	None	wfilters('haar')
Daubechies	Wavelet order = 4	wfilters('db4')
Symlets	Wavelet order = 3	wfilters('sym3')
Coiflets	Wavelet order = 1	wfilters('coif1')
Biorthogonal	Filter order [synthesis / analysis] = [3/1]	wfilters('bior3.1')
Reverse Biorthogonal	Filter order [synthesis / analysis] = [3/1]	wfilters('rbio3.1')
Discrete Meyer	None	wfilters('dmey')

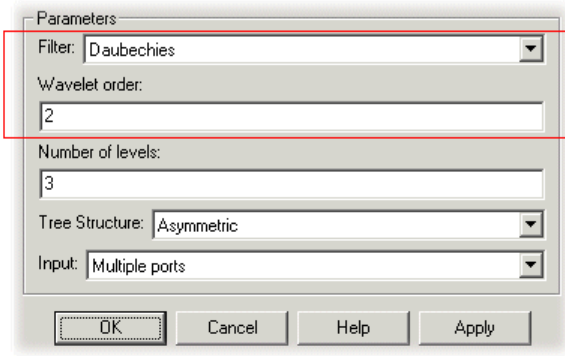
Examples

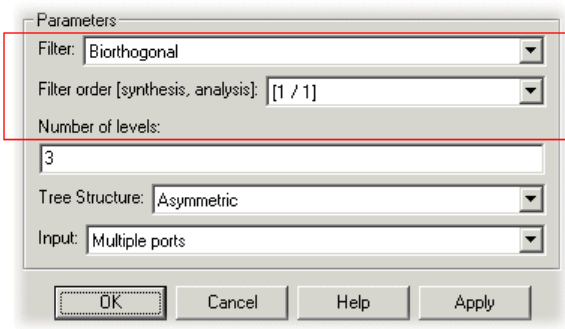
See “Examples” on page 2-441 in the Dyadic Analysis Filter Bank block reference.

Dialog Box

The parameters displayed in the block dialog vary depending on the setting of the Filter parameter. Only some of the parameters described below are visible in the dialog box at any one time.







Note To use this block to reconstruct a signal decomposed by a Dyadic Analysis Filter Bank block, all the parameters in this block must be the same as the corresponding parameters in the Dyadic Analysis Filter Bank block (except the Lowpass FIR filter coefficients and Highpass FIR filter coefficients; see the descriptions of these parameters).

Filter

The type of filter used to determine the high- and low-pass FIR filters in the dyadic synthesis filter bank:

- Select User defined to explicitly specify the filter coefficients in the Lowpass FIR filter coefficients and Highpass FIR filter coefficients parameters.
- Select a wavelet such as Biorthogonal or Daubechies to specify a wavelet-based filter. The block uses the Wavelet Toolbox function wfilters to construct the filters. Extra parameters such as Wavelet order or Filter order [synthesis / analysis] might become enabled. For a list of the supported wavelets, see the table Specifying Filters with the Filter Parameter and Related Parameters on page 2-452.

Lowpass FIR filter coefficients

A vector of filter coefficients (descending powers of z) that specifies coefficients used by all the lowpass filters in the filter bank. This parameter is enabled when you set Filter to User defined. The lowpass filter should be a half-band filter that passes the frequency band stopped by the filter specified in the Highpass FIR filter coefficients parameter. To perfectly reconstruct a signal decomposed by the Dyadic Analysis Filter Bank, the filters in this block must be designed to perfectly reconstruct the outputs of the analysis filter bank. Otherwise, the reconstruction is not perfect. The default values of this parameter specify a perfect reconstruction filter for the default settings of the Dyadic Analysis Filter Bank (based on a Daubechies wavelet with wavelet order 3).

Highpass FIR filter coefficients

A vector of filter coefficients (descending powers of z) that specifies coefficients used by all the highpass filters in the filter bank. This parameter is enabled when you set Filter to User defined. The highpass filter should be a half-band filter that passes the frequency band stopped by the filter specified in the Lowpass FIR filter coefficients parameter. To perfectly reconstruct a signal decomposed by the Dyadic Analysis Filter Bank, the filters in this block must be designed to perfectly reconstruct the outputs

of the analysis filter bank. Otherwise, the reconstruction is not perfect. The default values of this parameter specify a perfect reconstruction filter for the default settings of the Dyadic Analysis Filter Bank (based on a Daubechies wavelet with wavelet order 3).

Wavelet order

The order of the wavelet selected in the Filter parameter. This parameter is enabled only when you set Filter to certain types of wavelets, as shown in the table Specifying Filters with the Filter Parameter and Related Parameters on page 2-452.

Filter order [synthesis / analysis]

The order of the wavelet for the synthesis and analysis filter stages. For example, when you set the Filter parameter to Biorthogonal and set the Filter order [synthesis / analysis] parameter to [2 / 6], the block calls the wfilters function with input argument 'bior2.6'. This parameter is enabled only when you set Filter to certain types of wavelets, as shown in Specifying Filters with the Filter Parameter and Related Parameters on page 2-452.

Number of levels

The number of filter bank levels. An n-level asymmetric structure has n+1 inputs, and an n-level symmetric structure has 2^n inputs, as shown in “n-Level Asymmetric Dyadic Synthesis Filter Bank” and “n-Level Symmetric Dyadic Synthesis Filter Bank”.

Tree structure

The structure of the filter bank: Asymmetric, or Symmetric. See the figures “n-Level Asymmetric Dyadic Synthesis Filter Bank” and “n-Level Symmetric Dyadic Synthesis Filter Bank”.

Input

Set to Multiple ports to accept each input subband at a separate port (the topmost port accepts the subband with the highest frequency band). Set to Single port to accept one vector or matrix of concatenated subbands at a single port. For more information, see “Input Requirements” on page 2-448.

References

Fliege, N. J. Multirate Digital Signal Pr ocessing: Multirate Systems, Filter Banks, Wavelets . West Sussex, England: John Wiley & Sons, 1994.

Strang, G. and T. Nguyen. Wavelets and Filter Banks . Wellesley, MA: Wellesley-Cambridge Press, 1996.

Vaidyanathan, P. P. Multirate Systems and Filter Banks . Englewood Cliffs, NJ: Prentice Hall, 1993.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Dyadic Analysis Filter Bank	Signal Processing Blockset
Two-Channel Synthesis Subband Filter	Signal Processing Blockset

See “Multirate Filters” for related information.

Edge Detector

Purpose

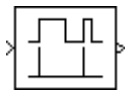
Detect transition from zero to nonzero value

Library

Signal Management / Switches and Counters

dspswit3

Description

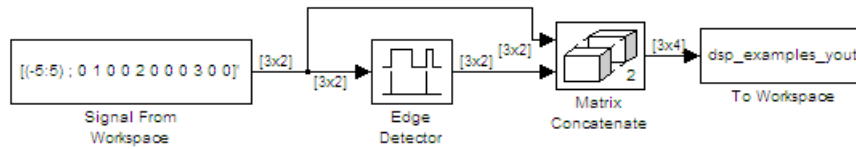


The Edge Detector block generates an impulse (the value 1) in a given output channel when the corresponding channel of the input transitions from zero to a nonzero value. Otherwise, the block generates zeros in each channel.

The output has the same dimension and sample rate as the input. When the input is frame based, the output is frame based; otherwise, the output is sample based. For frame-based input, an edge that is split across two consecutive frames (that is, a zero at the bottom of the first frame, and a nonzero value at the top of the following frame) is counted in the frame that contains the nonzero value.

Examples

In the model below, the Edge Detector block locates the edges (zero to nonzero transitions) in a two-channel frame-based input with frame size 3. The two input channels are horizontally concatenated with the two output channels to create the four-channel workspace variable dsp_examples_yout.



Adjust the block parameters as described below. (Use the default settings for the To Workspace block.)

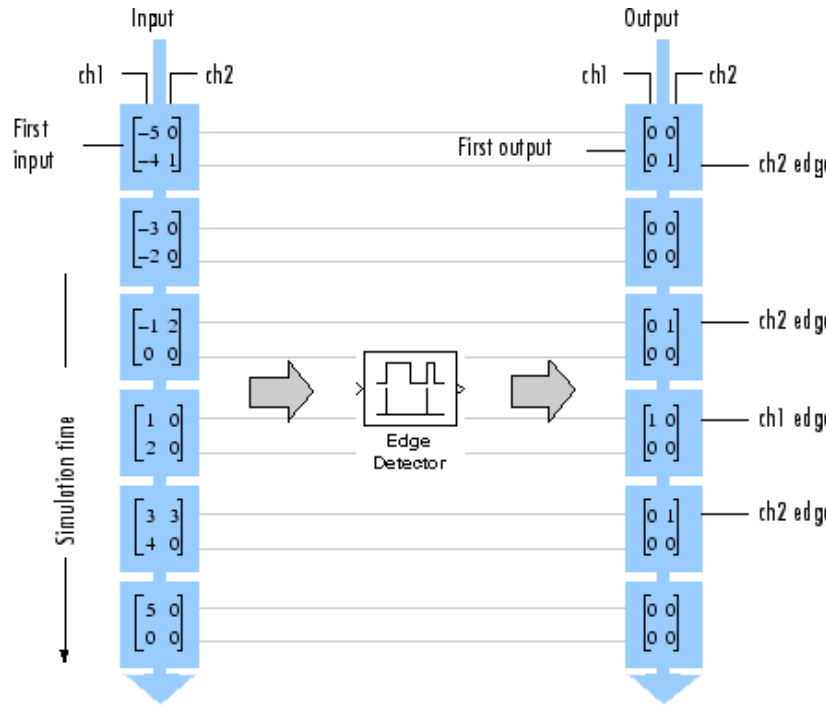
- Set the Signal From Workspace block parameters as follows:
  - Signal = [(-5:5) ; 0 1 0 0 2 0 0 0 3 0 0]'
  - Sample time = 1

- Samples per frame = 3

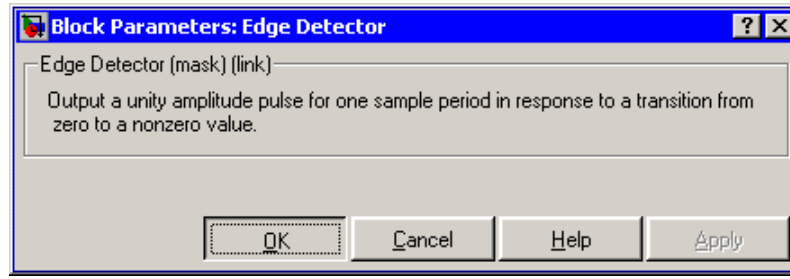
• Set the Matrix Concatenate b lock parameters as follows:

- Number of inputs = 2
- Mode = Multidimensional array
- Concatenate dimension = 2

As shown below, the block Þnds edges at sample 7 in channel 1, and at samples 2, 5, and 9 in channel 2.



Dialog Box



Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Fixed point (signed and unsigned)
- Boolean — The block might output Boolean values depending on the input data type, and whether Boolea n support is enabled or disabled.
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

See Also

Counter	Signal Processing Blockset
Event-Count Comparator	Sig nal Processing Blockset

Purpose

Detect threshold crossing o f accumulated nonzero inputs

Library

Signal Management / Switches and Counters

dspswit3

Description

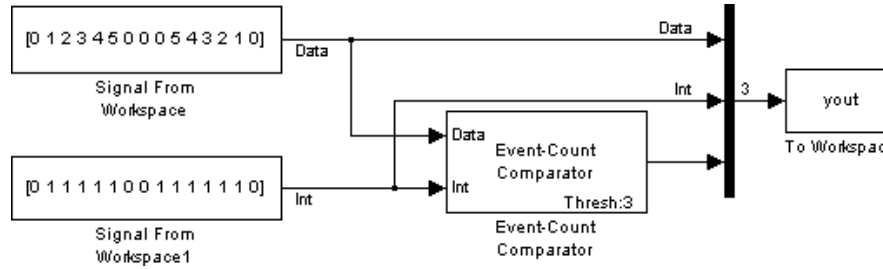
The Event-Count Comparator block records the number of nonzero inputs to the Data port during the period that the block is enabled by a high signal (the value 1) at the Int port. Both inputs must be scalars; the input to the Int port can be sample based or frame based. When the input to the Data port is frame based, the output is frame based; otherwise, the output is sample based.

When the number of accumulated nonzero inputs Þrst equals the Event threshold setting, the block waits one additional sample interval, and then sets the output high (1). The block holds the output high until recording is restarted by a low-to-high (0-to-1) transition at the Int port.

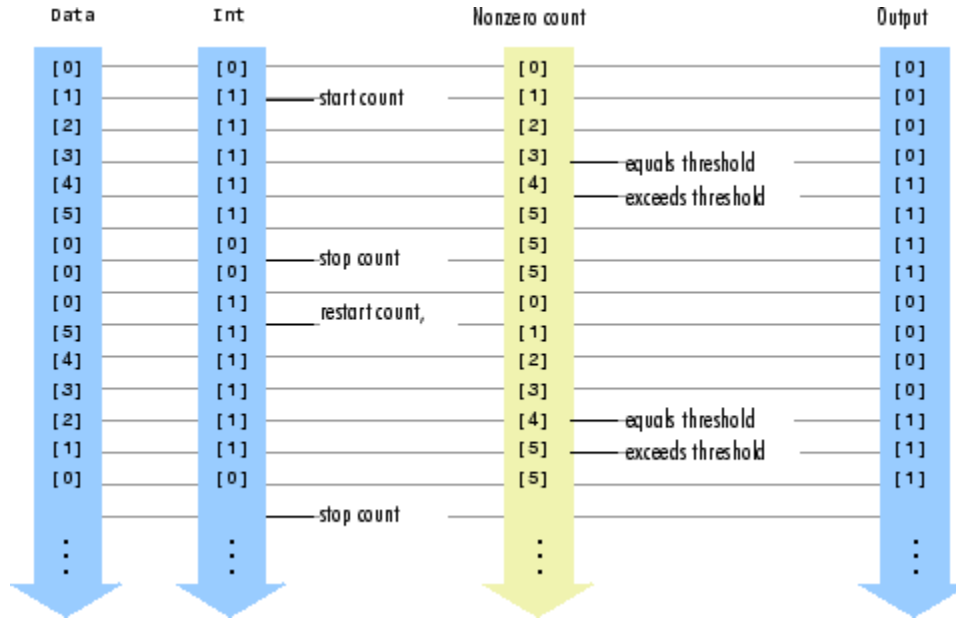
The Event-Count Comparator block accepts real and complex ßoating-point and Þxed-point inputs. However, because the block has discrete state, it does not support constant or continuous sample times. Therefore, at least one input or output port of the Event-Count Comparator block must be connected to a block whose Sample time parameter is discrete. The Event-Co unt Comparator block inherits this non-in Þnite discrete sample time.

Examples

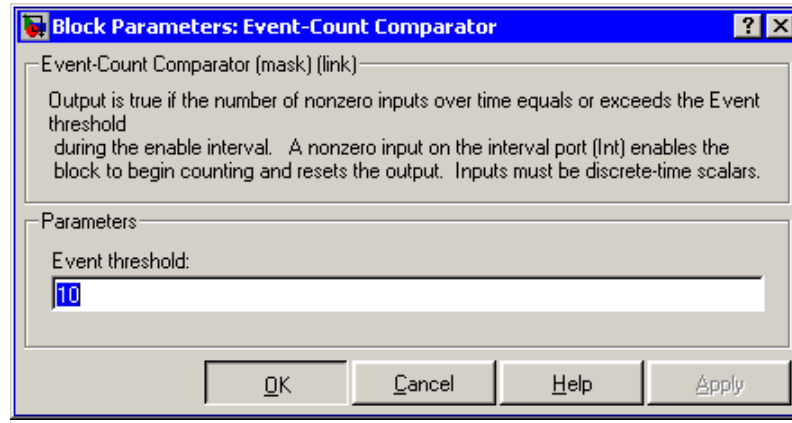
In the model below, the Event-Count Comparator block (Event threshold = 3) detects two threshold crossings in the input to the Data port, one at sample 4 and one at sample 12.



All inputs and outputs are multiplexed into the workspace variable yout , whose contents are shown in the Þgure below. The two left columns in the illustration show the inputs to the Data and Int ports, the center column shows the state of the block's internal counter, and the right column shows the block's output.



Dialog Box



Event threshold
: Specify the value against which to compare the number of nonzero inputs. Tunable.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Fixed point (signed and unsigned)
- Boolean
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

See Also

Counter	Signal Processing Blockset
Edge Detector	Signal Processing Blockset

Extract Diagonal

Purpose

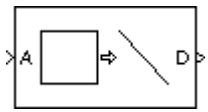
Extract main diagonal of input matrix

Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

dspmtrx3

Description

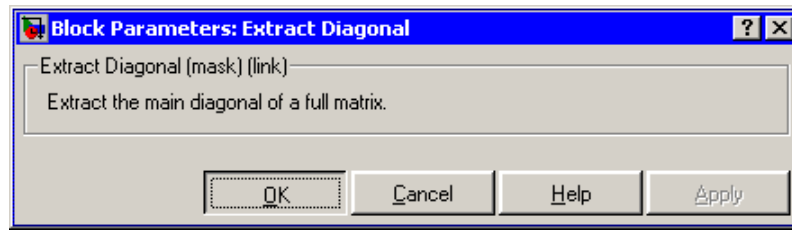


The Extract Diagonal block populat es the 1-D output vector with the elements on the main diagonal of the M-by-N input matrix A.

D = diag(A) Equivalent MATLAB code

The output vector has length min(M,N) , and is always sample based.

Dialog Box



Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Fixed point (signed and unsigned)
- Boolean — Block outputs are always Boolean.
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

See Also

Constant Diagonal Matrix	Signal Processing Blockset
Create Diagonal Matrix	Signal Processing Blockset

Extract Triangular Matrix	Signal Processing Blockset
diag	MATLAB

Extract Triangular Matrix

Purpose

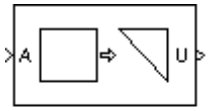
Extract lower or upper triangle from input matrices

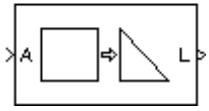
Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

dspmtrx3

Description





The Extract Triangular Matrix block creates a triangular matrix output from the upper or lower triangular elements of an M-by-N input matrix. A length- M 1-D vector input is treated as an M-by-1 matrix.

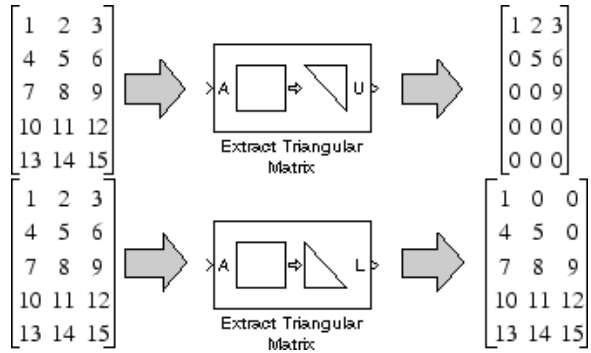
The Extract parameter selects between the two components of the input:

- Upper — Copies the elements on and a bove the main diagonal of the input matrix to an output matrix of the same size. The Þrst row of the output matrix is therefore identical to the Þrst row of the input matrix. The elements below the main diagonal of the output matrix are zero.
- Lower — Copies the elements on and b elow the main diagonal of the input matrix to an output matrix of the same size. The Þrst column of the output matrix is therefore identical to the Þrst column of the input matrix. The elements above the main diagonal of the output matrix are zero.

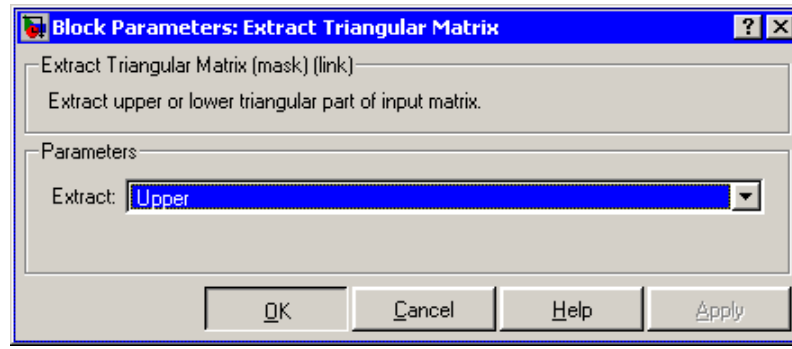
The output has the same frame status as the input.

Examples

The example below shows the extra ction of upper and lower triangles from a 5-by-3 input matrix.



Dialog Box



Extract

The component of the matrix to copy to the output, upper triangle or lower triangle.

Supported Data Types

Port	Supported Data Types
A	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
U	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
L	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Autocorrelation LPC	Signal Processing Blockset
Cholesky Factorization	Signal Processing Blockset
Constant Diagonal Matrix	Signal Processing Blockset
Extract Diagonal	Signal Processing Blockset

Forward Substitution	Signal Processing Blockset
LDL Factorization	Signal Processing Blockset
LU Factorization	Signal Processing Blockset
tril	MATLAB
triu	MATLAB

Fast Block LMS Filter

Purpose

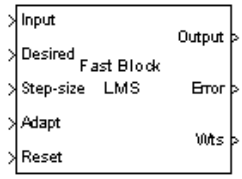
Compute output, error, and weigh ts using LMS adaptive algorithm

Library

Filtering / Adaptive Filters

dspadpt3

Description



Fast Block LMS Filter

The Fast Block LMS Filter block implements an adaptive least mean-square (LMS) Þlter, where the adaptation of the Þlter weights occurs once for every block of data s amples. The block estimates the Þlter weights, or coef Þcients, needed to convert the input signal into the desired signal. Connect the signal you want to Þlter to the Input port. This input signal can be a sample-based scalar or a single-channel frame-based signal. Connect the signal you want to model to the Desired port. The desired signal mu st have the same data type, frame status, complexity, and dimensions as the input signal. The Output port outputs the Þltered input signal, which can be sample or frame based. The Error port outputs the result of subtracting the output signal from the desired signal.

The block calculates the Þlter weights using the Block LMS Filter equations. For more information, see Block LMS Filter. The Fast Block LMS Filter block implements the convolution operation involved in the calculations of the Þltered output, y, and the weight update function in the frequency domain using the FFT al gorithm used in the Overlap-Save FFT Filter block. See Overlap-Save FFT Filter for more information.

Use the Filter length parameter to specify the length of the Þlter weights vector.

The Block size parameter determines how many samples of the input signal are acquired before the Þlter weights are updated. The input frame length must be a multiple of the Block size parameter.

The Step-size (mu) parameter corresponds to μ in the equations. You can either specify a step-size using the input port, Step-size, or enter a value in the Block Parameters: Block LMS Filter dialog box.

Use the Leakage factor (0 to 1) parameter to specify the leakage factor, $0 < 1 - \mu\alpha \leq 1$, in the leaky LMS algorithm shown below.

$$\mathbf{w}(k) = (1 - \mu\alpha)\mathbf{w}(k-1) - f(\mathbf{u}(n), e(n), \mu)$$

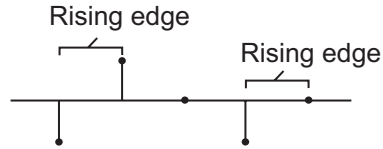
Enter the initial filter weights, $\mathbf{w}(0)$, as a vector or a scalar in the Initial value of filter weights text box. When you enter a scalar, the block uses the scalar value to create a vector of filter weights. This vector has length equal to the filter length and all of its values are equal to the scalar value.

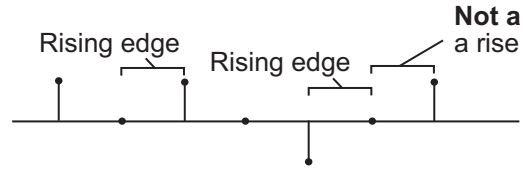
When you select the Adapt port check box, an Adapt port appears on the block. When the input to this port is nonzero, the block continuously updates the filter weights. When the input to this port is zero, the filter weights remain at their current values.

When you want to reset the value of the filter weights to their initial values, use the Reset input parameter. The block resets the filter weights whenever a reset event is detected at the Reset port. The reset signal rate must be the same rate as the data signal input.

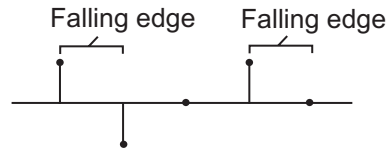
From the Reset input list, select None to disable the Reset port. To enable the Reset port, select one of the following from the Reset input list:

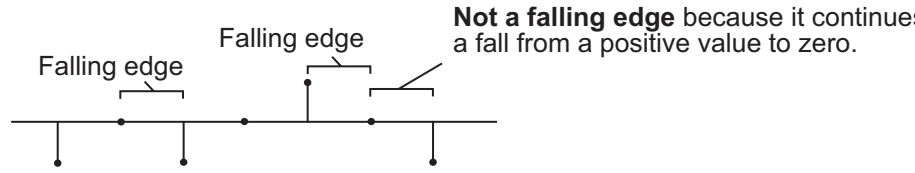
- Rising edge — Triggers a reset operation when the Reset input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following figure)





- Falling edge — Triggers a reset operation when the Reset input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)



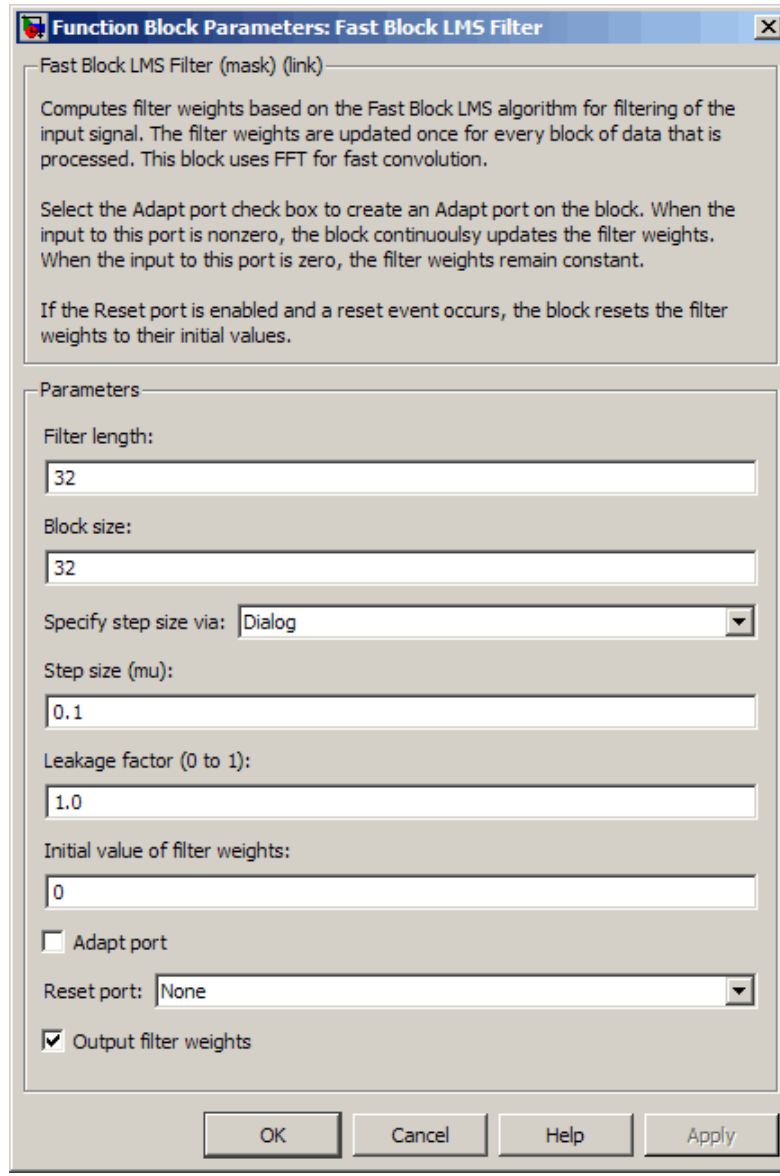


- Either edge — Triggers a reset operation when the Reset input is a Rising edge or Falling edge (as described above)
- Non-zero sample — Triggers a reset opera tion at each sample time that the Reset input is not zero

Note When running simulations in the Simulink MultiTasking mode, sample-based reset signals have a one-sample latency, and frame-based reset signals have one frame of latency. Thus, there is a one-sample or one-frame delay between the time the block detects a reset event, and when it applies the reset. For more information on latency and the Simulink tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Models with Multiple Sample Rates” in the Real-Time Workshop User’s Guide .

Select the Output filter weights check box to create a Wts port on the block. For each iteration, the block outputs the current updated filter weights from this port.

Dialog Box


Function Block Parameters: Fast Block LMS Filter
Fast Block LMS Filter (mask) (link)
Computes filter weights based on the Fast Block LMS algorithm for filtering of the input signal. The filter weights are updated once for every block of data that is processed. This block uses FFT for fast convolution.
Select the Adapt port check box to create an Adapt port on the block. When the input to this port is nonzero, the block continuoulsy updates the filter weights. When the input to this port is zero, the filter weights remain constant.
If the Reset port is enabled and a reset event occurs, the block resets the filter weights to their initial values.
Parameters
Filter length:
32
Block size:
32
Specify step size via:
Dialog
Step size (mu):
0.1
Leakage factor (0 to 1):
1.0
Initial value of filter weights:
0
Adapt port
Reset port:
None
Output filter weights
OK
Cancel
Help
Apply

Filter length

Enter the length of the FIR filter weights vector. The sum of the block size and the filter length must be a power of 2.

Block size

Enter the number of samples to acquire before the filter weights are updated. The input frame length must be an integer multiple of the block size. The sum of the block size and the filter length must be a power of 2.

Specify step-size via

Select Dialog to enter a value for mu, or select Input port to specify mu using the Step-size input port.

Step-size (mu)

Enter the step-size. Tunable.

Leakage factor (0 to 1)

Enter the leakage factor, $0 < 1 - \mu\alpha \leq 1$. Tunable.

Initial value of filter weights

Specify the initial values of the FIR filter weights.

Adapt port

Select this check box to enable the Adapt input port.

Reset input

Select this check box to enable the Reset input port.

Output filter weights

Select this check box to export the filter weights from the Wts port.

References

Hayes, M.H. Statistical Digital Signal Processing and Modeling. New York: John Wiley & Sons, 1996.

Supported Data Types

- Double-precision floating point
- Single-precision floating point

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point
Desired	• Must be the same as Input for ßoating-point signals • Must be any Þxed-point data type when Input is Þxed point
Step-size	• Must be the same as Input for ßoating-point signals • Must be any Þxed-point data type when Input is Þxed point
Adapt	• Double-precision ßoating point • Single-precision ßoating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Reset	• Double-precision ßoating point • Single-precision ßoating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Must be the same as Input for ßoating-point signals • Must be the same as Desired for Þxed-point signals

Port	Supported Data Types
Error	• Must be the same as Input for ßoating-point signals • Must be the same as Desired for Þxed-point signals
Wts	• Must be the same as Input for ßoating-point signals • Obeys the Weights parameter for Þxed-point signals

Purpose

Compute fast Fourier transform (FFT) of input

Library

Transforms

dspxfrm3

Description

The FFT block computes the fast Fourier transform (FFT) of each row of a sample-based 1-by-P input vector, u, or across the first dimension (P) of an N-D input array, u. When you select the Inherit FFT length from input dimensions check box, P must be an integer power of two, and the FFT length, M, is set equal to P. If you do not select the check box, P can be any length, and the value of the FFT length parameter must be a positive integer power of two. For user-specified FFT lengths, when M does not equal P, zero padding or modulo-M data wrapping occurs before the FFT operation, as per Orfanidis [1]:

```
y = fft(u,M)   % P <= M

y(:,l) = fft(datawrap(u(:,l),M))   % P > M; l = 1,...,N
```

When the input length, P, is greater than the FFT length, M, you may see magnitude increases in your FFT output. These magnitude increases occur because the FFT block uses modulo-M data wrapping to preserve all available input samples.

To avoid such magnitude increases, you can truncate the length of your input sample, P, to the FFT length, M. To do so, place a Pad block before the FFT block in your model.

The kth entry of the lth output channel, y(k, l), equals the kth point of the M-point discrete Fourier transform (DFT) of the lth input channel:

$$y(k,l) = \sum_{p=1}^{P} u(p,l) e^{-j2\pi(p-1)(k-1)/M} \qquad k = 1,\ldots,M$$

This block supports real and complex floating-point and fixed-point inputs.

Input and Output Characteristics

The following table describes val id inputs to the FFT block, their corresponding outputs, and the dimension along which the block computes the DFT.

- Valid inputs to the FFT block. Inputs can be real- or complex-valued, and must be in linear order.
- The dimension along which the block computes the DFT.
- The corresponding block output cha racteristics. The output port rate must equal the input port rate.

Valid Block Inputs	Dimension Along Which Block Computes DFT	Corresponding Block Output Characteristics
N-D array	First dimensi on	• Sample based. • Complex valued. • N-D array with the size of the Þrst dimension equal to the FFT length, M, and all other dimensions the same size as the input. • Each output column contains the M-point DFT of the corresponding input channel in linear or bit-reversed order.

Valid Block Input s	Dimension Along W hich Block Computes DF T	Corresponding Bl ock Output Character istics
Sample-based 1-by-P row vector	Row	• Sample based. • Complex valued. • 1-by-M row vector. • Each output row contains the M-point DFT of th e corresponding input channel in line ar or bit-reversed o rder.
Unoriented length- P 1-D vector	Vector	Unoriented, length- M, complex-valued 1-D output vector containing M-point DFT of input in linear or bit-reversed order.

Selecting the Twiddle Factor Computation Method

The Twiddle factor computation parameter determines how the block computes the necessary sine and cosine terms to calculate the term $e^{-j2\pi(p-1)(k-1)/M}$, shown in the first equation of this block reference page.

The block only supports Table lookup mode for fixed-point signals.

The Twiddle factor computation parameter has two settings, each with its advantages and disadvantages, as described in the following table.

Twiddle Factor Computation Parameter Setting	Sine and Cosine Computation Method	Effect on Block Performance
Table lookup	The block computes and stores the trigonometric values before the simulation starts and retrieves them during the simulation. When you generate code from the block, the processor running the generated code stores the trigonometric values computed by the block, and retrieves the values during code execution.	The block usually runs much more quickly, but requires extra memory for storing the precomputed trigonometric values. You can optimize the table for memory consumption or speed, as described in "Optimizing the Table of Trigonometric Values" on page 2-482.
Trigonometric fcn	The block computes sine and cosine values during the simulation. When you generate code from the block, the processor running the generated code computes the sine and cosine values while the code runs.	The block usually runs more slowly, but does not need extra data memory. For code generation, the block requires a support library to emulate the trigonometric functions, increasing the size of the generated code.

Optimizing the Table of Trigonometric Values

When you set the Twiddle factor computation parameter to Table lookup , you also need to set the Optimize table for parameter.

This parameter optimizes the table of trigonometric values for speed or memory by varying the number o f table entries as summarized in the following table.

Optimize Table for Parameter Setting	Number of Table Entries for N-Point FFT	Memory Required for Single-Precision, 512-Point FFT
Speed	3N/4 — ßoating point N — Þxed point	$\left(\frac{3\times 512}{4}\,\text{table entries}\right)\times\left(4\,\frac{\text{bytes}}{\text{table entry}}\right)$ $=1536\,\text{bytes}$
Memory	N/4 — ßoating point Not supported for Þxed point	$\left(\frac{512}{4}\,\text{table entries}\right)\times\left(4\,\frac{\text{bytes}}{\text{table entry}}\right)$ $=512\,\text{bytes}$

Ordering Output Column Entries

You can set the Output in bit-reversed order parameter to specify the ordering of the column elements of the block output. If you select the Output in bit-reversed order check box, the output appears in bit-reversed order. If you clear the Output in bit-reversed order check box, the output appears in linear order.

Note The FFT block calculates its output in bit-reversed order. Linearly ordering the FFT block output requires an extra bit-reversal operation. Thus, in many situations, you can increase the speed of the FFT block by selecting the Output in bit-reversed order check box.

For more information ordering of the output, see “Linear and Bit-Reversed Output Order”.

Algorithms Used for FFT Computation

Depending on whether the block’s input is real- or complex-valued and whether you want the output in linear or bit-reversed order, the block uses one or more of the following algorithms as summarized in the following table:

- Bit-reversal operation
- Double-signal algorithm
- Half-length algorithm
- Radix-2 decimation-in-time (DIT) algorithm
- Radix-2 decimation-in-frequency (DIF) algorithm

Complexity of Input	Output Ordering	Algorithms Used for FFT Computation
Complex	Linear	Bit-reversal operation and radix-2 DIT
Complex	Bit-reversed	Radix-2 DIF
Real	Linear	Bit-reversal operation and radix-2 DIT in conjunction with the half-length and double-signal algorithms
Real	Bit-reversed	Radix-2 DIF in conjunction with the half-length and double-signal algorithms

The efficiency of the FFT algorithm can be enhanced for real input signals by forming complex-valued sequences from the real-valued

sequences prior to the computation of the DFT. When there are 2N+1 real input channels, the FFT block forms these complex-valued sequences by applying the double-signal algorithm to the first 2N input channels, and the half-length algorithm to the last odd-numbered channel.

For real input signals with fixed-point data types, it is possible to see different numerical results in the output of the last odd numbered channel, even when all input channels are identical. This numerical difference results from differences in the double-signal algorithm and the half-length algorithm.

You can eliminate this numerical difference in two ways:

- Using full precision arithmetic for fixed-point input signals
- Changing the input data type to floating point

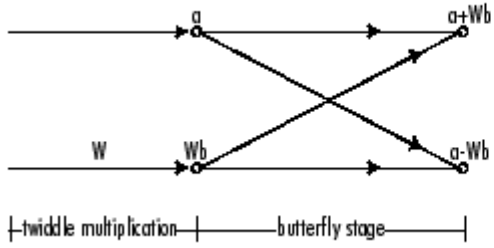
For more information on the double-signal and half-length algorithms, see Proakis [2]. "Efficient Computation of the DFT of Two Real Sequences" on page 475 describes the double signal algorithm. "Efficient Computation of the DFT of a 2N-Point Real Sequence" on page 476 describes the half-length algorithm.

Fixed-Point Data Types

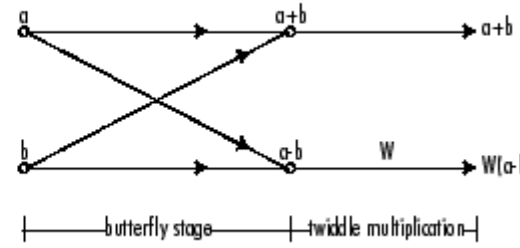
The following diagrams show the data types used within the FFT block for fixed-point signals. You can set the sine table, accumulator, product output, and output data types displayed in the diagrams in the FFT block dialog as discussed in "Dialog Box" on page 2-486.

The block first casts inputs to the FFT block to the output data type and stores them in the output buffer. Each butterfly stage then processes signals in the accumulator data type, and the block then casts the final output of the butterfly back into the output data type. The block multiplies in a twiddle factor before each butterfly stage in a decimation-in-time FFT, and after each butterfly stage in a decimation-in-frequency FFT.

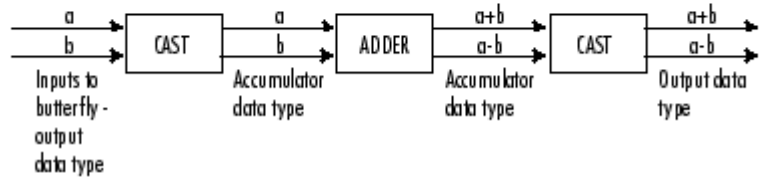
Decimation-in-Time FFT



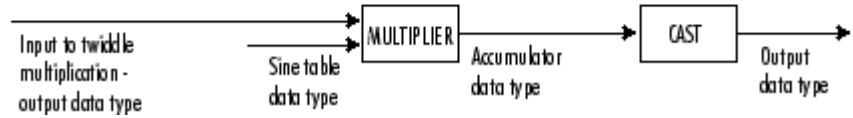
Decimation-in-Frequency FFT



Butterfly Stage Data Types



Twiddle Multiplication Data Types



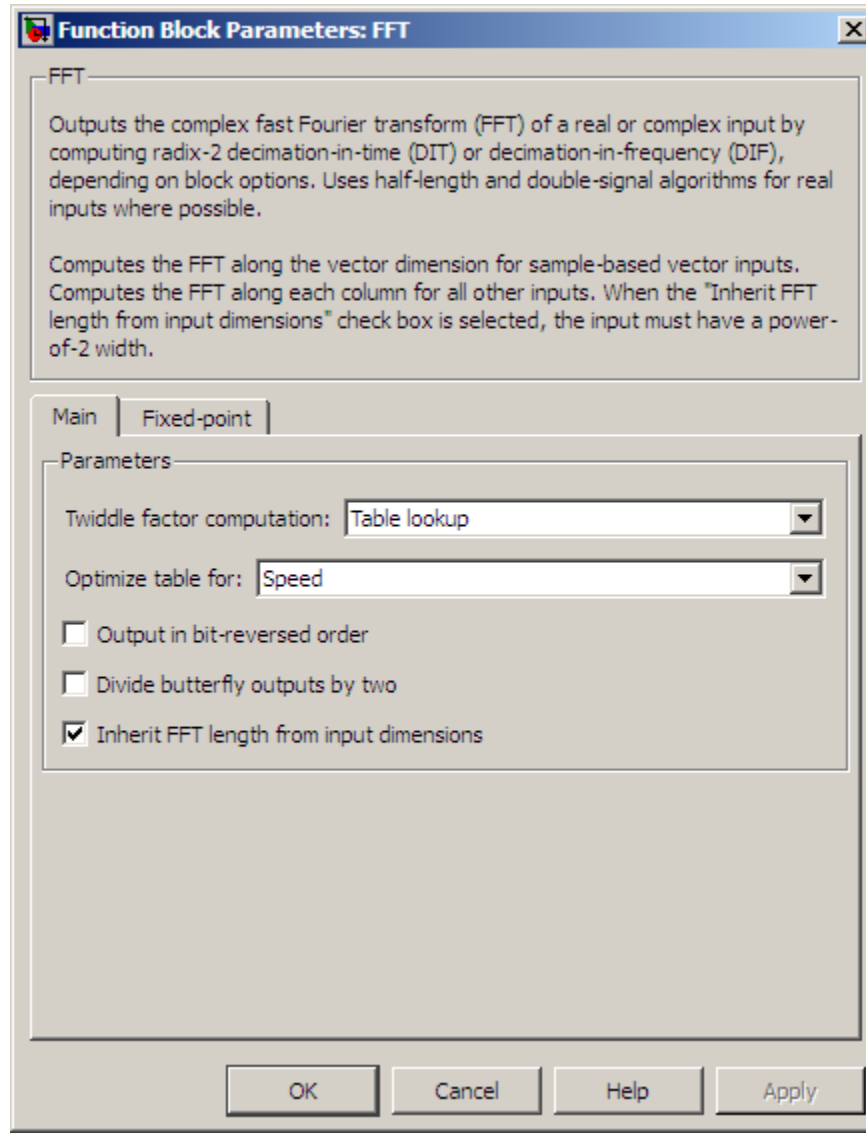
The output of the multiplier appears in the accumulator data type because both of the inputs to the multiplier are complex. For details on the complex multiplication performe d, see “Multiplication Data Types”.

Examples

See the section on “Transforming Time-Domain Data into the Frequency Domain” in the Signal Processing Blockset User’s Guide .

Dialog Box

The Main pane of the FFT block dialog appears as follows.



Twiddle factor computation

Specify the computation method of the term $e^{-j2\pi(p-1)(k-1)/M}$, shown in the first equation of this block reference page.

In Table lookup mode, the block computes and stores the sine and cosine values before the simulation starts.

In Trigonometric fcn mode, the block computes the sine and cosine values during the simulation. See "Selecting the Twiddle Factor Computation Method" on page 2-481.

This parameter must be set to Table lookup for fixed-point signals.

Optimize table for

Select the optimization of the table of sine and cosine values for Speed or Memory. This parameter becomes available only when you set the Twiddle factor computation parameter to Table lookup. See "Selecting the Twiddle Factor Computation Method" on page 2-481.

This parameter must be set to Speed for fixed-point signals.

Output in bit-reversed order

Designate the order of the output channel elements relative to the ordering of the input elements. When you select this check box, the output channel elements appear in bit-reversed order relative to the input ordering. Otherwise, the output column elements appear in linear order relative to the input ordering.

Linearly ordering the output requires extra data sorting manipulation, so in some situations it might be better to output in bit-reversed order.

Divide butterfly outputs by two

When you select this parameter, the output of each butterfly of the FFT is divided by two. When you do not select this parameter, the block does not scale the output.

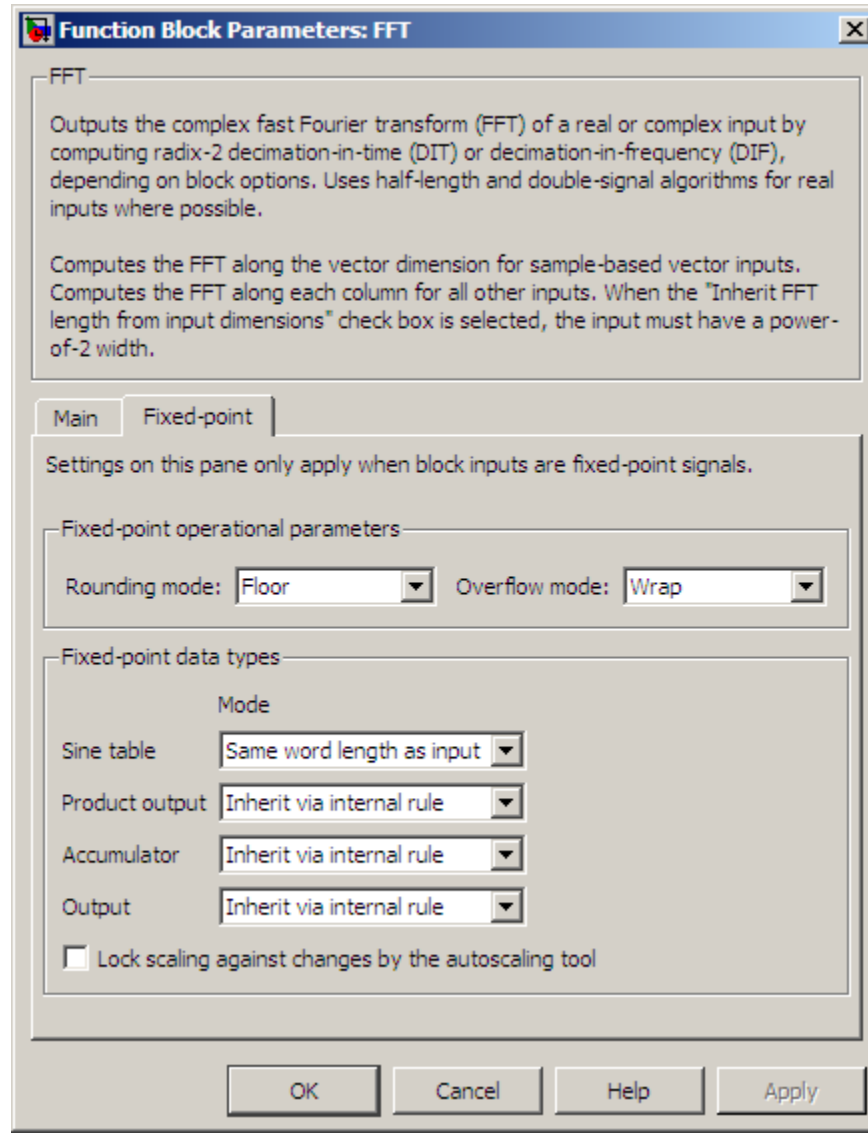
Inherit FFT length from input dimensions

Select to inherit the FFT length fr om the input dimensions. When you select this parameter, the input length P must be a power of two. When you do not select this parameter, the FFT length parameter becomes available.

FFT length

Specify a power-of-two FFT leng th. This parameter becomes available only when you do not select the Inherit FFT length from input dimensions parameter.

The Fixed-point pane of the FFT block dialog appears as follows.


Function Block Parameters: FFT
FFT
Outputs the complex fast Fourier transform (FFT) of a real or complex input by computing radix-2 decimation-in-time (DIT) or decimation-in-frequency (DIF), depending on block options. Uses half-length and double-signal algorithms for real inputs where possible.
Computes the FFT along the vector dimension for sample-based vector inputs. Computes the FFT along each column for all other inputs. When the "Inherit FFT length from input dimensions" check box is selected, the input must have a power-of-2 width.
Main
Fixed-point
Settings on this pane only apply when block inputs are fixed-point signals.
Fixed-point operational parameters
Rounding mode:
Floor
Overflow mode:
Wrap
Fixed-point data types
Mode
Sine table
Same word length as input
Product output
Inherit via internal rule
Accumulator
Inherit via internal rule
Output
Inherit via internal rule
Lock scaling against changes by the autoscaling tool
OK
Cancel
Help
Apply

Rounding mode

Select the rounding mode for fixed-point operations. The sine table values do not obey this parameter; instead, they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations. The sine table values do not obey this parameter; instead, they are always saturated.

Sine table

Choose how you specify the word length of the values of the sine table. The fraction length of the sine table values always equals the word length minus one:

- When you select Same word length as input, the word length of the sine table values match that of the input to the block.
- When you select Specify word length, you can enter the word length of the sine table values, in bits.

The sine table values do not obey the Rounding mode and Overflow mode parameters; instead, they are always saturated and rounded to Nearest.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-44 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Inherit via internal rule, the block calculates the product output word length and fraction length automatically. For information about how the product output word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as input, these characteristics match those of the input to the block.

- When you select Binary point scaling , you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths. See “Fixed-Point Data Types” on page 2-44 and “Mu ltiplication Data Types” for illustrations depicting the use of the accumulator data type in this block:

- When you select Inherit via internal rule , the block calculates the accumulator word length and fraction length automatically. For informati on about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Inherit via internal rule , the block calculates the output word length and fraction length

automatically. The internal rule first calculates an ideal output word length and fraction length using the following equations:

$$WL_{ideal\ output} = WL_{input} + floor(\log_2(FFT\ length - 1)) + 1$$

$$FL_{ideal\ output} = FL_{input}$$

Using these ideal results, the internal rule then selects word lengths and fraction lengths that are appropriate for your hardware. For more information, see “Inherit via Internal Rule”.

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

References

[1] Orfanidis, S. J. Introduction to Signal Processing. Upper Saddle River, NJ: Prentice Hall, 1996, p. 497.

[2] Proakis, John G. and Dimitris G. Manolakis. Digital Signal Processing, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 1996.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

DCT	Signal Processing Blockset
IFFT	Signal Processing Blockset
Pad	Signal Processing Blockset
fft	MATLAB
ifft	MATLAB
bitrevorder	Signal Processing Toolbox

Filter Realization Wizard

Purpose

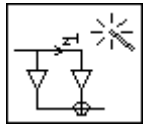
Construct filter realizations using Digital Filter block or Sum, Gain, and Delay blocks

Library

Filtering / Filter Designs

dsparch4

Description



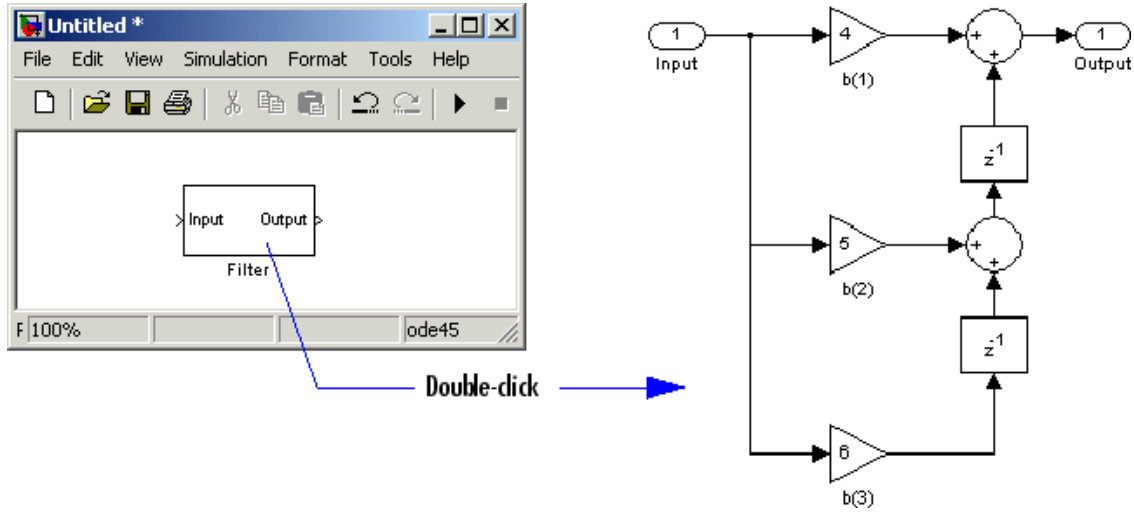
Note Use this block to implement fixed-point or floating-point digital filters using Sum, Gain, and Delay blocks or the Digital Filter block. You can either design a filter by using the block's filter design and analysis parameters, or import the coefficients of a filter you have designed elsewhere.

The following blocks also implement digital filters, but serve slightly different purposes:

- Digital Filter — Use to implement floating-point or fixed-point filters that you have already designed
- Digital Filter Design — Use to design, analyze, and then implement floating-point filters.

The Filter Realization Wizard is a tool for automatically implementing a digital filter. You must specify a filter, its structure, and the data types for the filter's inputs, outputs, and computations. The filter can support double-precision, single-precision, or fixed-point data types.

The Filter Realization Wizard can implement a digital filter in one of two ways. It can use a Digital Filter block, or it can create a subsystem block that implements the specified filter using Sum, Gain, and Delay blocks. If the Filter Realization Wizard creates a Digital Filter block, double-click the block to open the Block Parameters: Filter dialog box. If it creates a subsystem, double-click the subsystem block to see the filter implementation as shown in the figure below.



The subsystem block applies the specified filter to any sample-based input signal, or any frame-based row vector signal, and outputs the result. For more information about filter implementation, see "Specifying the Filter Implementation" on page 2-498.

The parameters of the Filter Realization Wizard are a part of a larger GUI, the Filter Design and Analysis Tool (`fdatool`), from the Signal Processing Toolbox product. You can use the tools in FDATool to design and analyze your filter, and then use the Filter Realization Wizard parameters to implement the filter in your models.

Sections of This Reference Page

- "Valid Inputs and Corresponding Outputs" on page 2-496
- "Specifying the Filter and Its Data Type Support" on page 2-496
- "Supported Filter Structures" on page 2-498
- "Specifying the Filter Implementation" on page 2-498
- "Corresponding Method for dfilt" on page 2-500
- "Dialog Box" on page 2-501

- “References” on page 2-503
- “Supported Data Types” on page 2-503
- “See Also” on page 2-503

Valid Inputs and Corresponding Outputs

When the Filter Realization Wizard implements the specified filter by creating a new subsystem block, the block applies the specified filter to an input signal and outputs the result.

Valid Inputs

The subsystem block accepts inputs that are

- Sample-based vectors and matrices
- Frame-based row vectors (non recursive structures only)

Corresponding Outputs

The output of the subsystem block has the same dimensions and frame status as the input.

What Is Considered an Independent Channel

The subsystem block treats each element of a vector or matrix as an independent channel.

Specifying the Filter and Its Data Type Support

To specify a purely double-precision filter, you can either design a filter using the Design Filter panel, or import a filter using the Import Filter panel. (You can import dfilt filter objects as well as vectors of filter coefficients designed using Signal Processing Toolbox functions and Filter Design Toolbox functions.)

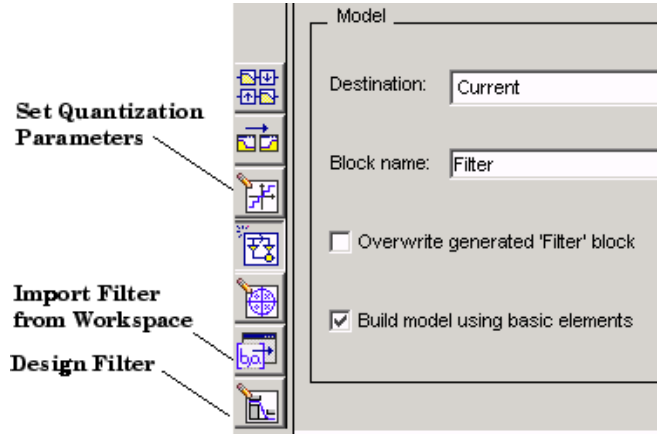
You can also specify a fixed-point filter or a single-precision filter. You can specify such filters by using the Set Quantization Parameters panel, which requires the Filter Design Toolbox product.

Note Running a model containing implementations of fixed-point filters requires the Simulink Fixed Point product, but you can still edit models containing such filter implementations without it. See the Simulink Fixed Point documentation for more information.

See the following topics to learn how to use the panels to specify your filter:

- For more information on the Design Filter panel, see in the Signal Processing Toolbox documentation.
- For more information on the Import Filter panel, see "Importing a Filter Design" in the Signal Processing Toolbox documentation.
- For more information on the Set Quantization Parameters panel, see "Switching FDATool to Quantization Mode" in the Filter Design Toolbox documentation.

To open a panel, click the appropriate button in the lower-left corner of FDATool.



Supported Filter Structures

The Filter Realization Wizard supports the following structures:

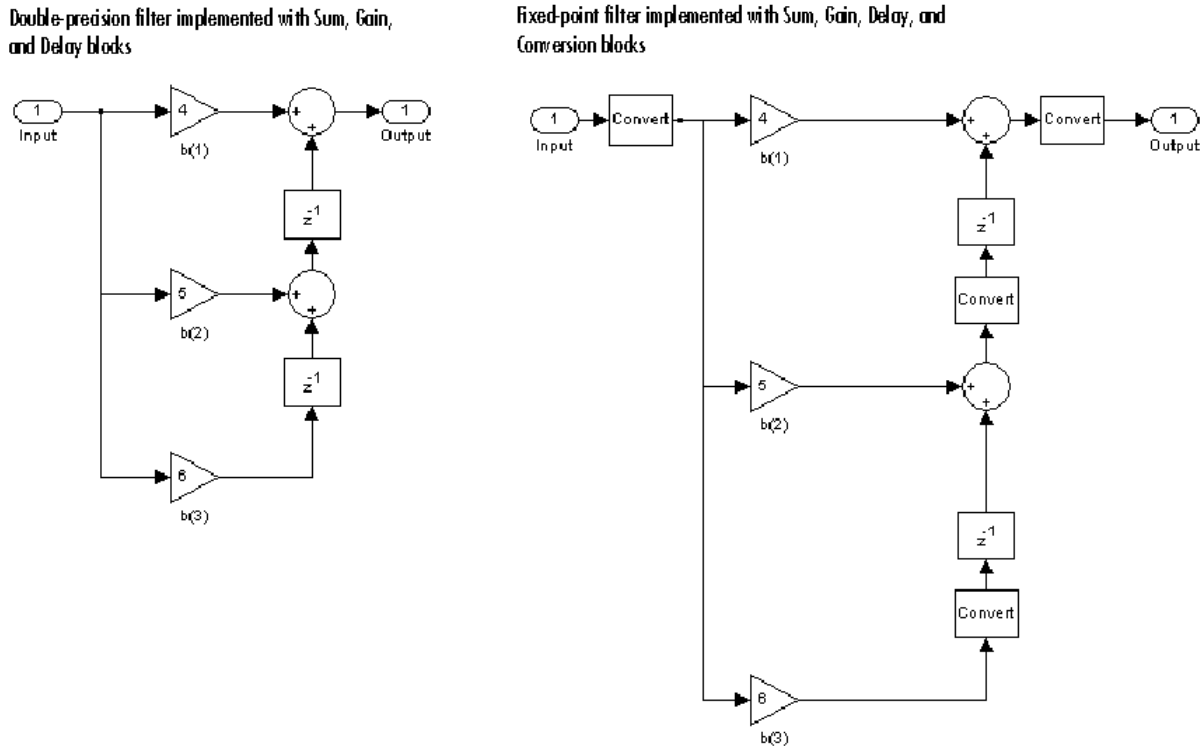
- Direct form I
- Direct form I, second-order sections
- Direct form I transposed
- Direct form I transposed, second-order sections
- Direct form II
- Direct form II, second-order sections
- Direct form II transposed
- Direct form II transposed, second-order sections
- Direct form FIR
- Direct form FIR transposed
- Direct form symmetric FIR
- Direct form antisymmetric FIR
- Lattice all-pass
- Lattice AR
- Lattice ARMA
- Lattice MA for maximum phase
- Lattice MA for minimum phase
- Cascade
- Parallel

Specifying the Filter Implementation

You can determine how the Filter Realization Wizard models the specified filter using the Build model using basic elements check box. When you select this check box, the Filter Realization Wizard

creates a subsystem block that implements your Þlter using Sum, Gain, and Delay blocks. When you clear thi s check box, the Fil ter Realization Wizard uses a Digital Filter block to implement your Þlter. The Build model using basic elements check box is only available when your Þlter can be implemented using a Digital Filter block.

If you have Signal Processing Bloc kset software, Signal Processing Toolbox software, and Filter Design T oolbox software installed on your system, the Filter Realization Wizard can generate a subsystem that represents either a double-precision or Þxed-point Þlter. You must install the Simulink Fixed Point product to simulate a Þxed-point Þlter. You can still edit the blocks used to implement the Þlter without installing the Simulink Fixed Point product.



Implementations of Double-Precision and Fixed-Point Filters

Corresponding Method for dfilt

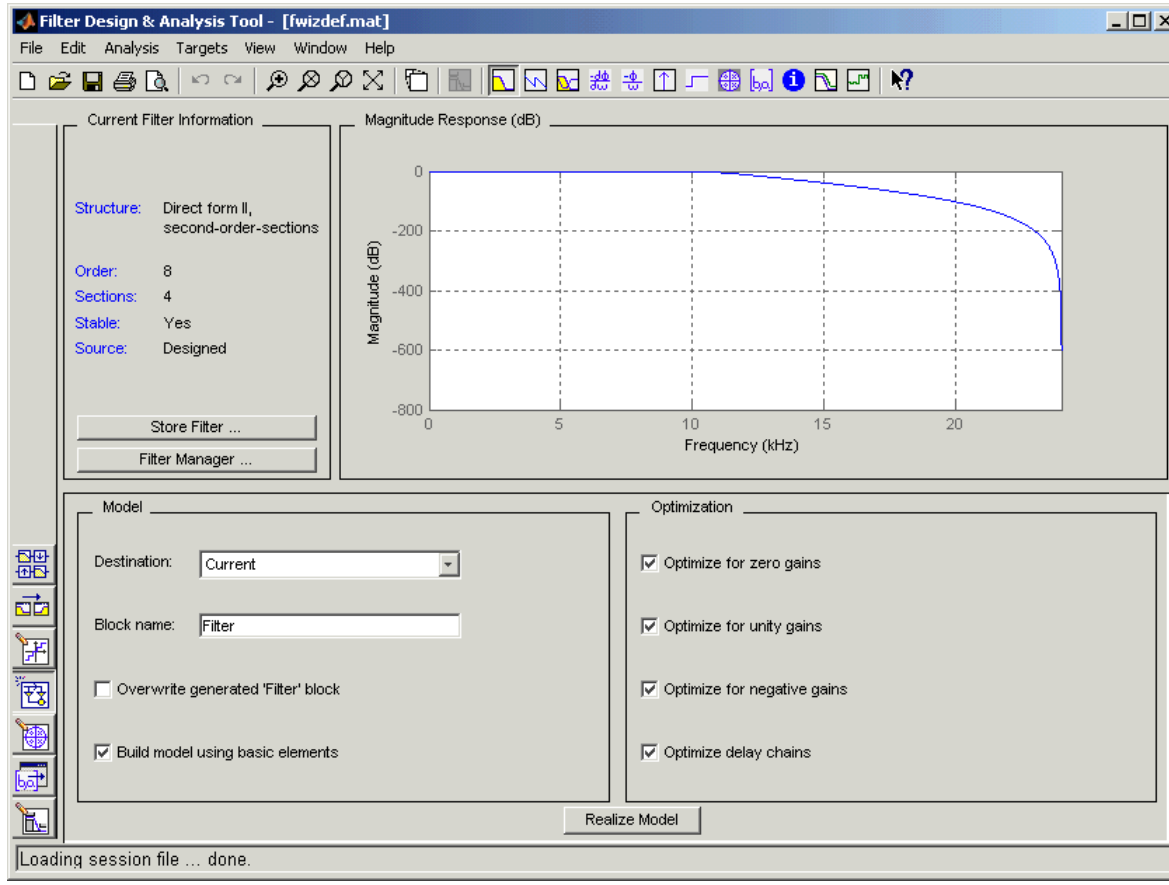
The Signal Processing Toolbox dfilt (digital filter) object in has a method, realizemdl, that allows you to access the capabilities of the Filter Realization Wizard from the command line.

For more information about the realizemdl method, see the following:

- The topic on "Methods" in the dfilt reference page in the Signal Processing Toolbox documentation
- The realizemdl reference page in the Filter Design Toolbox documentation

Dialog Box

Note The following parameters for the Filter Realization Wizard are in the Realize Model pane of the Filter Design and Analysis Tool (FDATool) GUI. To open different panels of FDATool, click the different buttons at the lower-left corner. For more information about relevant panels, see “Specifying the Filter and Its Data Type Support” on page 2-496.



Destination

Specify where the new filter block should be created. This can be in a new model or in the current (most recently selected) model.

Block Name

Enter the name of the new filter block.

Overwrite generated block "Filter" block

When selected, the block overwrites any filter block in the current model with the name specified in the Block Name parameter. This parameter is enabled when the Destination parameter is set to Current.

Build model using basic elements

Select this check box to implement your filter using Sum, Gain, and Delay blocks. Clear this check box to implement your filter using the Digital Filter block. This parameter is only available when your filter can be modeled using the Digital Filter block.

Note that when your filter is implemented using Sum, Gain, and Delay blocks, inputs to the filter must be sample based.

Optimize for zero gains

When selected, the block removes zero-gain paths from the filter structure. For an example, see "Optimizing the Filter Structure".

Optimize for unity gains

When selected, the block substitutes gains equal to 1 with a wire (short circuit). For an example, see "Optimizing the Filter Structure".

Optimize for negative gains

When selected, the block substitutes gains equal to -1 with a wire (short circuit), and changes the corresponding sums to subtractions. For an example, see "Optimizing the Filter Structure".

Optimize delay chains

When selected, the block substitutes any delay chains made up of n unit delays with a single delay by n. For an example, see "Optimizing the Filter Structure".

Realize Model

Click to create a subsystem block that implements the specified filter using Sum, Gain, and Delay blocks. To see the filter implementation, double-click the subsystem block. The subsystem block applies the specified filter to any sample-based input signal or frame-based row vector signal, and outputs the result.

Note For more information about relevant parameters in other panels of FDATool, see "Specifying the Filter and Its Data Type Support" on page 2-496.

References

Oppenheim, A. V. and R. W. Schafer. Discrete-Time Signal Processing. Englewood Cliffs, NJ: Prentice Hall, 1989.

Proakis, J. and D. Manolakis. Digital Signal Processing. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

Supported Data Types

- Double-precision floating point
- Single-precision floating point — Supported only when you install the following products: Filter Design Toolbox and Simulink Fixed Point
- Fixed point (signed and unsigned) — Supported only when you install the following products: Filter Design Toolbox, "Simulink Fixed Point", and Fixed-Point Toolbox

See Also

Digital Filter	Signal Processing Blockset
Digital Filter Design	Signal Processing Blockset
filter	Filter Design Toolbox

realizemdl	Filter Design Toolbox
dfilt	Signal Processing Toolbox
filter	Signal Processing Toolbox

- “Filters” — Examples of when and how to use Signal Processing Blockset Þltering blocks
- “Choosing Between Filter Design Blocks”

Purpose

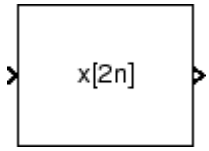
Filter and downsample input signals

Library

Filtering / Multirate Filters

dspmlti4

Description



The FIR Decimation block resamples the discrete-time input at a rate K times slower than the input sample rate, where the integer K is specified by the Decimation factor parameter. This process consists of two steps:

- The block filters the input data using a direct-form FIR filter.
- The block downsamples the filtered data to a lower rate by discarding K–1 consecutive samples following every sample retained.

The FIR Decimation block implements the above FIR filtering and downsampling steps together using a polyphase filter structure, which is more efficient than straightforward filter-then-decimate algorithms. See Fliege [1] for more information.

The FIR filter coefficients parameter specifies the numerator coefficients of the FIR filter transfer function H(z).

$$H(z) = B(z) = b_1 + b_2 z^{-1} + \ldots + b_m z^{-(m-1)}$$

The length- m coefficient vector, [b(1) b(2) ... b(m)], can be generated by one of the filter design functions in Signal Processing Toolbox software, such as the fir1 function used in Example 1 below. The filter should be lowpass with normalized cutoff frequency no greater than 1/K. All filter states are internally initialized to zero.

The FIR Decimation block supports real and complex floating-point and fixed-point inputs, except for complex unsigned fixed-point inputs. This block supports triggered subsystems when you select Maintain input frame rate for the Framing parameter.

Sample-Based Operation

An M-by-N sample-based matrix input is treated as M*N independent channels, and the block decimates each channel over time. The output sample period is K times longer than the input sample period ($T_{so} = KT_{si}$), and the input and output sizes are identical.

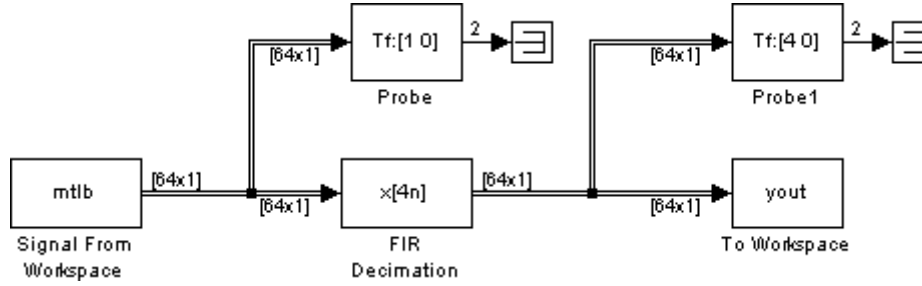
Frame-Based Operation

An M_i-by-N frame-based matrix input is treated as N independent channels, and the block decimates each channel over time. The Framing parameter determines how the block adjusts the rate at the output to accommodate the reduced number of samples. There are two available options:

- Maintain input frame size

  The block generates the output at the decimated rate by using a proportionally longer frame period at the output port than at the input port. For decimation by a factor of K, the output frame period is K times longer than the input frame period ($T_{fo} = KT_{fi}$), but the input and output frame sizes are equal.

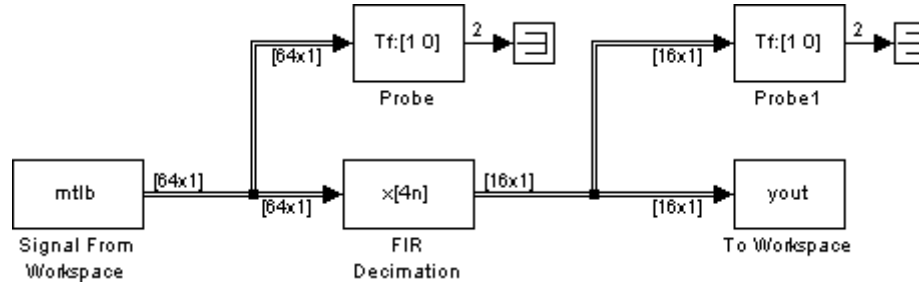
  The following model shows a single-channel input with a frame period of 1 second (Sample time = 1/64 and Samples per frame = 64 in the Signal From Workspace block) being decimated by a factor of 4 to a frame period of 4 seconds. The input and output frame sizes are identical.



- Maintain input frame rate

The block generates the output at the decimated rate by using a proportionally smaller frame size than the input. For decimation by a factor of K, the output frame size is K times smaller than the input frame size ($M_o = M_i/K$), but the input and output frame rates are equal. The input frame size, M_i, must be a multiple of the decimation factor, K.

The following model shows a single-channel input of frame size 64 being decimated by a factor of 4 to a frame size of 16. The block's input and output frame rates are identical.



Latency

The FIR Decimation block has zero-tasking latency for all single-tasking and multitasking cases, except one:

Sample-Based Latency	Frame-Based Latency — Maintain input frame rate	Frame-Based Latency — Maintain input frame size
None	None	One frame (M_i samples)

Zero-tasking latency means that the block propagates the first filtered input sample (received at t=0) as the first output sample, followed by filtered input samples K+1, 2K+1, and so on.

In cases of one-frame latency, the first M_i output rows contain zeros, where M_i is the input frame size. The first filtered input sample (first

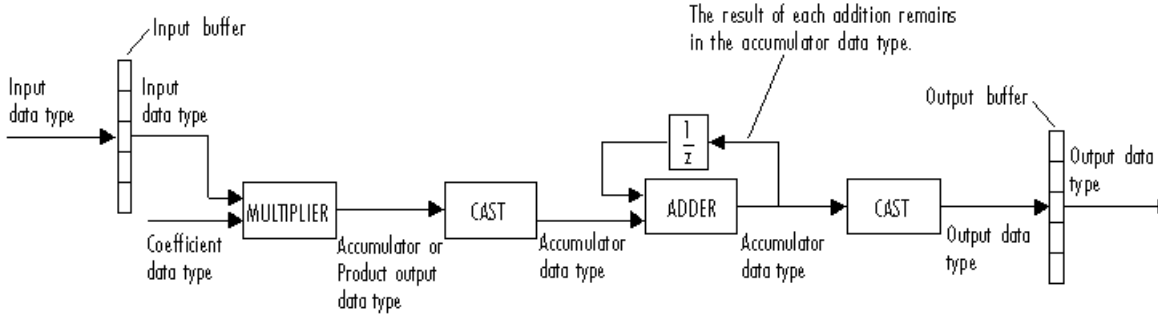
Þltered row of the input matrix) appears in the output as sample M_i+1, followed by Þltered input samples K+1, 2K+1, and so on.

When the block exhibits la tency, enter a value in the Output buffer initial conditions text box to specify the value to output at the output port until the Þrst Þltered input sample is available. The default initial condition value is 0.

Note For more information on latency and the Simulink tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Models with Multiple Sample Rates” in the Real-Time Workshop User’s Guide .

Fixed-Point Data Types

The following diagram shows the data types used within the FIR Decimation block for Þxed-point signals.



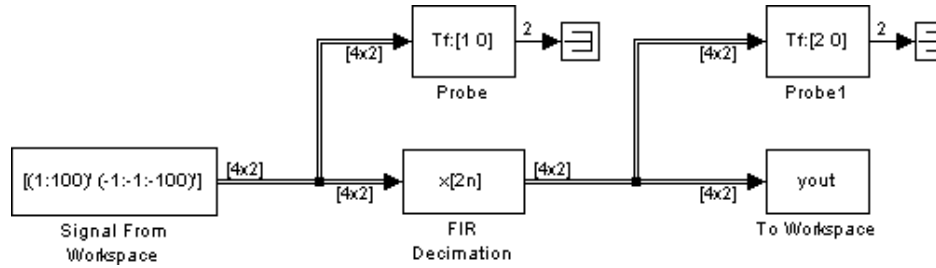
You can set the coefÞcient, product output, accumulator, and output data types in the block dialog as discussed in “Dialog Box” on page 2-511. This diagram shows that data is stored in the input buffer in the same data type and scaling as the inp ut. Filtered data is stored in the output buffer in the output data type and scaling that you set in the block dialog. Any initial conditions are also stored in the output buffer in the output data type and scaling you set in the block dialog.

The output of the multiplier is in the product output data type when at least one of the inputs to the multiplier is real. When both of the inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see “Multiplication Data Types”.

Examples

Example 1

Construct the following frame-based model.



Adjust the block parameters as follows:

- Configure the Signal From Workspace block to generate a two-channel signal with frame size of 4 and sample period of 0.25. This represents an output frame period of 1 (0.25*4). The first channel should contain the positive ramp signal 1, 2, ..., 100, and the second channel should contain the negative ramp signal -1, -2, ..., -100.
  - Signal = [(1:100)' (-1:-1:-100)']
  - Sample time = 0.25
  - Samples per frame = 4
- Configure the FIR Decimation block to decimate the two-channel input by decreasing the output frame rate by a factor of 2 relative to the input frame rate. Use a third-order filter with normalized cutoff frequency, f_{n0}, of 0.25. (Note that f_{n0} satisfies $f_{n0} \leq 1/K$.)
  - FIR filter coefficients = fir1(3,0.25)

- Downsample factor = 2
- Framing = Maintain input frame size

The filter coefficient vector generated by fir1(3,0.25) is

[0.0386 0.4614 0.4614 0.0386]

or, equivalently,

$$H(z) = B(z) = 0.0386 + 0.04614z^{-1} + 0.04614z^{-2} + 0.0386z^{-3}$$

- Configure the Probe blocks by clearing the Probe width, Probe complex signal, and Probe signal dimensions check boxes (if desired).

This model is multirate because there are at least two distinct sample rates, as shown by the two Probe blocks. To run this model in the Simulink multitasking mode, make the following settings in the Solver pane of the Configuration Parameters dialog box:

- From the Type list, select Fixed-step.
- From the Solver list, select Discrete (no continuous states).
- From the Tasking mode for periodic sample times list, select MultiTasking.
- Set the Stop time to 30.

Run the model and look at the output, yout. The first few samples of each channel are shown below.

```
yout =
         0         0
         0         0
         0         0
         0         0
    0.0386   -0.0386
    1.5000   -1.5000
```

3.5000	-3.5000
5.5000	-5.5000
7.5000	-7.5000
9.5000	-9.5000
11.5000	-11.5000

Since this is a frame-based multirate model, the first four (M_i) output rows are zero. The first filtered input matrix row appears in the output as sample 5 (that is, sample M_i+1).

Example 2

The doc_polyphasedec model illustrates the underlying polyphase implementations of the FIR Decimation block. Run the model and view the results on the scope. The output of the FIR Decimation block is the same as the output of the Polyphase Decimation Filter block.

Example 3

The doc_mrf_nlp model illustrates the use of the FIR Decimation block in a number of multistage multirate filters.

Dialog Box

The FIR Decimation block can operate in two different modes. Select the mode in the Coefficient source group box. If you select

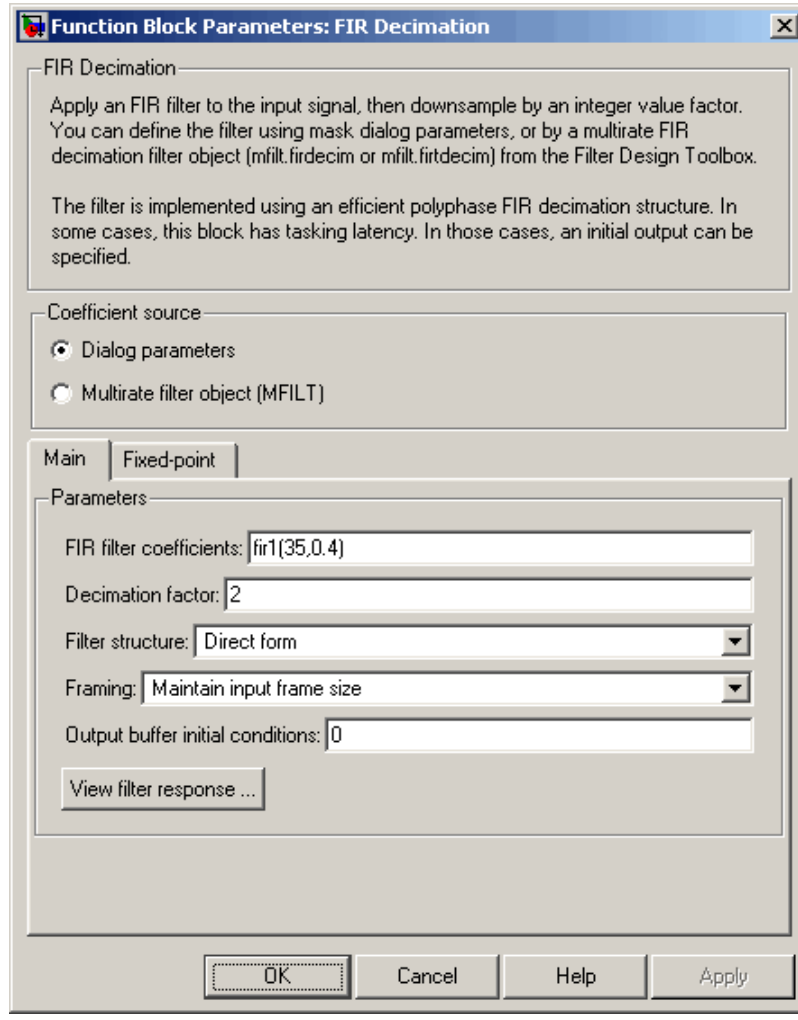
- Dialog parameters, you enter information about the filter such as structure and coefficients in the block mask.
- Multirate filter object (MFILT), you specify the filter using a Filter Design Toolbox mfilt object.

Different items appear on the FIR Decimation block dialog depending on whether you select Dialog parameters or Multirate filter object (MFILT) in the Coefficient source group box. See the following sections for details:

- "Specify Filter Characteristics in Dialog" on page 2-512
- "Specify Multirate Filter Object" on page 2-518

Specify Filter Characteristics in Dialog

The Main pane of the FIR Decimation block dialog appears as follows when Dialog parameters is selected in the Coefficient source group box.



FIR filter coefficients
: Specify the lowpass FIR filter coefficients, in descending powers of z.

Decimation factor
: Specify the integer factor, K, by which to decrease the sample rate of the input sequence.

Filter Structure
: Choose whether to implement a Direct form or Direct form transposed filter.

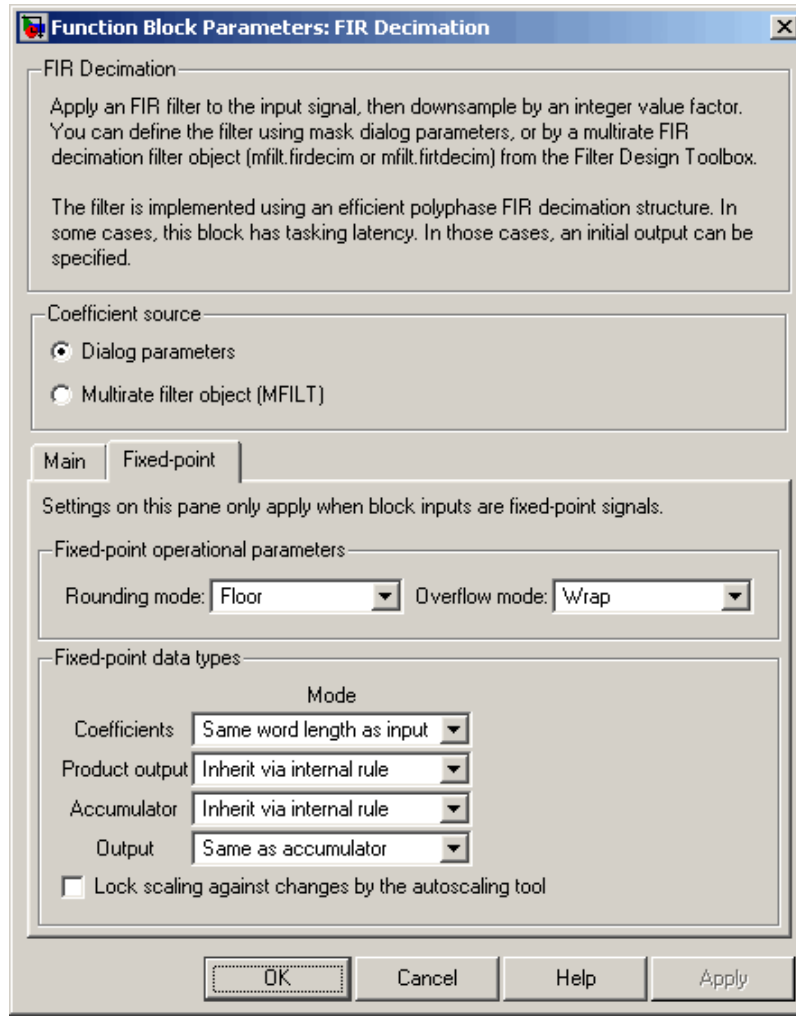
Framing
: For frame-based operation, specify the method by which to implement the decimation; reduce the output frame rate, or reduce the output frame size. This parameter cannot be set to Maintain input frame rate for sample-based signals.

Output buffer initial conditions
: When the block exhibits latency, enter a value in the Output buffer initial conditions text box to specify the value to output at the output port until the first filtered input sample is available. The default initial condition value is zero.

View filter response
: This button opens the Filter Visualization Tool (fvtool) from the Signal Processing Toolbox product and displays the filter response of the filter defined in the block. For more information on FVTool, see the Signal Processing Toolbox documentation.

Note This button is only available when the Filter Design Toolbox product is installed. If you specify a filter in the Multirate filter variable parameter, you must apply the filter by clicking the Apply button before using the View filter response button.

The Fixed point pane of the FIR Decimation block dialog appears as follows when Dialog parameters is specified in the Coefficient source group box.



Rounding mode

Select the rounding mode for fixed-point operations. The filter coefficients do not obey this parameter; they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations. The filter coefficients do not obey this parameter; they are always saturated.

Coefficients

Choose how you specify the word length and the fraction length of the filter coefficients:

- When you select Same word length as input, the word length of the filter coefficients match that of the input to the block. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select Specify word length, you can enter the word length of the coefficients, in bits. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select Binary point scaling, you can enter the word length and the fraction length of the coefficients, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the coefficients. This block requires power-of-two slope and a bias of zero.

The filter coefficients do not obey the Rounding mode and the Overflow mode parameters; they are always saturated and rounded to Nearest.

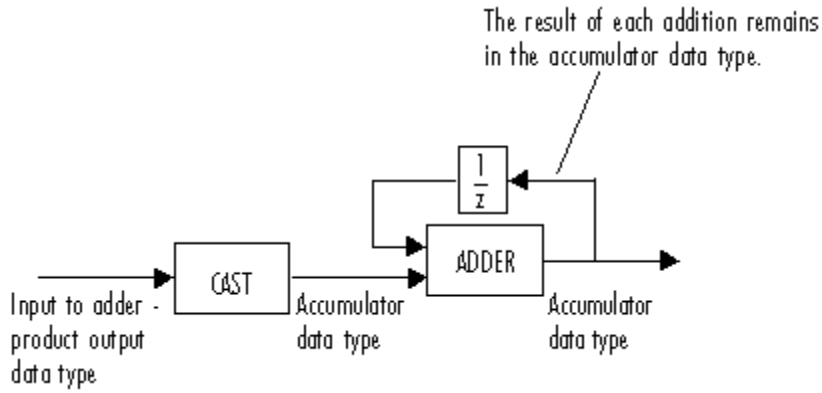
Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Fixed-Point

Data Types” on page 2-508 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Inherit via internal rule, the product output word length and fraction length are calculated automatically. For information on about how the product output word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator



As depicted in this graphic, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to

it. Use this parameter to specify how you would like to designate this accumulator word and fraction lengths.

You also use this parameter to specify the accumulator word and fraction lengths resu lting from a complex-complex multiplication in the block. See “Multiplication Data Types” for more information.

- When you select Inherit via internal rule , the accumulator word length and fraction length are calculated automatically. For informati on about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as accumulator , these characteristics match those of the accumulator.

  A special case occurs when Inherit via internal rule is speciÞed for Accumulator , and block inputs and coef Þcients are complex. In that case, the output word length is one less than the accumulator word length.
- When you select Same as product output , these characteristics match those of the product output.

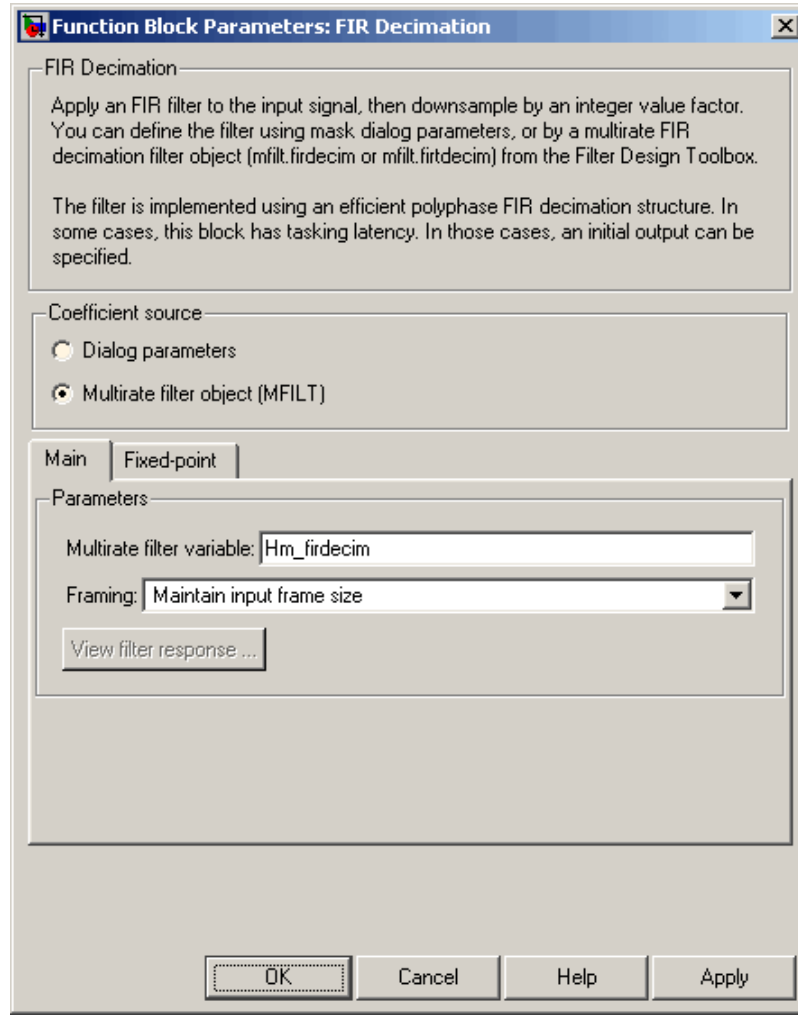
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool.

Specify Multirate Filter Object

The Main pane of the FIR Decimation block dialog appears as follows when Multirate filter object (MFILT) is specified in the Coefficient source group box.



Multirate filter variable

Specify the multirate filter object (mfilt) that you would like the block to implement. You can do this in one of three ways:

- You can fully specify the mfilt object in the block mask.
- You can enter the variable name of a mfilt object that is deÞned in any workspace.
- You can enter a variable name for a mfilt object that is not yet deÞned, as shown in the default value.

For more information on creating mfilt objects, see the mfilt function reference page in the Filter Design Toolbox documentation.

Framing

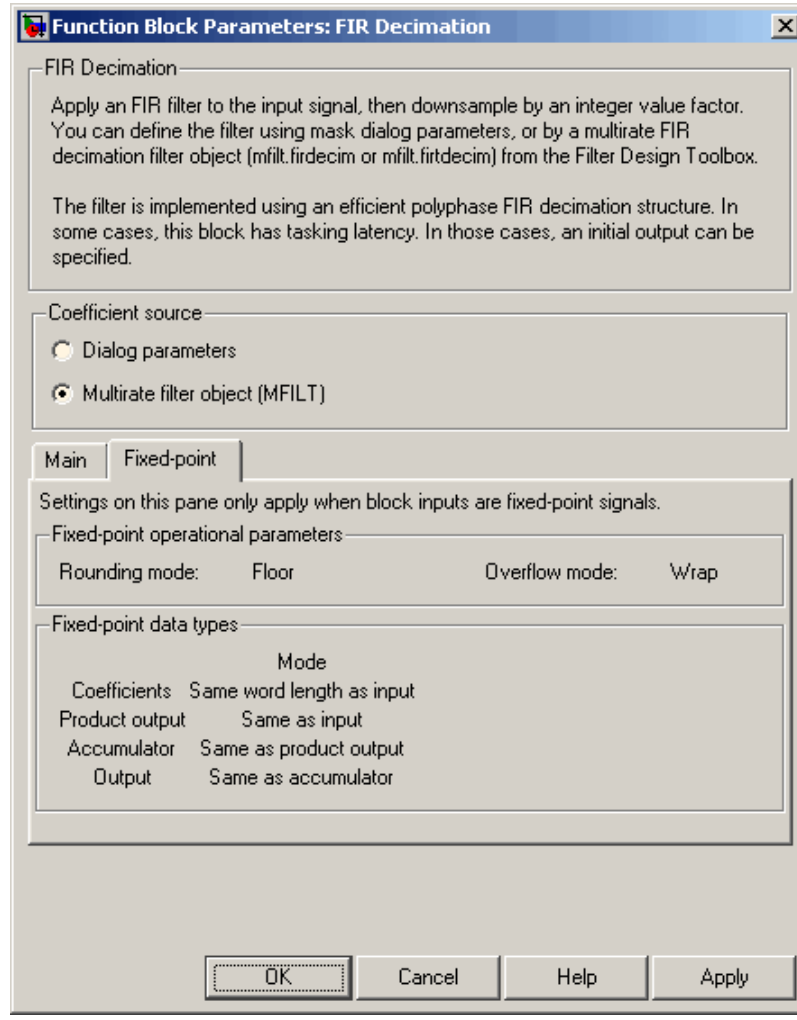
For frame-based operation, specify the method by which to implement the decimation; reduce the output frame rate, or reduce the output frame size. This parameter cannot be set to Maintain input frame rate for sample-based signals.

View filter response

This button opens the Filter Visualization Tool (fvtool) from the Signal Processing Toolbox product and displays the Þlter response of the mfilt object speciÞed in the Multirate filter variable parameter. For more information on FVTool, see the Signal Processing Toolbox documentation.

Note This button is only available when the Filter Design Toolbox product is installed. If you specify a Þlter in the Multirate filter variable parameter, you must apply the Þlter by clicking the Apply button before using the View filter response button.

The Fixed-point pane of the FIR Decimation block dialog appears as follows when Multirate filter object (MFILT) is speciÞed in the Coefficient source group box.



The fixed-point settings of the filter object specified on the Main pane are displayed on the Fixed-point pane. You cannot change these settings directly on the block mask. To change the fixed-point settings you must edit the filter object directly.

For more information on multirate filter objects, see the mfilt function reference page in the Filter Design Toolbox documentation.

References

[1] Fliege, N. J. Multirate Digital Signal Processing: Multirate Systems, Filter Banks, Wavelets . West Sussex, England: John Wiley & Sons, 1994.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Downsample	Signal Processing Blockset
FIR Interpolation	Signal Processing Blockset
FIR Rate Conversion	Signal Processing Blockset
decimate	Signal Processing Toolbox
fir1	Signal Processing Toolbox
fir2	Signal Processing Toolbox
firls	Signal Processing Toolbox

Purpose

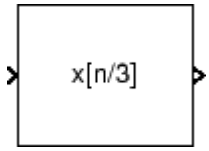
Upsample and filter input signals

Library

Filtering / Multirate Filters

dspmlti4

Description



The FIR Interpolation block resamples the discrete-time input at a rate L times faster than the input sample rate, where the integer L is specified by the Interpolation factor parameter. This process consists of two steps:

- The block upsamples the input to a higher rate by inserting L-1 zeros between samples.
- The block filters the upsampled data with a direct-form FIR filter.

The FIR Interpolation block implements the above upsampling and FIR filtering steps together using a polyphase filter structure, which is more efficient than straightforward upsample-then-filter algorithms. See N.J. Fliege, Multirate Digital Signal Processing: Multirate Systems, Filter Banks, Wavelets for more information.

The FIR filter coefficients parameter specifies the numerator coefficients of the FIR filter transfer function H(z).

$$H(z) = B(z) = b_1 + b_2 z^{-1} + \ldots + b_m z^{-(m-1)}$$

The coefficient vector, [b(1) b(2) ... b(m)], can be generated by one of the Signal Processing Toolbox filter design functions (such as fir1), and should have a length greater than the interpolation factor (m>L). The filter should be lowpass with normalized cutoff frequency no greater than 1/L. All filter states are internally initialized to zero.

The FIR Interpolation block supports real and complex floating-point and fixed-point inputs except for complex unsigned fixed-point inputs. This block supports triggered subsystems when you select Maintain input frame rate for the Framing parameter.

Sample-Based Operation

An M-by-N sample-based matrix input is treated as M*N independent channels, and the block interpolates each channel over time. The output sample period is L times shorter than the input sample period ($T_{so} = T_{si}/L$), and the input and output sizes are identical.

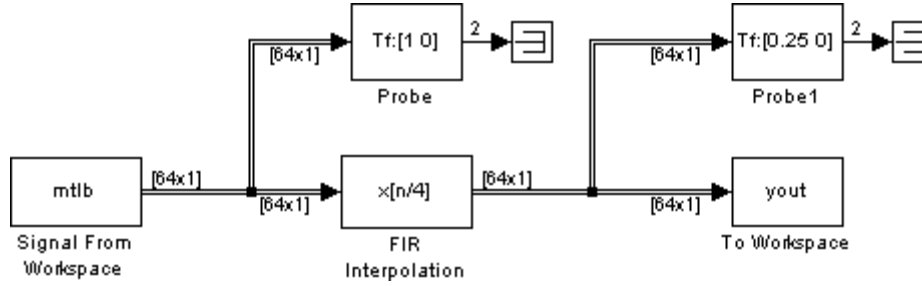
Frame-Based Operation

An M_i-by-N frame-based matrix input is treated as N independent channels, and the block interpol ates each channel over time. The Framing parameter determines how the b lock adjusts the rate at the output to accommodate the added sa mples. There are two available options:

- Maintain input frame size

  The block generates the output at the interpolated rate by using a proportionally shorter frame period at the output port than at the input port. For interpolation by a factor of L, the output frame period is L times shorter than the input frame period ($T_{fo} = T_{fi}/L$), but the input and output frame sizes are equal.

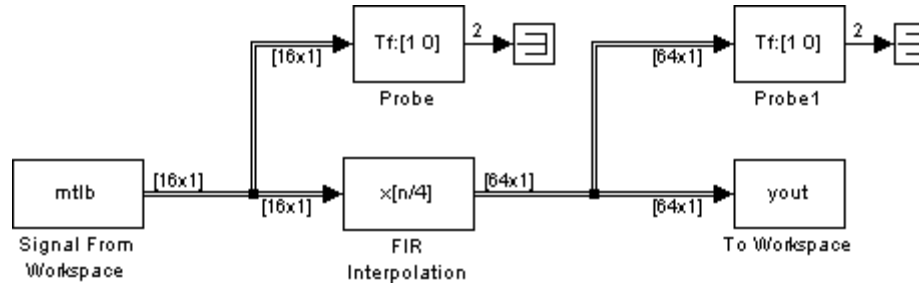
  The example below shows a single-channel input with a frame period of 1 second (Sample time = 1/64 and Samples per frame = 64 in the Signal From Workspace block) being interpolated by a factor of 4 to a frame period of 0.25 second. The input and output frame sizes are identical.



- Maintain input frame rate

The block generates the output at the interpolated rate by using a proportionally larger frame size than the input. For interpolation by a factor of L, the output frame size is L times larger than the input frame size ($M_o = M_i * L$), but the input and output frame rates are equal.

The example below shows a single-channel input of frame size 16 being interpolated by a factor of 4 to a frame size of 64. The block's input and output frame rates are identical.



Zero Latency

The FIR Interpolation block has zero-tasking latency for all single-rate operations. The block is single rate for the particular combinations of sampling mode and parameter settings shown in the table below.

Sampling Mode	Parameter Settings
Sample based	Interpolation factor parameter, L, is 1.
Frame based	Interpolation factor parameter, L, is 1, or Framing parameter is Maintain input frame rate .

Note that in sample-based mode, single-rate operation occurs only in the trivial case of factor -of-1 interpolation.

The block also has zero latency for sample-based multirate operations in the Simulink single-tasking mode. Zero-tasking latency means that

the block propagates the first filtered input (received at t=0) as the first input sample, followed by L-1 interpolated values, the second filtered input sample, and so on.

Nonzero Latency

The FIR Interpolation block is multirate for all settings other than those in the previous table. The amount of latency for multirate operation depends on the Simulink tasking mode and the block's sampling mode, as shown in the following table.

Multirate...	Sample-Based Latency	Frame-Based Latency
Single-tasking	None	None
Multitasking	L samples	L frames (M_i samples per frame)

When the block exhibits latency, the default initial condition is zero. Alternatively, you can enter a value in the Output buffer initial conditions text box. This value is divided by the Interpolation factor and output at the output port until the first filtered input sample is available.

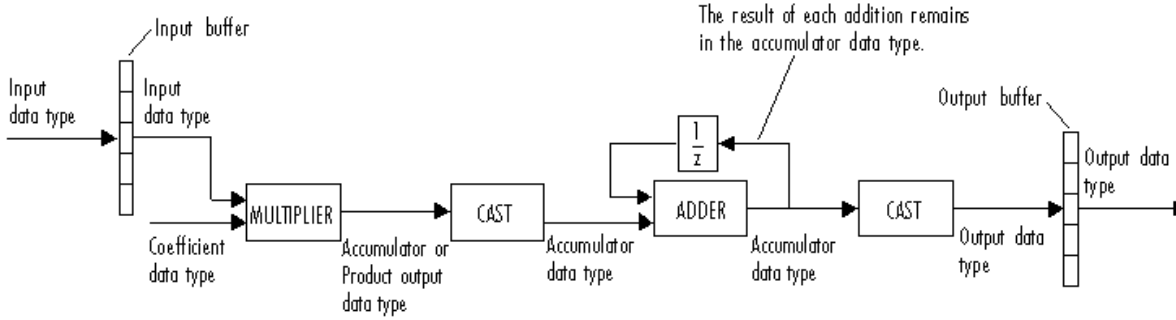
In sample-based cases, the scaled initial conditions appear at the start of each channel, followed immediately by the first filtered input sample, L-1 interpolated values, and so on.

In frame-based cases, with the default initial condition, the first M_iL output rows contain zeros, where M_i is the input frame size. The first filtered input sample (first filtered row of the input matrix) appears in the output as sample M_iL+1, followed by L-1 interpolated values, the second filtered input sample, and so on. See the following example for an illustration of this case.

Note For more information on latency and the Simulink tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Models with Multiple Sample Rates” in the Real-Time Workshop User’s Guide .

Fixed-Point Data Types

The following diagram shows the data types used within the FIR Interpolation block for fixed-point signals.



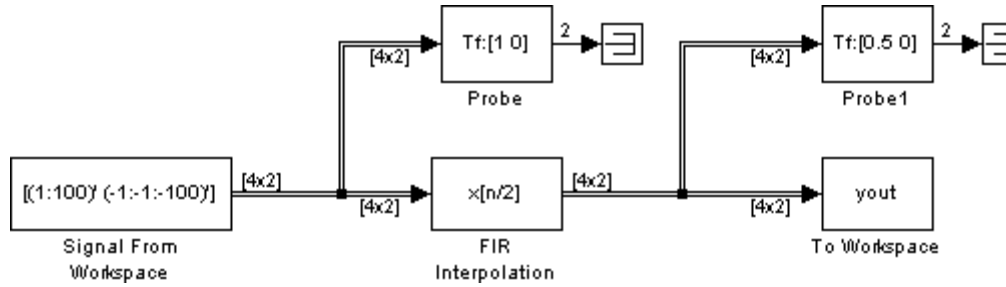
You can set the coefficient, product output, accumulator, and output data types in the block dialog as discussed in “Dialog Box” on page 2-530. The diagram shows that input data is stored in the input buffer in the same data type and scaling as the input. Filtered data is stored in the output buffer in the output data type and scaling that you set in the block dialog. Any initial conditions are also stored in the output buffer in the output data type and scaling you set in the block dialog.

The output of the multiplier is in the product output data type when at least one of the inputs to the multiplier is real. When both of the inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see “Multiplication Data Types”.

Examples

Example 1

Construct the frame-based model shown below.



Adjust the block parameters as follows:

- Conþgure the Signal From Workspace block to generate a two-channel signal with frame si ze of 4 and sample period of 0.25. This represents an output frame period of 1 (0.25*4). The þrst channel should contain the positive ramp signal 1, 2, ..., 100, and the second channel should contain the negative ramp signal -1, -2, ..., -100.
  - Signal = [(1:100)' (-1:-1:-100)']
  - Sample time = 0.25
  - Samples per frame = 4
- Conþgure the FIR Interpolation block to interpolate the two-channel input by increasing the output frame rate by a factor of 2 relative to the input frame rate. Use a third-order þlter (m=3) with normalized cutoff frequency, f_{n0}, of 0.25. (Note that f_{n0} and m satisfy $f_{n0} \leq 1/L$ and $m > L$.)
  - FIR filter coef ficients = fir1(3,0.25)
  - Interpolation factor = 2
  - Framing = Maintain input frame size

  The þlter coef þcient vector generated by fir1(3,0.25) is

  [0.0386 0.4614 0.4614 0.0386]

  or, equivalently,

$$H(z) = B(z) = 0.0386 + 0.04614z^{-1} + 0.04614z^{-2} + 0.0386z^{-3}$$

- Configure the Probe blocks by clearing the Probe width, Probe complex signal, and Probe signal dimensions check boxes (if desired).

This model is multirate because there are at least two distinct sample rates, as shown by the two Probe blocks. To run this model in the Simulink multitasking mode, open the Configuration Parameters dialog box. In the Select pane, click Solver. From the Type list, select Fixed-step, and from the Solver list, select Discrete (no continuous states). From the Tasking mode for periodic sample times list, select MultiTasking. Also set the Stop time to 30.

Run the model and look at the output, yout. The first few samples of each channel are shown below.

```
dsp_examples_yout =
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
    0.0386   -0.0386
    0.4614   -0.4614
    0.5386   -0.5386
    0.9614   -0.9614
    1.0386   -1.0386
```

Since we ran this frame-based multirate model in multitasking mode, the first eight (M_iL) output rows are zero. The first filtered input matrix row appears in the output as sample 9 (that is, sample M_iL+1). Every other row is an interpolated value.

Example 2

The doc_polyphaseinterp model illustrates the underlying polyphase implementations of the FIR Interpolation block. Run the model and view the results on the scope. The output of the FIR Interpolation block is the same as the output of the Polyphase Interpolation Filter block.

Example 3

The doc_mrf_nlp model illustrates the use of the FIR Interpolation block in a number of multistage multirate filters.

Dialog Box

The FIR Interpolation block can operate in two different modes. Select the mode in the Coefficient source group box. If you select

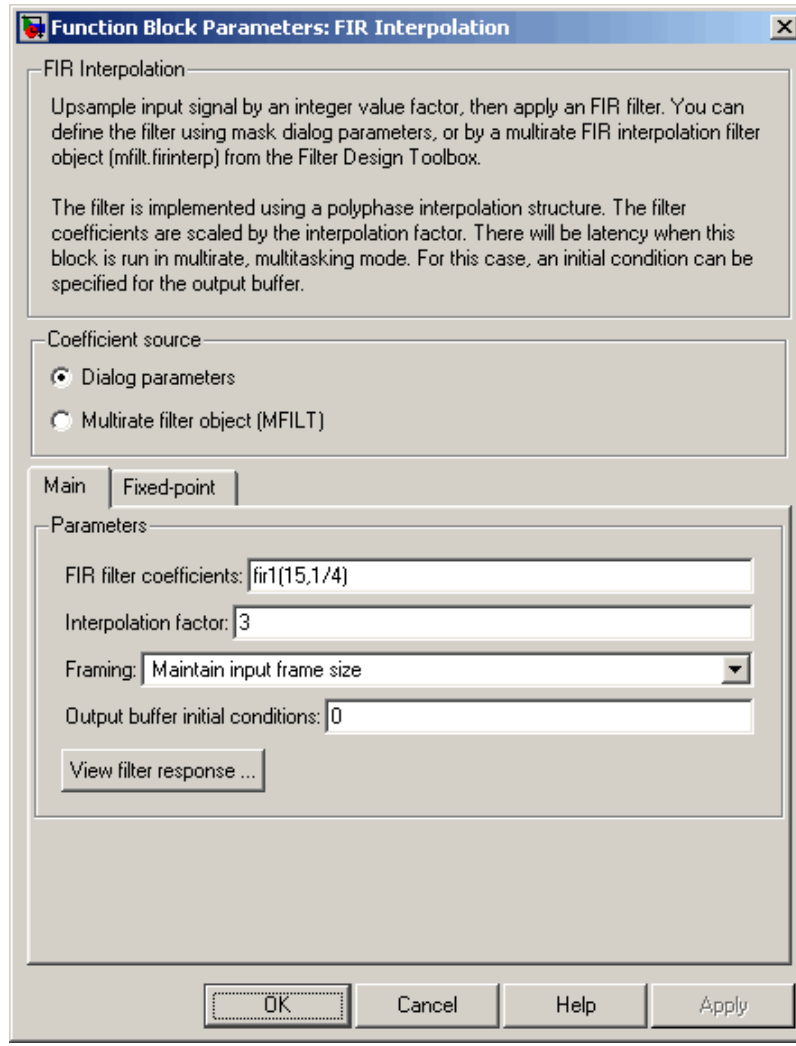
- Dialog parameters, you enter information about the filter such as structure and coefficients in the block mask.
- Multirate filter object (MFILT), you specify the filter using a Filter Design Toolbox mfilt object.

Different items appear on the FIR Interpolation block dialog depending on whether you select Dialog parameters or Multirate filter object (MFILT) in the Coefficient source group box. See the following sections for details:

- "Specify Filter Characteristics in Dialog" on page 2-531
- "Specify Multirate Filter Object" on page 2-538

Specify Filter Characteristics in Dialog

The Main pane of the FIR Interpolation block dialog appears as follows when Dialog parameters is selected in the Coefficient source group box.



FIR filter coefficients

Specify the FIR filter coefficients, in descending powers of z.

Interpolation factor

Specify the integer factor, L, by which to increase the sample rate of the input sequence.

Framing

For frame-based operation, specify the method by which to implement the interpolation: increase the output frame rate, or increase the output frame size. This parameter cannot be set to Maintain input frame rate for sample-based signals.

Output buffer initial conditions

When the block exhibits latency, enter a value in the Output buffer initial conditions text box to specify the value to output at the output port until the first filtered input sample is available. The default initial condition value is 0.

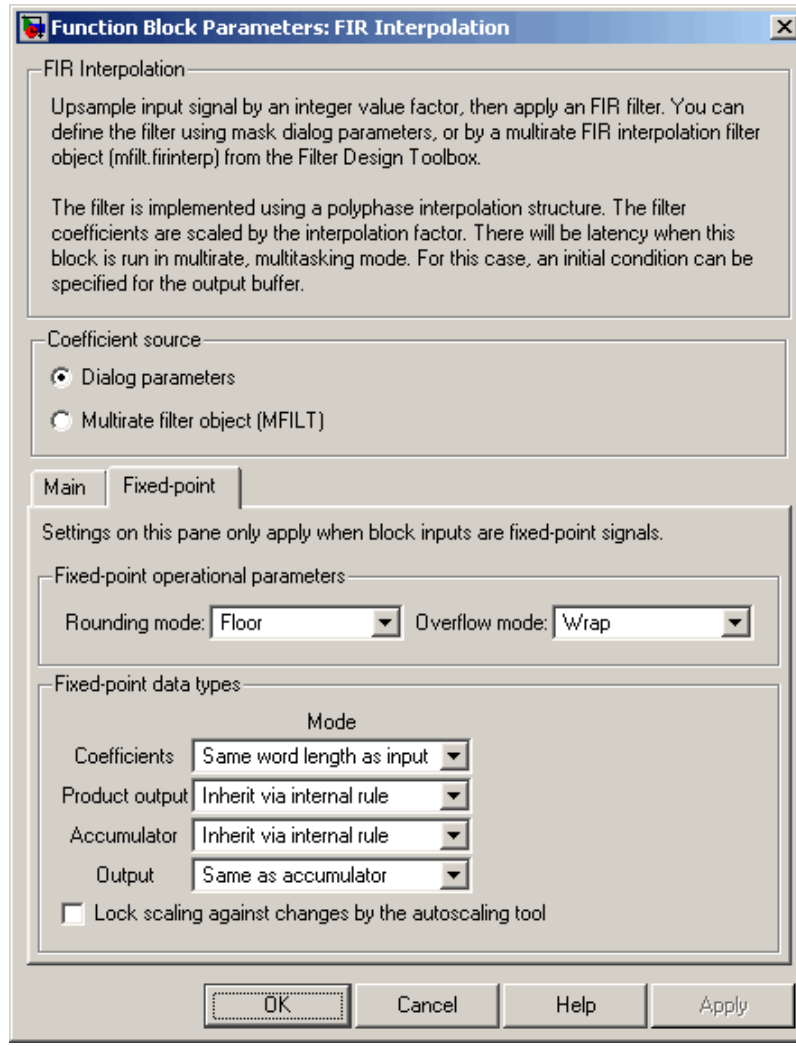
Output buffer initial conditions are stored in the output data type and scaling.

View filter response

This button opens the Filter Visualization Tool (fvtool) from the Signal Processing Toolbox product and displays the filter response of the filter defined in the block. For more information on FVTool, see the Signal Processing Toolbox documentation.

Note This button is only available when the Filter Design Toolbox product is installed. If you specify a filter in the Multirate filter variable parameter, you must apply the filter by clicking the Apply button before using the View filter response button.

The Fixed point pane of the FIR Interpolation block dialog appears as follows when Dialog parameters is specified in the Coefficient source group box.



Rounding mode

Select the rounding mode for fixed-point operations. The filter coefficients do not obey this parameter; they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations. The filter coefficients do not obey this parameter; they are always saturated.

Coefficients

Choose how you specify the word length and fraction length of the filter coefficients:

- When you select Same word length as input, the word length of the filter coefficients match that of the input to the block. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select Specify word length, you can enter the word length of the coefficients, in bits. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select Binary point scaling, you can enter the word length and the fraction length of the coefficients, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the coefficients. This block requires power-of-two slope and a bias of zero.

The filter coefficients do not obey the Rounding mode and the Overflow mode parameters; they are always saturated and rounded to Nearest.

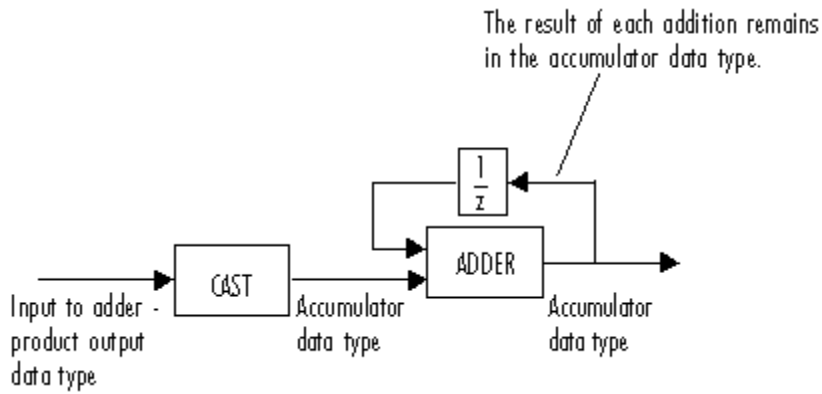
Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Fixed-Point

Data Types” on page 2-527 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Inherit via internal rule, the product output word length and fraction length are calculated automatically. For information on about how the product output word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator



As depicted above, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to

it. Use this parameter to specify how you would like to designate this accumulator word and fraction lengths.

You also use this parameter to specify the accumulator word and fraction lengths resu lting from a complex-complex multiplication in the block. See “Multiplication Data Types” for more information:

- When you select Inherit via internal rule , the accumulator word length and fraction length are calculated automatically. For informati on about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as accumulator , these characteristics match those of the accumulator.

  A special case occurs when Inherit via internal rule is speciÞed for Accumulator , and block inputs and coef Þcients are complex. In that case, the output word length be one less than the accumulator word length.
- When you select Same as product output , these characteristics match those of the product output.

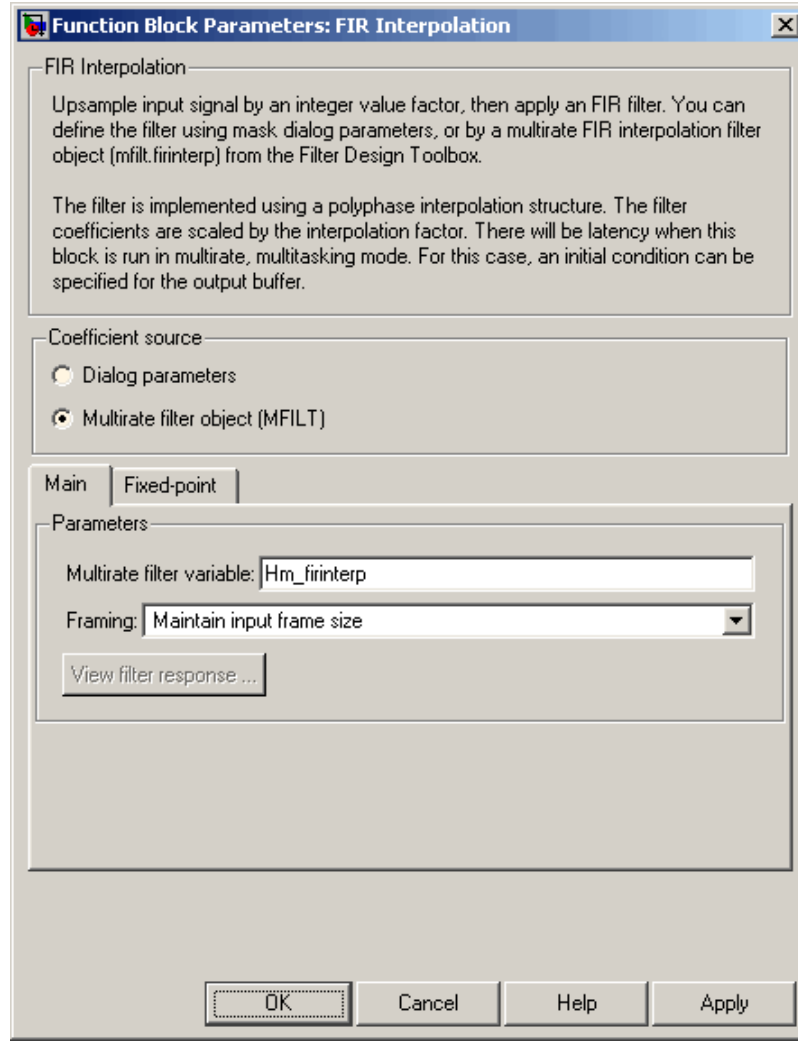
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool.

Specify Multirate Filter Object

The Main pane of the FIR Interpolation block dialog appears as follows when Multirate filter object (MFILT) is specified in the Coefficient source group box.



Multirate filter variable

Specify the multirate filter object (mfilt) that you would like the block to implement. You can do this in one of three ways:

- You can fully specify the mfilt object in the block mask.
- You can enter the variable name of a mfilt object that is deÞned in any workspace.
- You can enter a variable name for a mfilt object that is not yet deÞned, as shown in the default value.

For more information on creating mfilt objects, see the mfilt function reference page in the Filter Design Toolbox documentation.

Framing

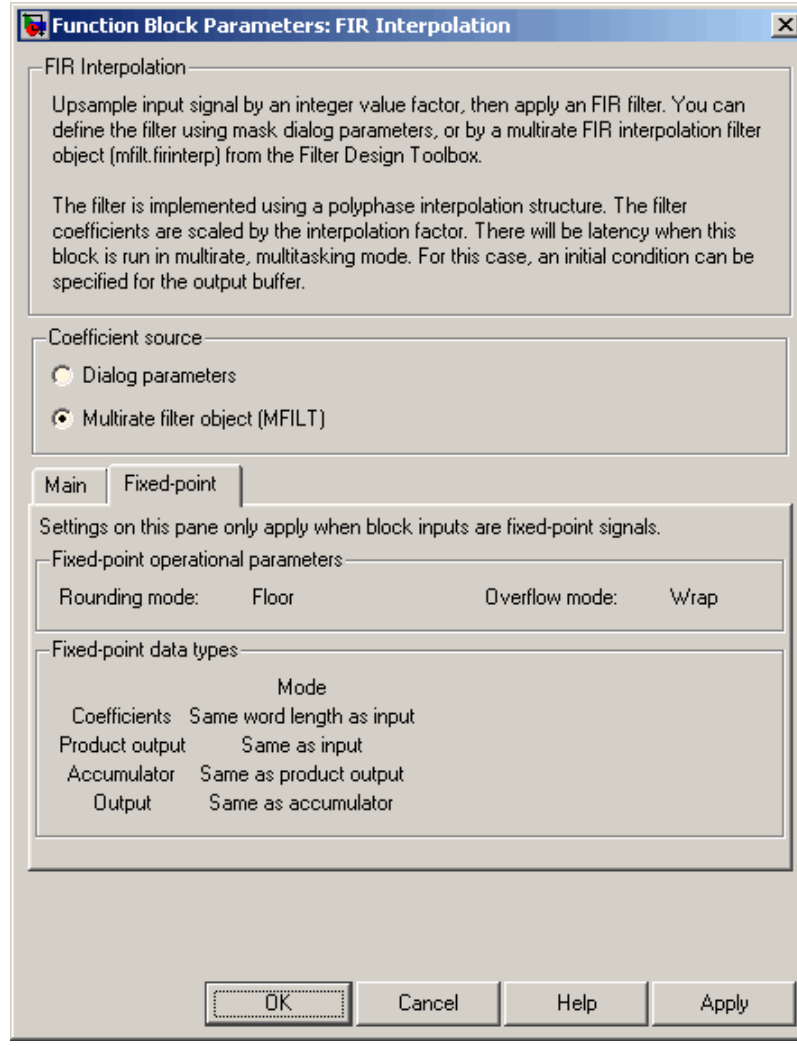
For frame-based operation, specify the method by which to implement the interpolation: increase the output frame rate, or increase the output frame size. This parameter cannot be set to Maintain input frame rate for sample-based signals.

View filter response

This button opens the Filter Visualization Tool (fvtool) from the Signal Processing Toolbox product and displays the Þlter response of the mfilt object speciÞed in the Multirate filter variable parameter. For more information on FVTool, see the Signal Processing Toolbox documentation.

Note This button is only available when the Filter Design Toolbox product is installed. If you specify a Þlter in the Multirate filter variable parameter, you must apply the Þlter by clicking the Apply button before using the View filter response button.

The Fixed-point pane of the FIR Interpolation block dialog appears as follows when Multirate filter object (MFILT) is speciÞed in the Coefficient source group box.



The fixed-point settings of the filter object specified on the Main pane are displayed on the Fixed-point pane. You cannot change these settings directly on the block mask. To change the fixed-point settings you must edit the filter object directly.

For more information on multirate filter objects, see the mfilt function reference page in the Filter Design Toolbox documentation.

References

Fliege, N. J. Multirate Digital Signal Processing: Multirate Systems, Filter Banks, Wavelets . West Sussex, England: John Wiley & Sons, 1994.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

FIR Decimation	Signal Processing Blockset
FIR Rate Conversion	Signal Processing Blockset
Upsample	Signal Processing Blockset
fir1	Signal Processing Toolbox
fir2	Signal Processing Toolbox
firls	Signal Processing Toolbox
interp	Signal Processing Toolbox

Purpose

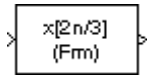
Upsample, filter, and downsample input signals

Library

Filtering / Multirate Filters

dspmlti4

Description



The FIR Rate Conversion block resamples the discrete-time input to a period K/L times the input sample period, where the integer K is specified by the Decimation factor parameter and the integer L is specified by the Interpolation factor parameter. The resampling process consists of the following steps:

1. The block upsamples the input to a higher rate by inserting L-1 zeros between input samples.
2. The upsampled data is passed through a direct-form II transpose FIR filter.
3. The block downsamples the filtered data to a lower rate by discarding K-1 consecutive samples following each sample retained.

K and L must be relatively prime integers; that is, the ratio K/L cannot be reducible to a ratio of smaller integers. The FIR Rate Conversion block implements the above three steps together using a polyphase filter structure, which is more efficient than straightforward upsample-filter-decimate algorithms. See Orfanidis [1] for more information.

The FIR filter coefficients parameter specifies the numerator coefficients of the FIR filter transfer function H(z).

$$H(z) = B(z) = b_1 + b_2 z^{-1} + \ldots + b_m z^{-(m-1)}$$

The coefficient vector, [b(1) b(2) ... b(m)], can be generated by one of the Signal Processing Toolbox filter design functions (such as fir1), and should have a length greater than the interpolation factor (m>L). The filter should be lowpass with normalized cutoff frequency

no greater than min(1/ L,1/K). All Þlter states are internally initialized to zero.

The FIR Rate Conversion block supports real and complex ßoating-point and Þxed-point inputs except for complex unsigned Þxed-point inputs.

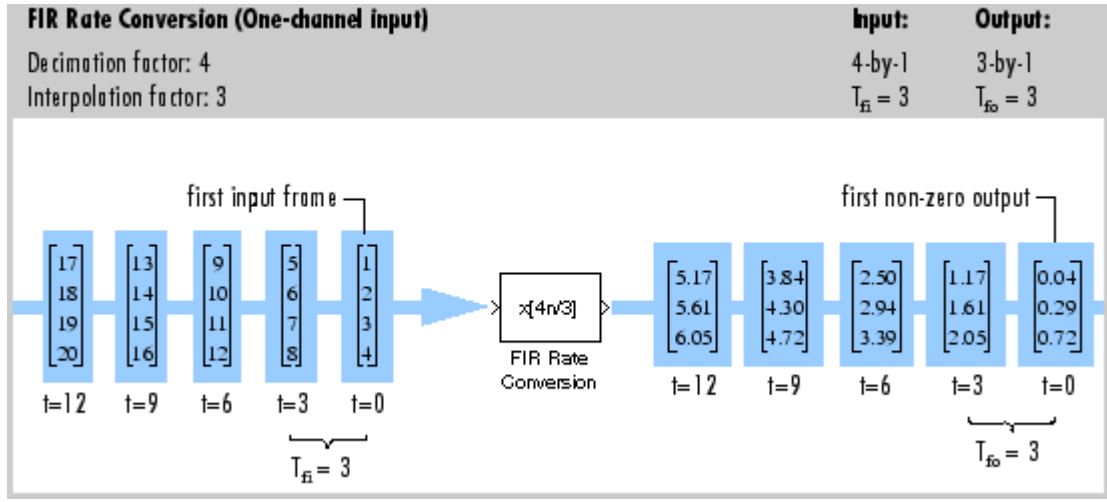
Frame-Based Operation

This block accepts only frame-based inputs. An M_i-by-N frame-based matrix input is treated as N independent channels, and the block resamples each channel independently over time.

The Interpolation factor , L, and Decimation factor , K, must satisfy the relation

$$\frac{K}{L} = \frac{M_i}{M_o}$$

for an integer output frame size M_o. The simplest way to satisfy this requirement is to let the Decimation factor equal the input frame size, M_i. The output frame size, M_o, is then equal to the Interpolation factor . This change in the frame size, from M_i to M_o, produces the desired rate conversion while leaving the output frame period the same as the input ($T_{fo} = T_{fi}$).

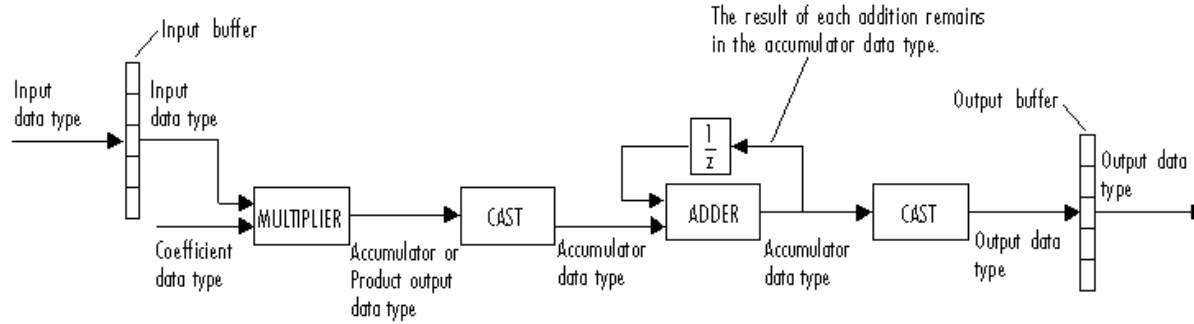


Latency

The FIR Rate Conversion block has no tasking latency. The block propagates the first filtered input (received at t=0) as the first output sample.

Fixed-Point Data Types

The following diagram shows the data types used within the FIR Rate Conversion block for fixed-point signals.



You can set the coefficient, product output, accumulator, and output data types in the block dialog as discussed in “Dialog Box” on page 2-547. The diagram shows that input data is stored in the input buffer in the same data type and scaling as the input. Filtered data is stored in the output buffer in the output data type and scaling that you set in the block dialog. Any initial conditions are also stored in the output buffer in the output data type and scaling you set in the block dialog.

The output of the multiplier is in the product output data type when at least one of the inputs to the multiplier is real. When both of the inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see “Multiplication Data Types”.

Examples

The doc_audio_src model provides a simple illustration of one way to convert a speech signal from one sample rate to another. In this model, the data is first sampled at 22,050 Hz and then resampled at 8000 Hz. If you listen to the output, you can hear that the high frequency content has been removed from the signal, although the speech sounds basically the same.

Diagnostics

An error is generated when the relation between K and L shown above is not satisfied.

> (Input port width)/(Output port width) must equal the (Decimation factor)/(Interpolation factor).

A warning is generated when L and K are not relatively prime; that is, when the ratio L/K can be reduced to a ratio of smaller integers.

> Warning: Integer conversion factors are not relatively prime in block '*modelname*/FIR Rate Conversion (Frame)'. Converting ratio *L*/*M* to *l*/*m*.

The block scales the ratio to be relatively prime and continues the simulation.

Dialog Box

The FIR Rate Conversion block can operate in two different modes. Select the mode in the Coefficient source group box. If you select

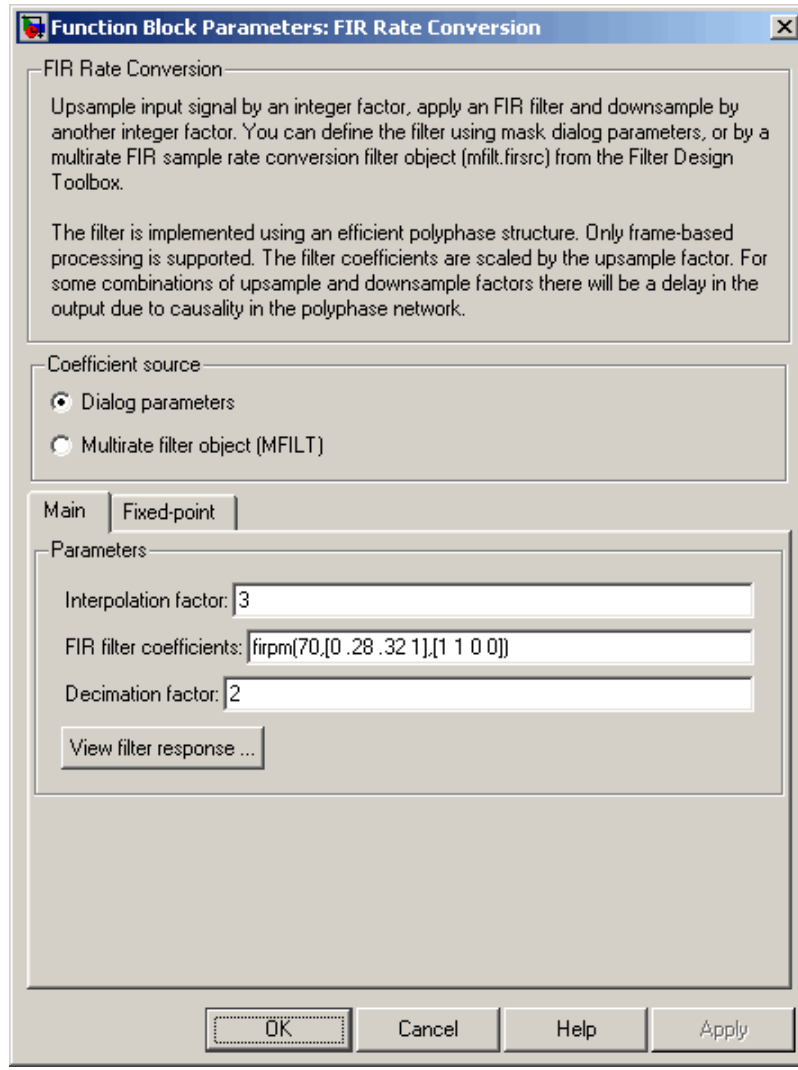
- Dialog parameters, you enter information about the filter such as structure and coefficients in the block mask.
- Multirate filter object (MFILT), you specify the filter using a Filter Design Toolbox mfilt object.

Different items appear on the FIR Rate Conversion block dialog depending on whether you select Dialog parameters or Multirate filter object (MFILT) in the Coefficient source group box. See the following sections for details:

- "Specify Filter Characteristics in Dialog" on page 2-547
- "Specify Multirate Filter Object" on page 2-554

Specify Filter Characteristics in Dialog

The Main pane of the FIR Rate Conversion block dialog appears as follows when Dialog parameters is selected in the Coefficient source group box.


Function Block Parameters: FIR Rate Conversion
FIR Rate Conversion
Upsample input signal by an integer factor, apply an FIR filter and downsample by another integer factor. You can define the filter using mask dialog parameters, or by a multirate FIR sample rate conversion filter object (mfilt.firsrc) from the Filter Design Toolbox.
The filter is implemented using an efficient polyphase structure. Only frame-based processing is supported. The filter coefficients are scaled by the upsample factor. For some combinations of upsample and downsample factors there will be a delay in the output due to causality in the polyphase network.
Coefficient source
Dialog parameters
Multirate filter object (MFILT)
Main
Fixed-point
Parameters
Interpolation factor: 3
FIR filter coefficients: firpm(70,[0 .28 .32 1],[1 1 0 0])
Decimation factor: 2
View filter response ...
OK
Cancel
Help
Apply

Interpolation factor

Specify the integer factor, L, by which to upsample the signal before filtering.

FIR filter coefficients

Specify the FIR filter coefficients in descending powers of z.

Decimation factor

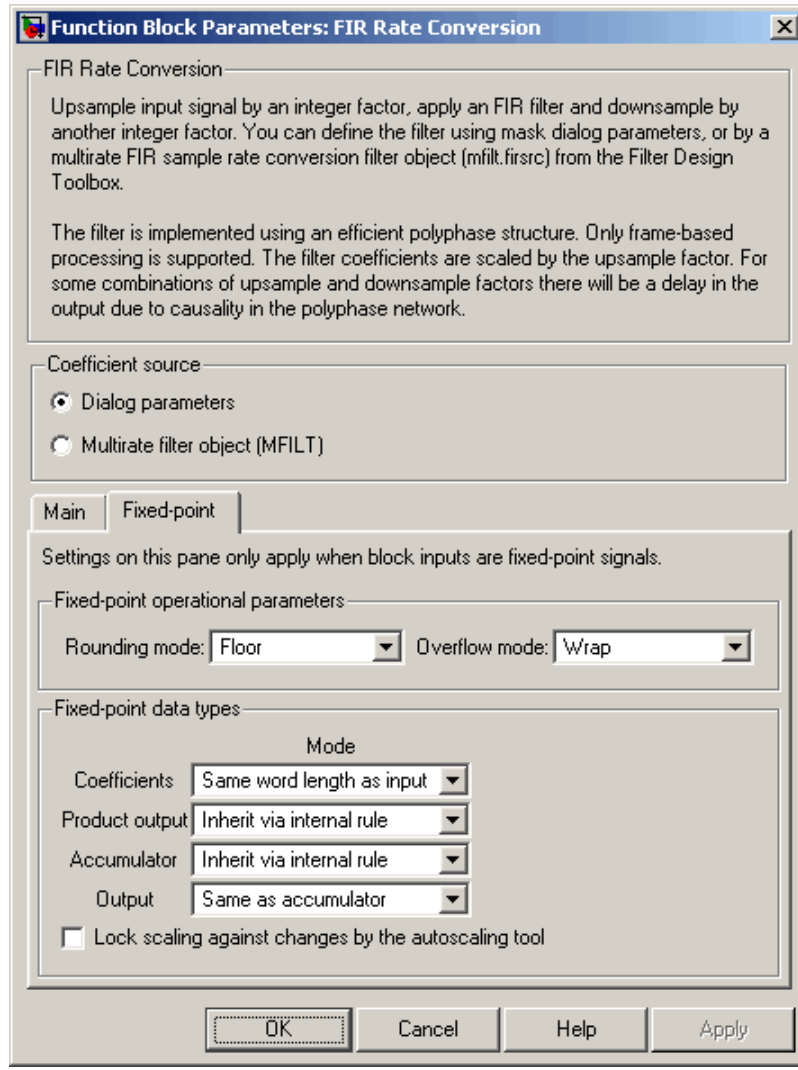
Specify the integer factor, K, by which to downsample the signal after filtering.

View filter response

This button opens the Filter Visualization Tool (fvtool) from the Signal Processing Toolbox product and displays the filter response of the filter defined in the block. For more information on FVTool, see the Signal Processing Toolbox documentation.

Note This button is only available when the Filter Design Toolbox product is installed. If you specify a filter in the Multirate filter variable parameter, you must apply the filter by clicking the Apply button before using the View filter response button.

The Fixed point pane of the FIR Rate Conversion block dialog appears as follows when Dialog parameters is specified in the Coefficient source group box.


Function Block Parameters: FIR Rate Conversion
FIR Rate Conversion
Upsample input signal by an integer factor, apply an FIR filter and downsample by another integer factor. You can define the filter using mask dialog parameters, or by a multirate FIR sample rate conversion filter object (mfilt.firsrc) from the Filter Design Toolbox.
The filter is implemented using an efficient polyphase structure. Only frame-based processing is supported. The filter coefficients are scaled by the upsample factor. For some combinations of upsample and downsample factors there will be a delay in the output due to causality in the polyphase network.
Coefficient source
Dialog parameters
Multirate filter object (MFILT)
Main
Fixed-point
Settings on this pane only apply when block inputs are fixed-point signals.
Fixed-point operational parameters
Rounding mode: Floor
Overflow mode: Wrap
Fixed-point data types
Mode
Coefficients
Same word length as input
Product output
Inherit via internal rule
Accumulator
Inherit via internal rule
Output
Same as accumulator
Lock scaling against changes by the autoscaling tool
OK
Cancel
Help
Apply

Rounding mode

Select the rounding mode for fixed-point operations. The filter coefficients do not obey this parameter; they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations. The filter coefficients do not obey this parameter; they are always saturated.

Coefficients

Choose how you specify the word length and fraction length of the filter coefficients.

- When you select Same word length as input, the word length of the filter coefficients match that of the input to the block. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select Specify word length, you can enter the word length of the coefficients, in bits. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select Binary point scaling, you can enter the word length and the fraction length of the coefficients, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the coefficients. This block requires power-of-two slope and a bias of zero.
- The coefficients do not obey the Rounding mode and the Overflow mode parameters; they are always saturated and rounded to Nearest.

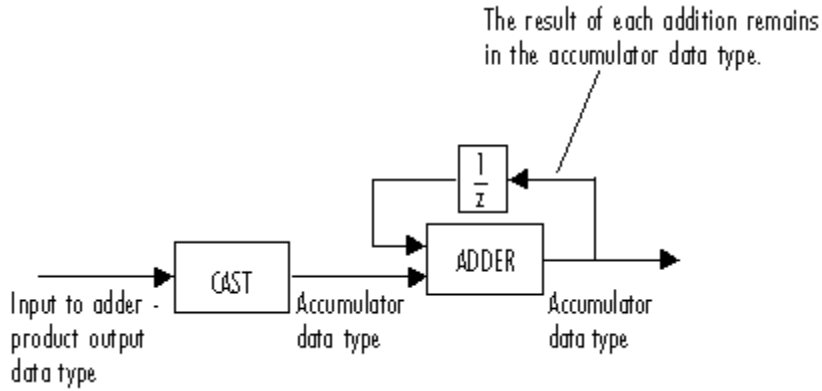
Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-508 and “Multiplication Data Types” for

illustrations depicting the use of the product output data type in this block.

- When you select Inherit via internal rule, the product output word length and fraction length are calculated automatically. For information on about how the product output word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator



As depicted above, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how you would like to designate this accumulator word and fraction lengths.

You also use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block. See “Multiplication Data Types” for more information.

- When you select Inherit via internal rule, the accumulator word length and fraction length are calculated automatically. For information about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as product output, these characteristics match those of the product output.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as accumulator, these characteristics match those of the accumulator.

  A special case occurs when Inherit via internal rule is specified for Accumulator, and block inputs and coefficients are complex. In that case, the output word length is one less than the accumulator word length.
- When you select Same as product output, these characteristics match those of the product output.
- When you select Same as input, these characteristics match those of the input to the block.

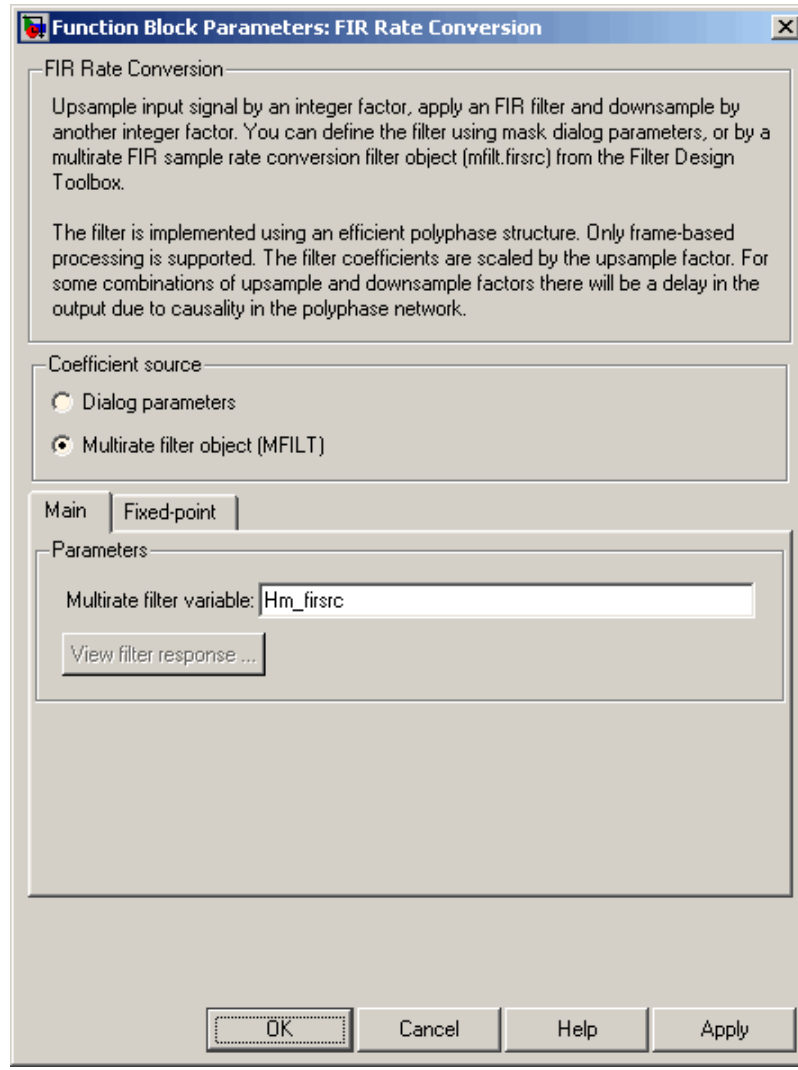
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool.

Specify Multirate Filter Object

The Main pane of the FIR Rate Conversion block dialog appears as follows when Multirate filter object (MFILT) is specified in the Coefficient source group box.


Function Block Parameters: FIR Rate Conversion
FIR Rate Conversion
Upsample input signal by an integer factor, apply an FIR filter and downsample by another integer factor. You can define the filter using mask dialog parameters, or by a multirate FIR sample rate conversion filter object (mfilt.firsrc) from the Filter Design Toolbox.
The filter is implemented using an efficient polyphase structure. Only frame-based processing is supported. The filter coefficients are scaled by the upsample factor. For some combinations of upsample and downsample factors there will be a delay in the output due to causality in the polyphase network.
Coefficient source
Dialog parameters
Multirate filter object (MFILT)
Main
Fixed-point
Parameters
Multirate filter variable:
Hm_firsrc
View filter response ...
OK
Cancel
Help
Apply

Multirate filter variable

Specify the multirate filter object (mfilt) that you would like the block to implement. You can do this in one of three ways:

- You can fully specify the mfilt object in the block mask.
- You can enter the variable name of a mfilt object that is defined in any workspace.
- You can enter a variable name for a mfilt object that is not yet defined, as shown in the default value.

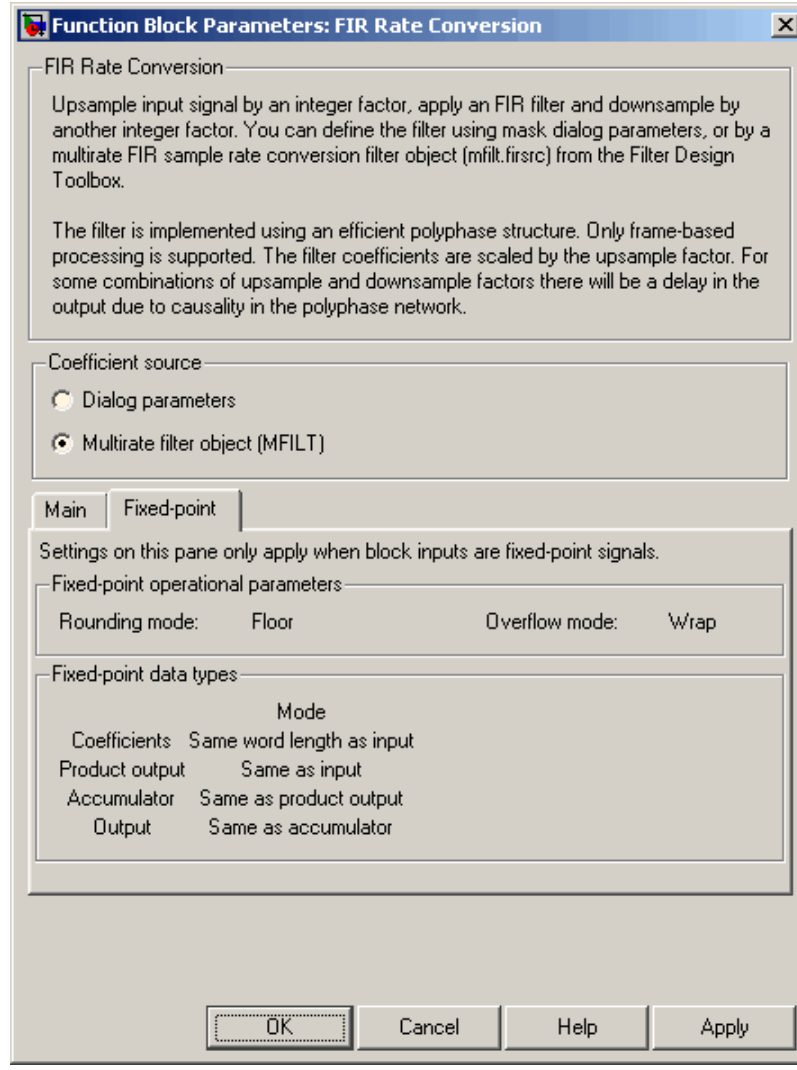
For more information on creating mfilt objects, see the mfilt function reference page in the Filter Design Toolbox documentation.

View filter response

This button opens the Filter Visualization Tool (fvtool) from the Signal Processing Toolbox product and displays the filter response of the mfilt object specified in the Multirate filter variable parameter. For more information on FVTool, see the Signal Processing Toolbox documentation.

Note This button is only available when the Filter Design Toolbox product is installed. If you specify a filter in the Multirate filter variable parameter, you must apply the filter by clicking the Apply button before using the View filter response button.

The Fixed-point pane of the FIR Rate Conversion block dialog appears as follows when Multirate filter object (MFILT) is specified in the Coefficient source group box.



The Þxed-point settings of the Þlter object speciÞed on the Main pane are displayed on the Fixed-point pane. You cannot change these

settings directly on the block mask. To change the fixed-point settings you must edit the filter object directly.

For more information on multirate filter objects, see the mfilt function reference page in the Filter Design Toolbox documentation.

References

[1] Orfanidis, S. J. Introduction to Signal Processing. Prentice Hall, 1996.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Downsample	Signal Processing Blockset
FIR Decimation	Signal Processing Blockset
FIR Interpolation	Signal Processing Blockset
Upsample	Signal Processing Blockset
fir1	Signal Processing Toolbox
fir2	Signal Processing Toolbox

firls	Signal Processing Toolbox
upfirdn	Signal Processing Toolbox

See the following sections for related information:

- “Converting Sample and Frame Rates”
- “Multirate Filters”

Flip

Purpose

Flip input vertically or horizontally

Library

Signal Management / Indexing

dspindex

Description

The Flip block vertically or horizontally reverses the M-by-N input matrix, u. The output always has the same dimension and frame status as the input.

When you select Columns from the Flip along menu, the block vertically flips the input so that the first row of the input is the last row of the output.

```
y = flipud(u)          % Equivalent MATLAB code
```

For convenience, length- M 1-D vector inputs are treated as M-by-1 column vectors for vertical flipping.

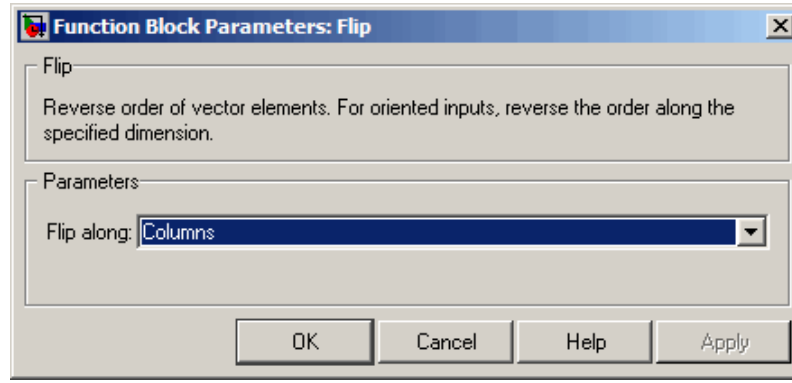
When you select Rows from the Flip along menu, the block horizontally flips the input so that the first column of the input is the last column of the output.

```
y = fliplr(u)          % Equivalent MATLAB code
```

For convenience, length- N 1-D vector inputs are treated as 1-by- N row vectors for horizontal flipping. The output always has the same dimension and frame status as the input.

This block supports Simulink virtual buses.

Dialog Box



Flip along

The dimension along which to ßip the input. Columns speciÞes vertical ßipping, while Rows speciÞes horizontal ßipping.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Selector	Simulink
Transpose	Signal Processing Blockset
Variable Selector	Signa l Processing Blockset
flipud	MATLAB
fliplr	MATLAB

Purpose

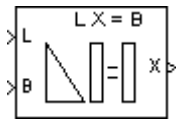
Solve LX=B for X when L is lower triangular matrix

Library

Math Functions / Matrices and Linear Algebra / Linear System Solvers

dspsolvers

Description



The Forward Substitution block solves the linear system LX=B by simple forward substitution of variables, where:

- L is the lower triangular M-by-M matrix input to the L port.
- B is the M-by-N matrix input to the B port.

The sample-based output is the M-by-N matrix X that is the solution of the equations. The block does not check the rank of the inputs.

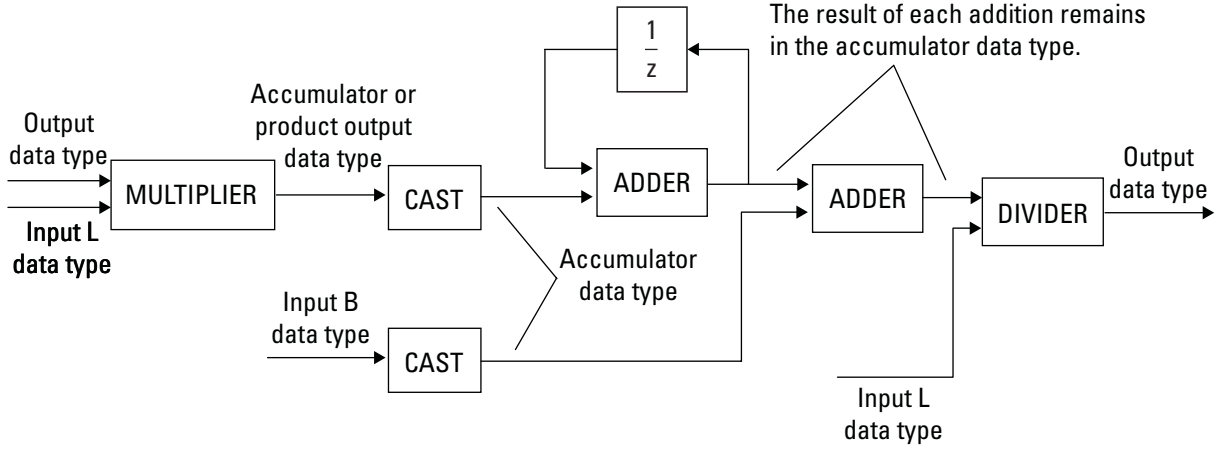
The block only uses the elements in the lower triangle of input L and ignores the upper elements. When you select Input L is unit-lower triangular, the block replaces the elements on the diagonal of L with 1s. This is useful when matrix L is the result of another operation, such as an LDL decomposition, that uses the diagonal elements to represent the D matrix.

The block treats a length-M vector input at port B as an M-by-1 matrix.

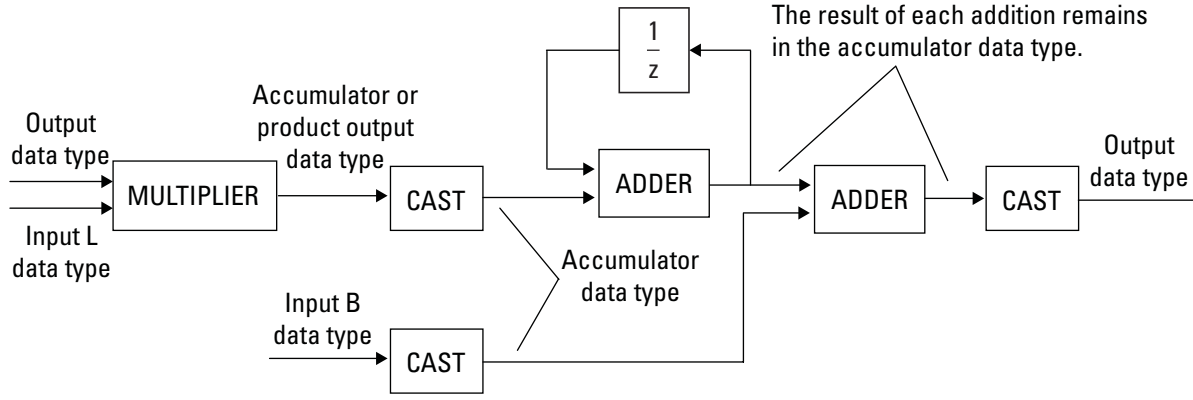
Fixed-Point Data Types

The following diagram shows the data types used within the Forward Substitution block for fixed-point signals.

When input L is not unit-lower triangular:



When input L is unit-lower triangular:

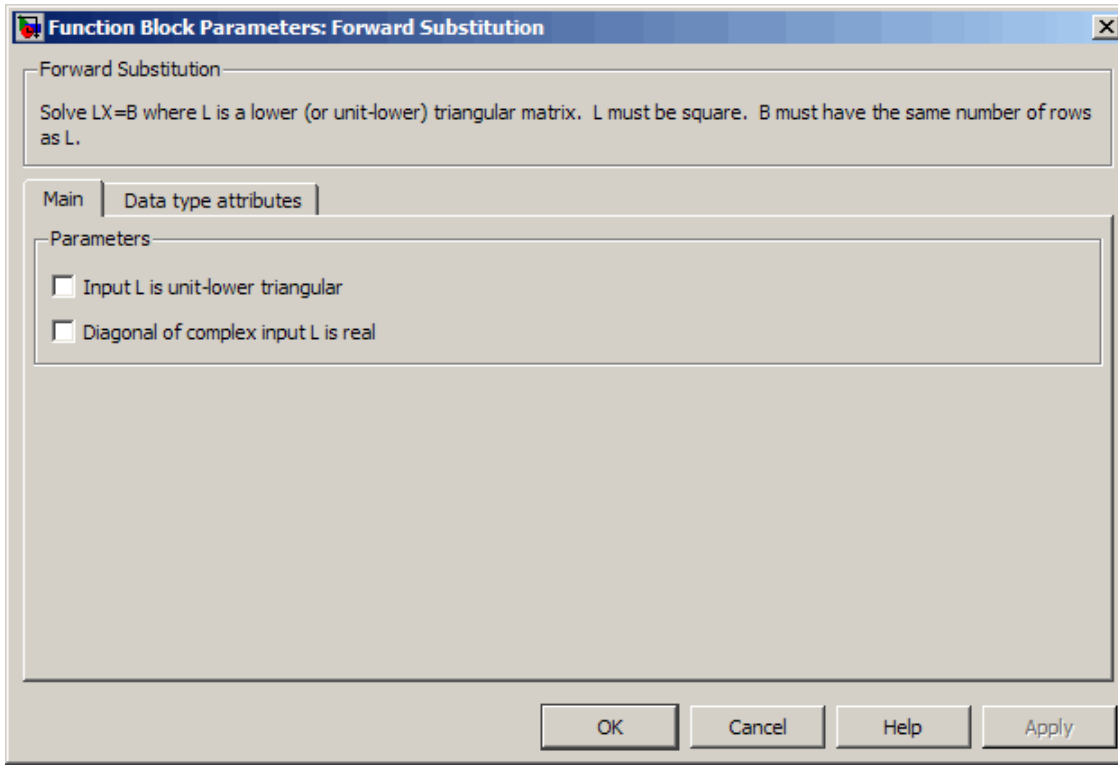


You can set the product output, accumulator, and output data types in the block dialog box, as discussed in the following section.

The output of the multiplier is in the product output data type when the input is real. When the input is complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see “Multiplication Data Types”.

Dialog Box

The Main pane of the Forward Substitution block dialog box appears as follows.



Input L is unit-lower triangular

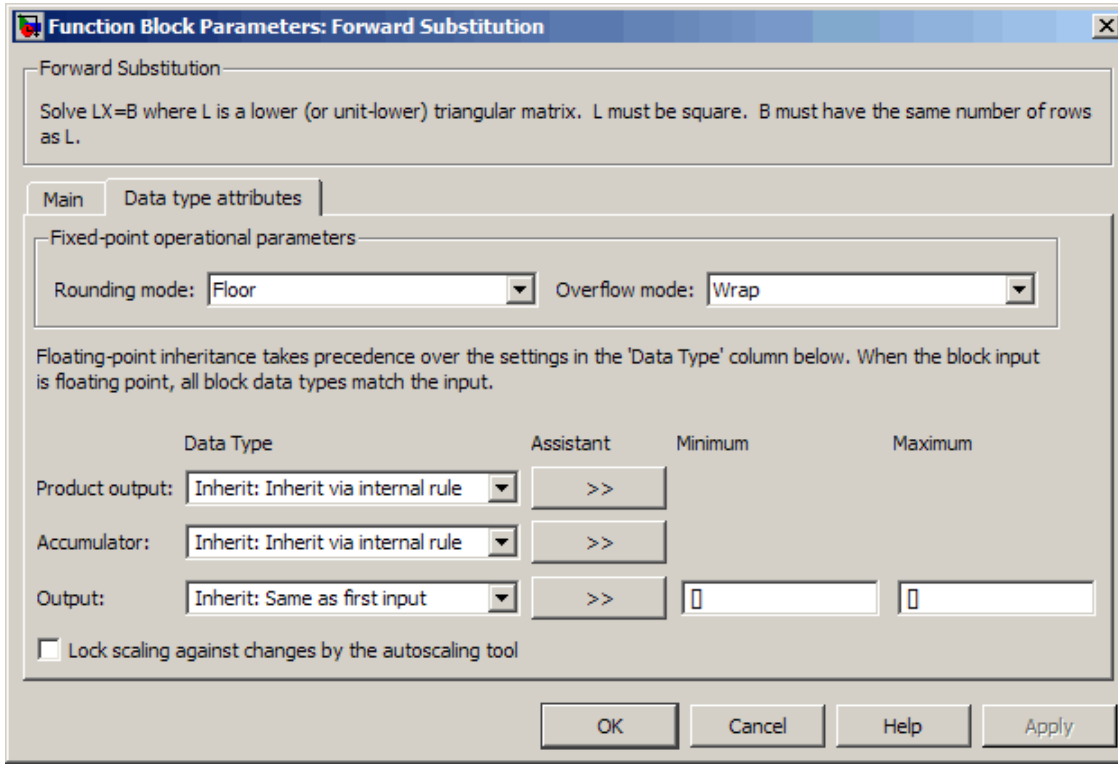
Select to replace the elements on the diagonal of L with 1s.

Diagonal of complex input L is real

Select to optimize simulation speed when the diagonal elements of complex input L are real. This parameter is only visible when Input L is unit-upper triangular is not selected.

Note When L is a complex Þxed-point signal, you must select either Input L is unit-lower triangular or Diagonal of complex input L is real . In such a case, the block ignores any imaginary part of the diagonal of L.

The Data type attributes pane of the Forward Substitution block dialog box appears as follows.



Note Floating-point inheritance takes precedence over the data type settings de Þned on this pane. When inputs are ßoating point, the block ignores these settings, and a ll internal data types are ßoating point.

Rounding mode

Select the rounding mode for Þxed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output data type

Specify the product output data type. See “Fixed-Point Data Types” on page 2-563 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block. You can set it to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule`
- An expression that evaluates to a data type, for example, `fixdt(1,16,0)`

Click the Show data type assistant button

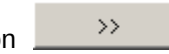
to display the Data Type Assistant, which helps you set the Product output data type parameter.

See “Using the Data Type Assistant” in Simulink User’s Guide for more information.

Accumulator data type

Specify the accumulator data type. See “Fixed-Point Data Types” on page 2-563 for illustrations depicting the use of the accumulator data type in this block. You can set it to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule`
- An expression that evaluates to a data type, for example, `fixdt(1,16,0)`

Click the Show data type assistant button 

to display the Data Type Assistant, which helps you set the Accumulator data type parameter.

See “Using the Data Type Assistant” in Simulink User’s Guide for more information.

Output data type

Specify the output data type. See “Fixed-Point Data Types” on page 2-563 for illustrations depicting the use of the output data type in this block. You can set it to:

- A rule that inherits a data type, for example, `Inherit: Same as first input`
- An expression that evaluates to a data type, for example, `fixdt(1,16,0)`

Click the Show data type assistant button

to display the Data Type Assistant, which helps you set the Output data type parameter.

See “Specifying Block Output Data Types” in Simulink User’s Guide for more information.

Minimum

Specify the minimum value that the block should output. The default value, `[]`, is equivalent to `-Inf`. Simulink software uses this value to perform:

- Simulation range checking (see “Checking Signal Ranges”)
- Automatic scaling of fixed-point data types

Maximum

Specify the maximum value that the block should output. The default value, `[]`, is equivalent to `Inf`. Simulink software uses this value to perform:

- Simulation range checking (see “Checking Signal Ranges”)
- Automatic scaling of fixed-point data types

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool.

Supported Data Types

Port	Supported Data Types
L	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
B	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
X	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

See Also

Backward Substitution	Signal Processing Blockset
Cholesky Solver	Signal Processing Blockset
LDL Solver	Signal Processing Blockset
Levinson-D urbin	Signal Processing Blockset

LU Solver	Signal Processing Blockset
QR Solver	Signal Processing Blockset

See “Linear System Solvers” for related information.

Purpose

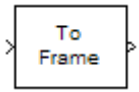
Specify sampling mode of output signal

Library

Signal Management / Signal Attributes

dspsigattribs

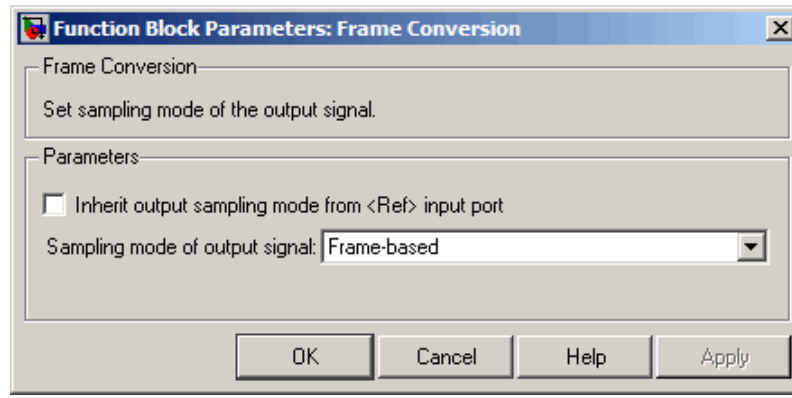
Description



The Frame Conversion block passes the input through to the output and sets the output sampling mode to the value of the Sampling mode of output signal parameter, which can be either Frame-based or Sample-based . The output sampling mode can also be inherited from the signal at the Ref (reference) input port, which you make visible by selecting the Inherit output sampling mode from <Ref> input port check box.

The Frame Conversion block does not make any changes to the input signal other than the sampling mode. In particular, the block does not rebuffer or resize 2-D inputs. Be cause 1-D vectors cannot be frame based, when the input is a length- M 1-D vector and the block is in Frame-based mode, the output is a frame-based M-by-1 matrix — that is, a single channel.

Dialog Box



Inherit output sampling mode from <Ref> input port

Select to enable the Ref port from which the block inherits the output sampling mode.

Sampling mode of output signal

Specify the sampling mode of the output signal, Frame-based or Sample-based .

Supported Data Types

Port	Supported Data Types
In	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Ref	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Buffer	Signal Processing Blockset
Check Signal Attributes	Signal Processing Blockset
Convert 1-D to 2-D	Signal Processing Blockset
Convert 2-D to 1-D	Signal Processing Blockset
Inherit Complexity	Signal Processing Blockset
Unbuffer	Signal Processing Blockset
Probe	Simulink
Reshape	Simulink
Signal Specification	Simulink

Frame Status Conversion (Obsolete)

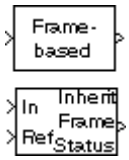
Purpose

Specify frame status of output as sample based or frame based

Library

dspobslib

Description



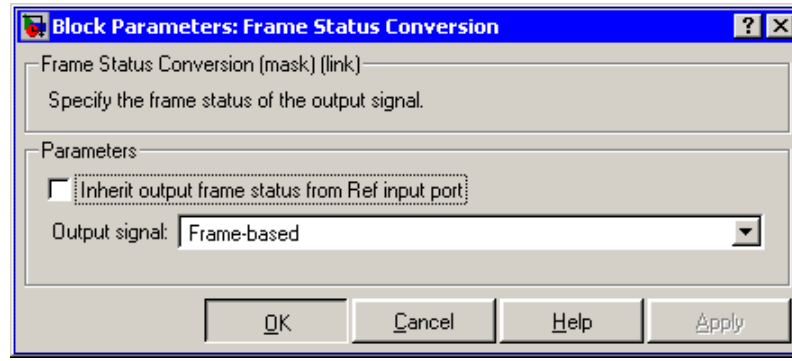
Note The Frame Status Conversion block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the Frame Conversion block.

The Frame Status Conversion block passes the input through to the output, and sets the output frame status to the Output signal parameter, which can be either Frame-based or Sample-based. The output frame status can also be inherited from the signal at the Ref (reference) input port, which is made visible by selecting the Inherit output frame status from Ref input port check box.

When the Output signal parameter setting or the inherited signal's frame status differs from the input frame status, the block changes the input frame status accordingly, but does not otherwise alter the signal. In particular, the block does not rebuffer or resize 2-D inputs. Because 1-D vectors cannot be frame based, when the input is a length-M 1-D vector, and the Output signal parameter is set to Frame-based, the output is a frame-based M-by-1 matrix (that is, a single channel).

When the Output signal parameter or the inherited signal's frame status matches the input frame status, the block passes the input through to the output unaltered.

Dialog Box



Inherit output frame status from Ref input port

When selected, enables the Ref input port from which the block inherits the output frame status.

Output signal

The output frame status, Frame-based or Sample-based .

Supported Data Types

Port	Supported Data Types
In	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Port	Supported Data Types
Ref	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Check Signal Attributes	Signal Processing Blockset
Convert 1-D to 2-D	Signal Processing Blockset
Convert 2-D to 1-D	Signal Processing Blockset
Inherit Complexity	Signal Processing Blockset

Purpose

Read audio data from computer's audio device

Library

Signal Processing Sources

dspsrcs4

Description

The From Audio Device block reads audio data from an audio device in real time. This block has the following limitations:

- Not supported for use with the Simulink Model block.
- Not currently supported on Solaris™ platforms.

Use the Device parameter to specify the dev ice from which to acquire audio. This parameter is automatically populated based on the audio devices installed on your system. If you plug or unplug an audio device from your system, type `clear mex` at the MATLAB command prompt to update this list.

Use the Number of channels parameter to specify the number of audio channels in the signal. For example:

- Enter `2` if the audio source is two channels (stereo).
- Enter `1` if the audio source is single channel (mono).
- Enter `6` if you are working with a 5.1 speaker system.

The block's output is an M-by-N matrix, where M is the number of consecutive samples and N is the number of audio channels.

Use the Sample rate (Hz) parameter to specify the number of samples per second in the signal. If the audio data is processed in uncompressed pulse code modulation (PCM) format, it should typically be sampled at one of the standard audio device rates: 8000, 11025, 22050, 44100, or 48000 Hz.

Use the Device data type parameter to specify the data type of the audio data that the device is placing in the buffer. You can choose:

- 8-bit integer
- 16-bit integer
- 24-bit integer
- 32-bit float
- Determine from output data type

If you choose Determine from output data type , the following table summarizes the block's behavior.

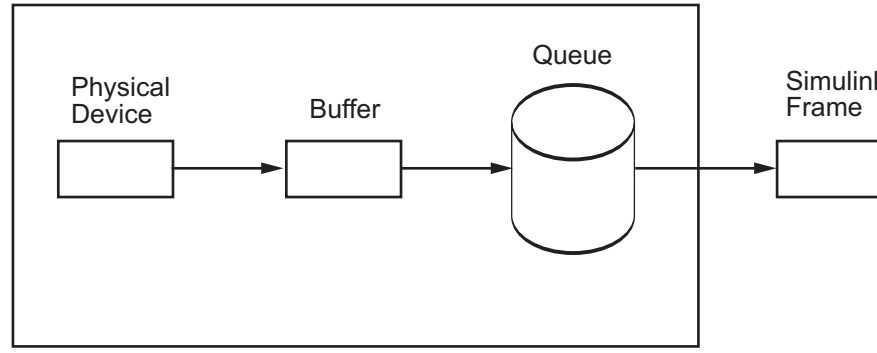
Output Data Type	Device Data Type
Double-precision floating point or single-precision floating point	32-bit floating point
32-bit integer	24-bit integer
16-bit integer	16-bit integer
8-bit integer	8-bit integer

If you choose Determine from output data type and the device does not support a data type, the block uses the next lowest precision data type supported by the device.

Use the Frame size (samples) parameter to specify the number of samples in the block's output. Use the Output data type parameter to specify the data type of audio data output by the block.

Buffering

The From Audio Device block buffers the data from the audio device using the process illustrated by the following figure.



From Audio Device Block

1 At the start of the simulation, th e audio device begins writing the input data to a buffer. This data has the data type speci Þed by the Device data type parameter.

2 When the buffer is full, the From Audio Device block writes the contents of the buffer to the queu e. Specify the size of this queue using the Queue duration (seconds) parameter.

3 As the audio device appends audio data to the bottom of the queue, the From Audio Device block pulls data from the top of the queue to Þll the Simulink frame. This data has the data type speci Þed by the Output data type parameter.

Select the Automatically determine buffer size check box to allow the block to calculate a conservative buffer size using the following equation:

$$size = 2^{\left\lfloor \log_2 \frac{sr}{10} \right\rfloor}$$

In this equation, size is the buffer size, and sr is the sample rate. If you clear this check box, the Buffer size (samples) parameter appears on the block. Use this parameter to specify the buffer size in samples.

When the simulation throughput rate is lower than the hardware throughput rate, the queue, which is initially empty, fills up. If the queue is full, the block drops the incoming data from the audio device. When the simulation throughput rate is higher than the hardware throughput rate, the From Audio Device block waits for new samples to become available.

Troubleshooting

When Simulink cannot keep up with an audio device that is operating in real time, the queue fills up and the block begins to lose audio data. Here are several ways to deal with this situation:

- Increase the queue duration.

  The Queue duration (seconds) parameter specifies the duration of the signal, in seconds, that can be buffered during the simulation. This is the maximum length of time that the block's data demand can lag behind the hardware's data supply.

- Increase the buffer size.

  The size of the buffer processed in each interrupt from the audio device affects the performance of your model. If the buffer is too small, a large portion of hardware resources are used to write data to the queue. If the buffer is too big, Simulink must wait for the device to fill the buffer before it moves the data to the queue, which introduces latency.

- Increase the simulation throughput rate.

  Two useful methods for improving simulation throughput rates are increasing the signal frame size and compiling the simulation into native code:

  - Increase frame sizes and convert sample-based signals to frame-based signals throughout the model to reduce the amount of block-to-block communication overhead. This can increase throughput rates in many cases. However, larger frame sizes generally result in greater model latency due to initial buffering operations.

- Generate executable code with Real-Time Workshop code generation software. Native code runs much faster than Simulink and should provide rates adequate for real-time audio processing.

Other ways to improve throughput rates include simplifying the model and running the simulation on a faster PC processor. For other ideas on improving simulation performance, see “Delay and Latency” and “Improving Simulation Performance and Accuracy” in the Simulink documentation.

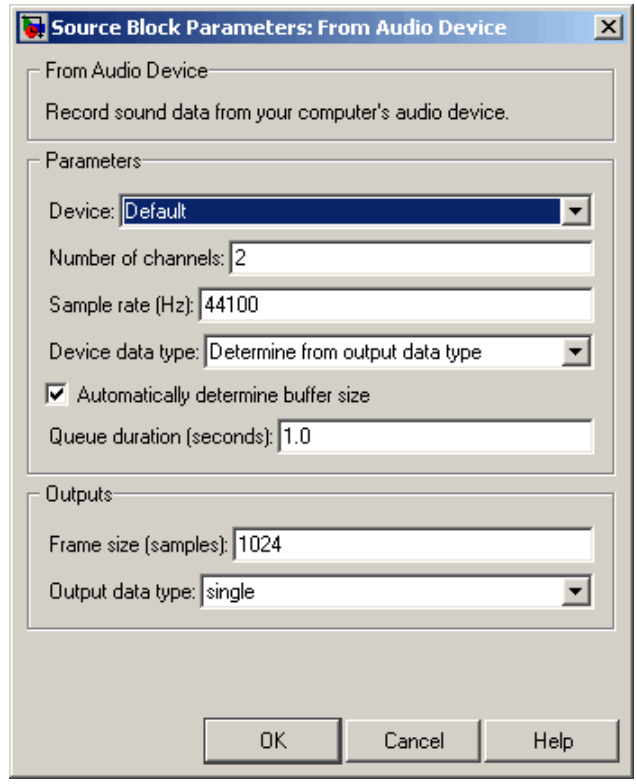
Audio Hardware API

The To Audio Device and From Audio Device blocks use the open-source PortAudio library in order to communicate with the audio hardware on a given computer. The PortAudio library supports a range of API’s designed to communicate with the audio hardware on a given platform. The following API choices were made when building the PortAudio library for the Signal Processing Blockset product:

- Windows: DirectSound
- Linux: OSS
- Mac: CoreAudio

If you are interested in using a different audio API, such as ASIO (Windows®) or ALSA (Linux®) please search for PortAudio on the Matlab Central website.

Dialog Box



Device
: Specify the device from which to acquire audio data.

Number of channels
: Specify the number of audio channels.

Sample rate (Hz)
: Specify the number of samples per second in the signal.

Device data type
: Specify the data type used by the device to acquire audio data.

Automatically determine buffer size
: Select this check box to enable the block to use a conservative buffer size.

Buffer size (samples)
: Specify the size of the buffer that the block uses to communicate with the audio device. This parameter is visible when the Automatically determine buffer size check box is cleared.

Queue duration (seconds)
: Specify the size of the queue in seconds.

Frame size (samples)
: Specify the number of samples in the block's output signal.

Output data type
: Select the data type of the block's output.

Supported Data Types

Port	Supported Data Types
Output	• Double-precision ßoating point • Single-precision ßoating point • 32-bit signed integers • 16-bit signed integers • 8-bit unsigned integers

See Also

From Wave File	Signal Processing Blockset
To Audio Device	Signal Processing Blockset
audiorecorder	MATLAB

From Multimedia File

Purpose

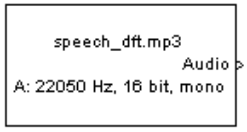
Read video frames and/or aud io samples from multimedia ple

Library

Signal Processing Sources

dspsrcs4

Description



The From Multimedia File block reads audio and video, or audio samples only or video frames only from a multimedia ple and imports them into a Simulink model. Video pr ocessing requires the Video and Image Processing Blockset™ product.

You can view the video frames using a To Video Display block and listen to the audio using a To Audio Device block.

Notes

- This block supports code generation for the host computer that has ple I/O available. This excludes RTWin (Real-Time Windows Target™ software, which does not support ple I/O).
- On the UNIX ® product platforms, this block supports only uncompressed AVI ples. Also, it does not support OpenDML extensions to the AVI stan dard. For example, AVI ples must be less than 4 gigabytes.
- This block performs best on platforms with Version 9.0 or later of DirectX ® software and Version 9.0 or later of Windows Media ® software.

The output ports of the From Multimedia File block change according to the content of the multimedia ple. If the ple contains video frames, the Image port appears on the block. If the ple contains audio samples, the Audio port appears on the block.

Port	Description
Image	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes
R, G, B	Matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B port must have same dimensions.
Audio	Vector of audio data

For sink blocks to display video data properly, double- and single-precision ßoating-point pixel values must be between 0 and 1. For other data types, the pixel va lues must be between the minimum and maximum values supported by their data type.

Use the File name parameter to specify the name of the multimedia Þle from which to read. If the location of this Þle is on your MATLAB path, enter the Þlename. If the location of this Þle is not on your MATLAB path, use the Browse button to specify the full path to the Þle as well as the Þlename. On Windows platforms, this parameter supports URLs that point to multimedia Þles. For more information, see “Supported File Formats – Windows Platforms” on page 2-596.

Select the Inherit sample time from **fi**le check box if you want the sample time of the block to be the same as the sample time of the multimedia Þle. If you clear this check box, use the Desired sample time parameter to specify th e block’s sample time.

The default sample time for the From Multimedia File block is determined by the Þle that it references. You can also set the sample time for this block manually. Unless you already know the video’s intended sample rate, set the sample rate from the Þle. If you have other sources in your model which nee d to operate at the same rate, you can set them to inherit the sample rate.

The table below provides the Sample time calculations that the From Multimedia File block uses for video and audio Þles:

For Video Files	For Audio Files	For Video and Audio Files
Sample time $= \frac{1}{FPS}$ where FPS is the Frames per Second.	Sample time $= \frac{1024}{SampleRate}$ where 1024 is the size of the audio frame, set by the block.	Sample time $= \frac{ceil(AudioSampleRate / FPS)}{AudioSampleRate}$ When audio sample time is $\frac{AudioSampleRate}{FPS}$ is non-integer, the calculation does not equal $\frac{1}{FPS}$. In this case, to prevent synchronization problems, the corresponding video frame is dropped when the audio stream leads the video stream by more than $\frac{1}{FPS}$.

Use the Number of times to play file parameter to enter the number of times to play the file. The number you enter must be a positive integer or inf.

Use the Output end-of-file indicator parameter to determine when the last video frame or audio sample in the multimedia file is output from the block. When you select this check box, a Boolean output port labeled EOF appears on the block. The output from the EOF port is 1 when the last video frame or audio sample is output from the block. Otherwise, the output from the EOF port is 0.

Use the Image color space parameter to specify whether you want the block to output RGB or intensity video frames. If you select RGB, use the Image signal parameter to specify how to output a color signal. If you select One multidimensional signal, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes,

at one port. If you select Separate color signals , additional ports appear on the block. Each port outputs one M-by-N plane of an RGB video stream.

Use the Audio output data type parameter to set the data type of the audio samples output at the Audio port. You can choose double , single , int16 , or uint8 .

Use the Video output data type parameter to set the data type of the video frames output at the R, G, B, or Image ports. You can choose double , single , int8 , uint8 , int16 , uint16 , int32 , uint32 , or Inherit from file .

The block outputs one video frame at each Simulink time step. To calculate the number of audio samples to output at each time step, the block divides the audio sample rate by the video frame rate (fps). In the event that the audio sample rate is not evenly divisible by the number of video frames per second, two things occur:

- The block rounds the number of aud io samples up to the nearest whole number.
- If necessary, the block will periodically drop a video frame to maintain synchronization for large audio and video Þles.

Troubleshooting

There are two issues that you need to be aware of when using the From Multimedia File block.

1 On Windows XP x64 platforms, the From Multimedia File block might not be able to read so me compressed multimedia Þles because Windows XP x64 ships with a limited set of 64-bit video and audio codecs.

   There are two known workarounds:

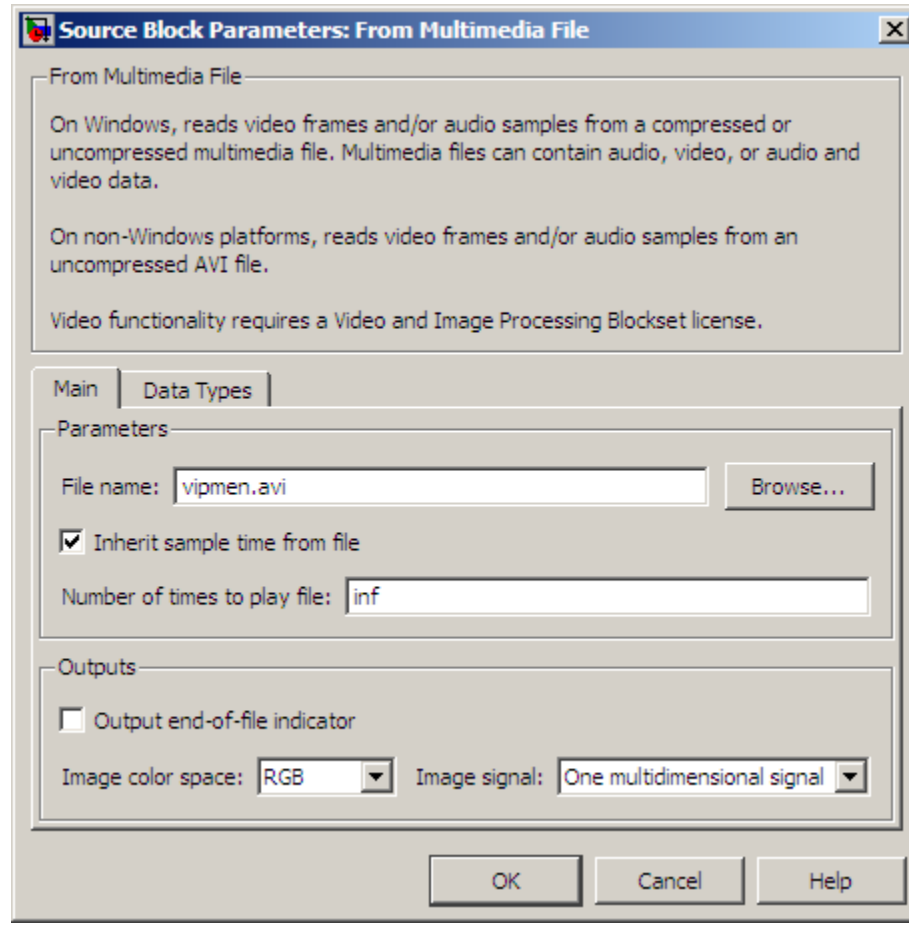
   a Run the 32-bit version of MATLAB on your Windows XP x64 platform. Windows XP x64 ships with many 32-bit codecs that might work with your compressed multimedia Þle.

b Change the multimedia Þle to a different format using a codec that is supported on the Windows XP x64 platform.

2 When working with the From Multimedia File block, you might encounter this error: "ClassFactory cannot supply requested class". The From Multimedia File b lock uses the Microsoft ® DirectX ® product infrastructure, which does not allow you to play some AVI Þles from a remote network location.

To workaround this issue, copy your AVI Þle to a local hard disk before using it with the From Multimedia File block.

Dialog Box



File name

Specify the name of the multimedia file from which to read. The block determines the type of file, for example audio and video, or audio or video only, and provides the associated parameters.

Inherit sample time from file
: Select this check box if you want the sample time of the block to be the same as the sample time of the multimedia file.

Desired sample time
: Specify the block's sample time. This parameter is available if you clear the Inherit sample time from file check box.

Number of times to play file
: Enter a positive integer or inf to represent the number of times to play the file.

Output end-of-file indicator
: Use this check box to determine whether the output is the last video frame or audio sample in the multimedia file.

Multimedia outputs
: Specify Video and audio, Video only, or Audio only output file type. This parameter is available only when a video signal has both audio and video.

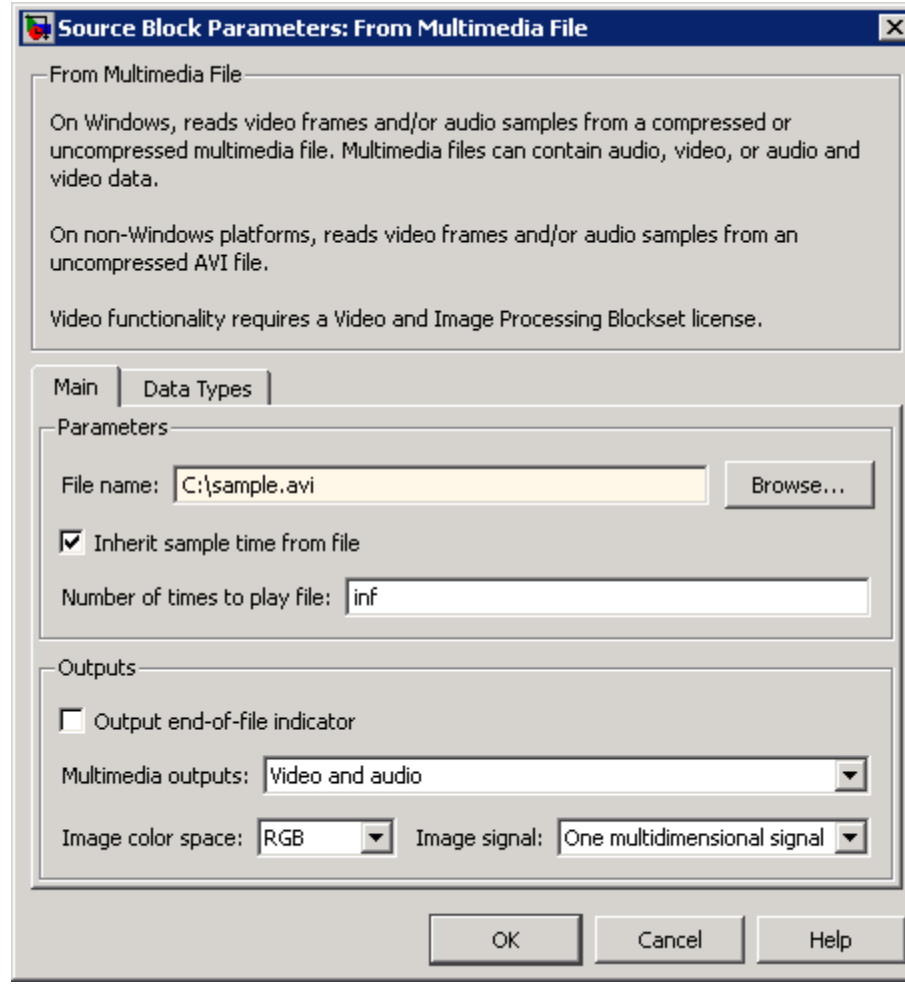


Image color space

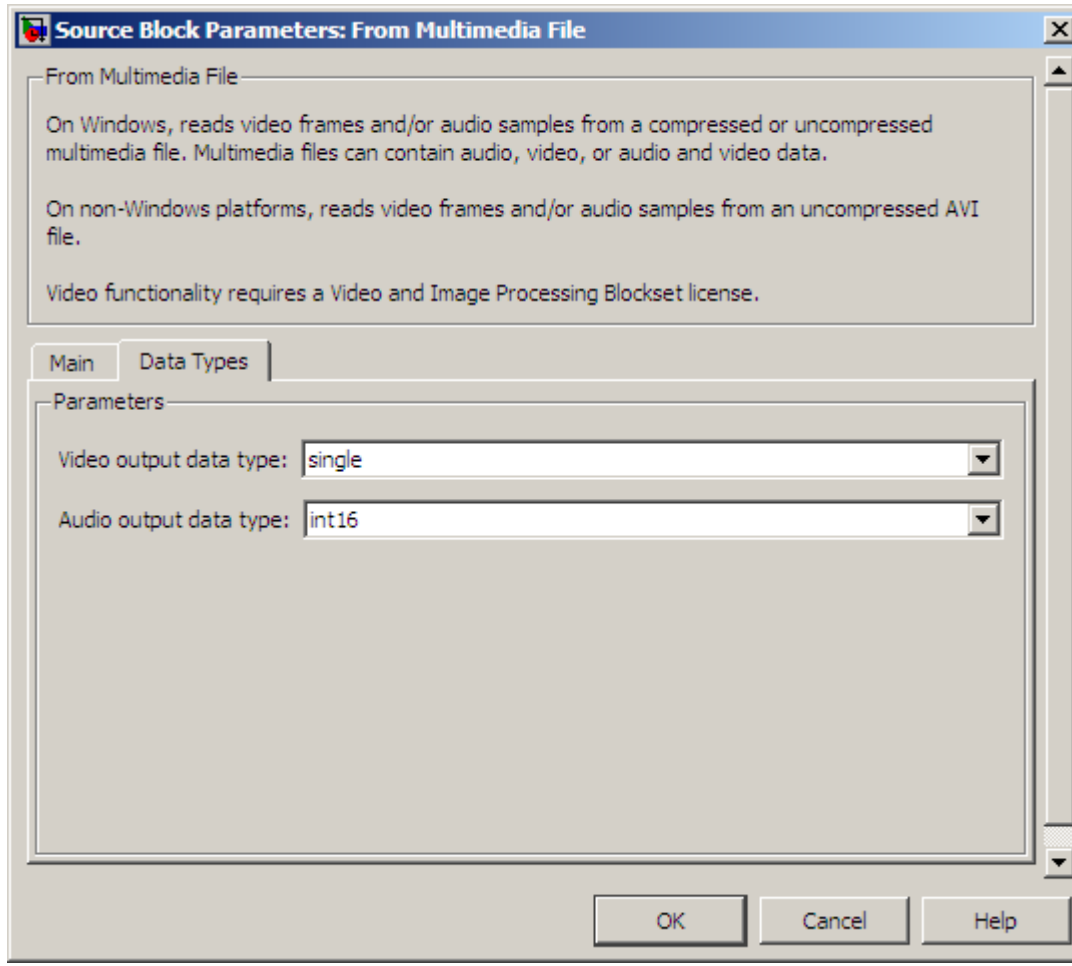
Specify whether you want the block to output RGB or Intensity video frames. This parameter is available for a signal which contains video.

Image signal

Specify how to output a color video signal. If you select One multidimensional signal, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select Separate color signals, additional ports appear on the block. Each port outputs one M-by-N plane of an RGB video stream. This parameter is only available if you set the Image color space parameter to RGB, and the signal contains video.

Samples per audio frame

Specify number of samples per audio frame. This parameter is only available for audio only files.



Audio output data type

Set the data type of the audio samples output at the Audio port. This parameter is only ava ilable if the multimedia Þle contains audio.

Video output data type

Set the data type of the video da ta output from the block. This parameter is only available if the multimedia Þle contains video.

Supported File Formats – Windows Platforms

This block can support the following Þle formats provided you have the necessary codecs installed on your system.

Format	Filename Extensions
Apple® QuickTime, Macintosh® AIFF Resource	.qt, .aif, .aifc, .aiff, .mov
Microsoft Windows Media formats	.avi, .asf, .asx, .rmi, .wav, .wma, .wax, .wmv
Moving Picture Experts Group (MPEG)	.mpg, .mpeg, .mlv, .mp2, .mp3, .mpa, .mpe
Audio formats	.au, .snd

Supported File Formats – UNIX Platforms

Format	Filename Extensions
Microsoft Windows Media formats (uncompressed)	.avi

Supported Data Types

Port	Supported Data Types	Supports Complex Values?
Image	• Double-precision ßoating point • Single-precision ßoating point • 8-, 16-, and 32-bit signed integers	

Port	Supported Data Types	Supports Complex Values?
	• 8-, 16-, and 32-bit unsigned integers	
R, G, B	Same as the Image port	No
Audio	• Double-precision ßoating point • Single-precision ßoating point • 16-bit signed integers • 8-bit unsigned integers	No

See Also

To Multimedia File	Signal Processing Blockset
From Wave File	Signal Processing Blockset
Image From Workspace	Video and Image Processing Blockset
To Video Display	Video and Image Processing Blockset
Video From Workspace	Video and Image Processing Blockset
Video Viewer	Video and Image Processing Blockset
	Simulink

From Wave Device (Obsolete)

Purpose

Read audio data from standard audio device in real-time (32-bit Windows operating systems only)

Library

dspwin32

Description

Note The From Wave Device block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the From Audio Device block.

The From Wave Device block reads audio data from a standard Windows audio device in real-time. It is compatible with most popular Windows hardware, including Sound Blaster cards. (Models that contain both this block and the To Wave Device block require a duplex-capable sound card.)

The Use default audio device parameter allows the block to detect and use the system's default audio hardware. This option should be selected on systems that have a single sound device installed, or when the default sound device on a multiple-device system is the desired source. In cases when the default sound device is not the desired input source, clear Use default audio device, and select the desired device in the Audio device menu parameter.

When the audio source contains two channels (stereo), the Stereo check box should be selected. When the audio source contains a single channel (mono), the Stereo check box should be cleared. For stereo input, the block's output is an M-by-2 matrix containing one frame (M consecutive samples) of audio data from each of the two channels. For mono input, the block's output is an M-by-1 matrix containing one frame (M consecutive samples) of audio data from the mono input. The frame size, M, is specified by the Samples per frame parameter. For M=1, the output is sample based; otherwise, the output is frame based.

The audio data is processed in uncompressed pulse code modulation (PCM) format, and should typically be sampled at one of the standard Windows audio device rates: 8000, 11025, 22050, or 44100 Hz. You can

select one of these rates from the Sample rate parameter. To specify a different rate, select the User-defined option and enter a value in the User-defined sample rate parameter.

The Sample Width (bits) parameter specifies the number of bits used to represent the signal samples read by the audio device. The following settings are available:

- 8 — allocates 8 bits to each sample, allowing a resolution of 256 levels
- 16 — allocates 16 bits to each sample, allowing a resolution of 65536 levels
- 24 — allocates 24 bits to each sample, allowing a resolution of 16777216 levels (only for use with 24-bit audio devices)

Higher sample width settings require more memory but yield better fidelity. The output from the block is independent of the Sample width (bits) setting. The output data type is determined by the Data type parameter setting.

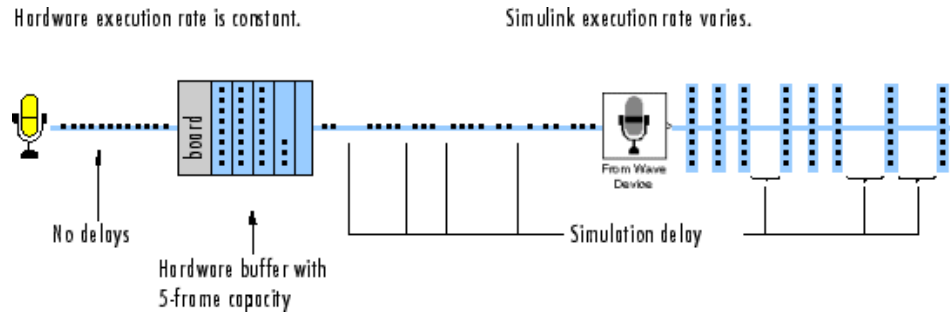
Buffering

Since the audio device accepts real-time audio input, Simulink software must read a continuous stream of data from the device throughout the simulation. Delays in reading data from the audio hardware can result in hardware errors or distortion of the signal. This means that the From Wave Device block must read data from the audio hardware as quickly as the hardware itself acquires the signal. However, the block often cannot match the throughput rate of the audio hardware, especially when the simulation is running from within Simulink rather than as generated code. (Simulink operations are generally slower than comparable hardware operations, and execution speed routinely varies during the simulation as the host operating system services other processes.) The block must therefore rely on a buffering strategy to ensure that signal data can be read on schedule without losing samples.

At the start of the simulation, the audio device begins writing the input data to a (hardware) buffer with a capacity of T_b seconds. The From Wave Device block immediately begins pulling the earliest samples off

the buffer (Þrst in, Þrst out) and collecting them in length- M frames for output. As the audio device continues to append inputs to the bottom of the buffer, the From Wave Device block continues to pull inputs off the top of the buffer at the best possible rate.

The following Þgure shows an audio signal being acquired and output with a frame size of 8 samples. The buffer of the sound board is approaching its Þve-frame capacity at the instant shown, which means that the hardware is adding samples to the buffer more rapidly than the block is pulling them off. (If the signal sample rate was 8 kHz, this small buffer could hold approximately 0.005 second of data.



When the simulation throughput rate is higher than the hardware throughput rate, the buffer remains empty throughout the simulation. If necessary, the From Wave Device bl ock simply waits for new samples to become available on the buffer (the block does not interpolate between samples). More typically, the simulation throughput rate is lower than the hardware throughput rate, and the buffer tends to Þll over the duration of the simulation.

Troubleshooting

When the buffer size is too small in relation to the simulation throughput rate, the buffer might Þll before the entire length of signal is processed. This usually results in a device error or undesired device output. When this problem occurs, y ou can choose to either increase the buffer size or the simulation throughput rate:

- Increase the buffer size

  The Queue duration parameter specifies the duration of signal, T_b (in real-time seconds), that can be buffered in hardware during the simulation. Equivalently, this is the maximum length of time that the block's data acquisition can lag the hardware's data acquisition. The number of frames buffered is approximately

  $$\frac{T_b F_s}{M}$$

  where F_s is the sample rate of the signal and M is the number of samples per frame. The required buffer size for a given signal depends on the signal length, the frame size, and the speed of the simulation. Note that increasing the buffer size might increase model latency.

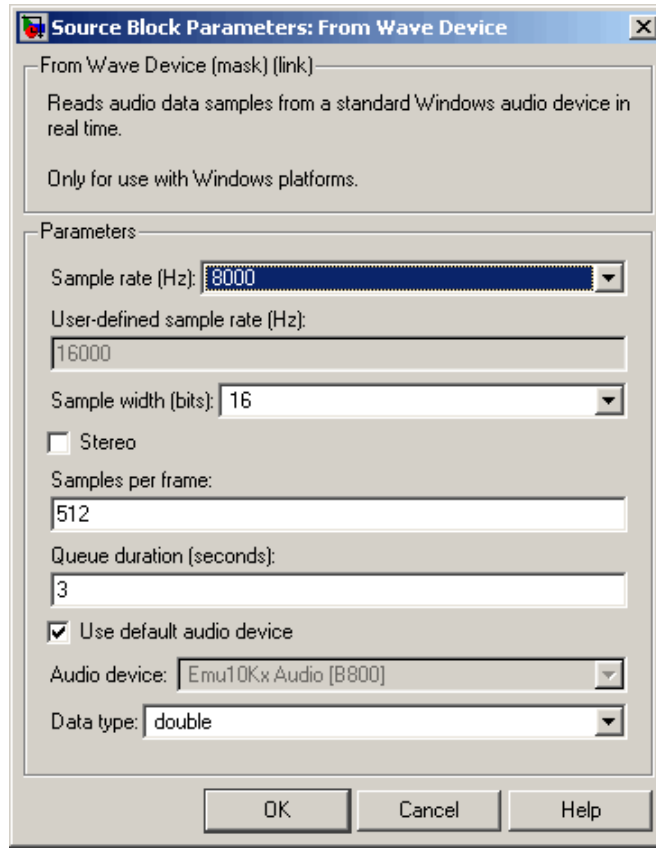
- Increase the simulation throughput rate

  Two useful methods for improving simulation throughput rates are increasing the signal frame size and compiling the simulation into native code:

  - Increase frame sizes (and convert sample-based signals to frame-based signals) throughout the model to reduce the amount of block-to-block communication overhead. This can drastically increase throughput rates in many cases. However, larger frame sizes generally result in greater model latency due to initial buffering operations.
  - Generate executable code with Real Time Workshop. Native code runs much faster than Simulink, and should provide rates adequate for real-time audio processing.

More general ways to improve throughput rates include simplifying the model, and running the simulation on a faster PC processor. See "Delay and Latency" and "Improving Simulation Performance and Accuracy" in the Simulink documentation for other ideas on improving simulation performance.

Dialog Box



Sample rate (Hz)

The sample rate of the audio data to be acquired. Select one of the standard Windows rates or the User-defined option.

User-defined sample rate (Hz)

The (nonstandard) sample rate of the audio data to be acquired.

Sample width (bits)

The number of bits used to represent each signal sample.

Stereo
: Specifies stereo (two-channel) inputs when selected, mono (one-channel) inputs when cleared. Stereo output is M-by-2; mono output is M-by-1.

Samples per frame
: The number of audio samples in each successive output frame, M. When the value of this parameter is 1, the block outputs a sample-based signal.

Queue duration (seconds)
: The length of signal (in seconds) to buffer to the hardware at the start of the simulation.

Use default audio device
: Reads audio input from the system's default audio device when selected. Clear to enable the Audio device ID parameter and select a device.

Audio device
: The name of the audio device from which to read the audio output (lists the names of the installed audio device drivers). Select Use default audio device when the system has only a single audio card installed.

Data type
: The data type of the output: double-precision, single-precision, signed 16-bit integer, or unsigned 8-bit integer.

Supported Data Types

- Double-precision floating point
- Single-precision floating point
- 16-bit signed integer
- 8-bit unsigned integer

See Also

From Wave File	Signal Processing Blockset
To Wave Device (Obsolete)	Signal Processing Blockset
audiorecorder	MATLAB

Purpose

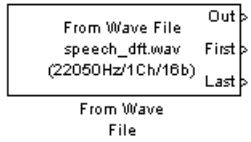
Read audio data from Microsoft Wave (.wav) Þle

Library

Signal Processing Sources

dspsrcs4

Description



The From Wave File block streams a udio data from a Microsoft Wave (.wav) Þle and generates a signal with one of the data types and amplitude ranges in the following table.

Output Data Type	Output Amplitude Range
double	±1
single	±1
int16	-32768 to 32767 (-2^{15} to 2^{15} - 1)
uint8	0 to 255

The audio data must be in uncompressed pulse code modulation (PCM) format.

```
y = wavread('filename')          % Equivalent MATLAB code
```

The block supports 8-, 16-, 24-, and 32-bit Microsoft Wave (.wav) Þles.

The File name parameter can specify an absolute or relative path to the Þle. When the Þle is on the MATLAB path or in the current directory (the directory returned by typing pwd at the MATLAB command line), you need only specify the Þle's name. You do not need to specify the .wav extension.

For an audio Þle containing C channels, the block's output is an M-by-C matrix containing one frame (M consecutive samples) of audio data from each channel. The frame size, M, is speciÞed by the Samples per output frame parameter. For M=1, the output is sample based; otherwise, the output is frame based.

The output frame period, T_{fo}, is

$$T_{fo} = \frac{M}{F_s}$$

where F_s is the data sample rate in Hz.

To reduce the required number of file accesses, the block acquires L consecutive samples from the file during each access, where L is specified by the Minimum number of samples for each read from file parameter ($L \geq M$). For $L < M$, the block instead acquires M consecutive samples during each access. Larger values of L result in fewer file accesses, which reduces run-time overhead.

Use the Data type parameter to specify the data type of the block's output. Your choices are double , single , uint8 , or int16 .

Select the Loop check box if you want to play the file more than once. Then, enter the number of times to play the file. The number you enter must be a positive integer or inf .

Use the Number of times to play file parameter to enter the number of times to play the file. The number you enter must be a positive integer or inf , to play the file until you stop the simulation.

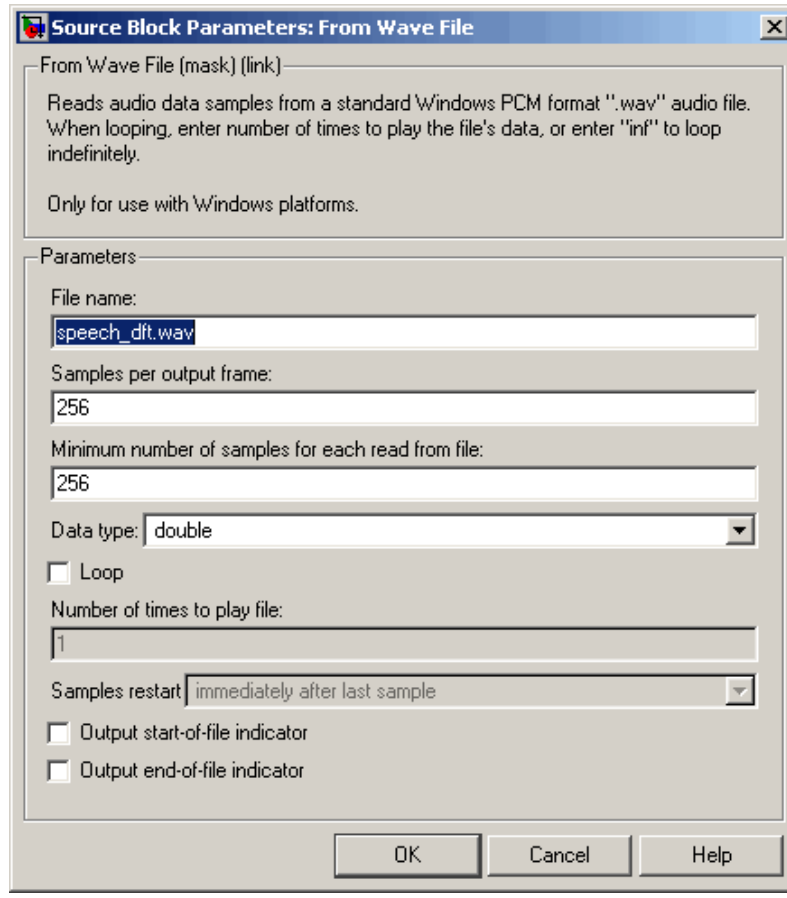
The Samples restart parameter determines whether the samples from the audio file repeat immediately or repeat at the beginning of the next frame output from the output port. When you select immediately after last sample , the samples repeat immediately. When you select at beginning of next frame , the frame containing the last sample value from the audio file is zero padded until the frame is filled. The block then places the first sample of the audio file in the first position of the next output frame.

Use the Output start-of-file indicator parameter to determine when the first audio sample in the file is output from the block. When you select this check box, a Boolean output port labeled SOF appears on the block. The output from the SOF port is 1 when the first audio sample in the file is output from the block. Otherwise, the output from the SOF port is 0.

Use the Output end-of-file indicator parameter to determine when the last audio sample in the file is output from the block. When you select this check box, a Boolean output port labeled EOF appears on the block. The output from the EOF port is 1 when the last audio sample in the file is output from the block. Otherwise, the output from the EOF port is 0.

The block icon shows the name, sample rate (in Hz), number of channels (1 or 2), and sample width (in bits) of the data in the specified audio file. All sample rates are supported; the sample width must be either 8, 16, 24, or 32 bits.

Dialog Box



File name

Enter the path and name of the ple to read. Paths can be relative or absolute.

Samples per output frame

Enter the number of samples in each output frame, M. When the value of this parameter is 1, the block outputs a sample-based signal.

Minimum number of samples for each read from file
: Enter the number of consecutive samples to acquire from the file with each file access, L.

Data type
: Select the output data type: double, single, uint8, or int16. The data type setting determines the output's amplitude range, as shown in the preceding table.

Loop
: Select this check box if you want to play the file more than once.

Number of times to play file
: Enter the number of times you want to play the file.

Samples restart
: Select immediately after last sample to repeat the audio file immediately. Select at beginning of next frame to place the first sample of the audio file in the first position of the next output frame.

Output start-of-file indicator
: Use this check box to determine whether the output contains the first audio sample in the file.

Output end-of-file indicator
: Use this check box to determine whether the output contains the last audio sample in the file.

Supported Data Types

- Double-precision floating point
- Single-precision floating point
- 16-bit signed integer
- 8-bit unsigned integer

See Also

From Audio Device	Signal Processing Blockset
Signal From Workspace	Signal Processing Blockset

To Wave File	Signal Processing Blockset
wavread	MATLAB

Purpose

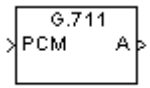
Quantize narrowband speech input signals

Library

Quantizers

dspquant2

Description



G.711 Codec

The G711 Codec block is a logarithmic scalar quantizer designed for narrowband speech. Narrowband speech is defined as a voice signal with an analog bandwidth of 4 kHz and a Nyquist sampling frequency of 8 kHz. The block quantizes a narrowband speech input signal so that it can be transmitted using only 8-bits. The G711 Codec block has three modes of operation: encoding, decoding, and conversion. You can choose the block's mode of operation by setting the Mode parameter.

If, for the Mode parameter, you choose Encode PCM to A-law, the block assumes that the linear PCM input signal has a dynamic range of 13 bits. Because the block always operates in saturation mode, it assigns any input value above $2^{12}-1$ to $2^{12}-1$ and any input value below -2^{12} to -2^{12}. The block implements an A-law quantizer on the input signal and outputs A-law index values. When you choose Encode PCM to mu-law, the block assumes that the linear PCM input signal has a dynamic range of 14 bits. Because the block always operates in saturation mode, it assigns any input value above $2^{13}-1$ to $2^{13}-1$ and any input value below -2^{13} to -2^{13}. The block implements a mu-law quantizer on the input signal and outputs mu-law index values.

If, for the Mode parameter, you choose Decode A-law to PCM, the block decodes the input A-law index values into quantized output values using an A-law lookup table. When you choose Decode mu-law to PCM, the block decodes the input mu-law index values into quantized output values using a mu-law lookup table.

If, for the Mode parameter, you choose Convert A-law to mu-law, the block converts the input A-law index values to mu-law index values. When you choose Convert mu-law to A-law, the block converts the input mu-law index values to A-law index values.

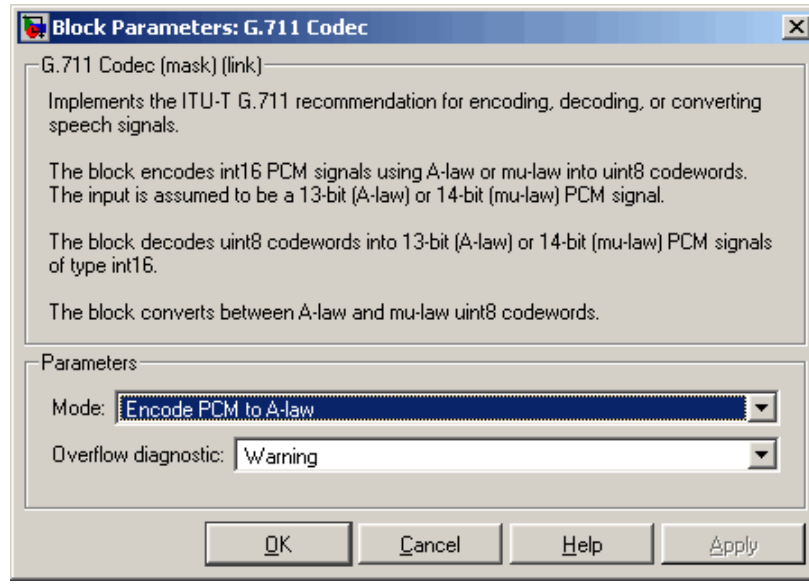
Note Set the Mode parameter to Convert A-law to mu-law or Convert mu-law to A-law only when the input to the block is A-law or mu-law index values.

If, for the Mode parameter, you choose Encode PCM to A-law or Encode PCM to mu-law, the Overflow diagnostic parameter appears on the block parameters dialog box. Use this parameter to determine the behavior of the block when overflow occurs. The following options are available:

- Ignore — Proceed with the computation and do not issue a warning message.
- Warning — Display a warning message in the MATLAB Command Window, and continue the simulation.
- Error — Display an error dialog box and terminate the simulation.

Note Like all diagnostic parameters on the Configuration Parameters dialog box, Overflow diagnostic parameter is set to Ignore in the code generated for this block by Real-Time Workshop code generation software.

Dialog Box



Mode

- When you choose Encode PCM to A-law , the block implements an A-law encoder.
- When you choose Encode PCM to mu-law , the block implements a mu-law encoder.
- When you choose Decode A-law to PCM , the block decodes the input index values into quantized output values using an A-law lookup table.
- When you choose Decode mu-law to PCM , the block decodes the input index values into quantized output values using a mu-law lookup table.
- When you choose Convert A-law to mu-law , the block converts the input A-law index values to mu-law index values.
- When you choose Convert mu-law to A-law , the block converts the input mu-law index values to A-law index values.

Overflow diagnostic

Use this parameter to determine the behavior of the block when overflow occurs.

- Select Ignore to proceed with the computation without a warning message.
- Select Warning to display a warning message in the MATLAB Command Window and continue the simulation.
- Select Error to display an error dialog box and terminate the simulation.

This parameter is only visible if, for the Mode parameter, you select Encode PCM to A-law or Encode PCM to mu-law.

References

ITU-T Recommendation G.711, "Pulse Code Modulation (PCM) of Voice Frequencies," General Aspects of Digital Transmission Systems; Terminal Equipments, International Telecommunication Union (ITU), 1993.

Supported Data Types

Port	Supported Data Types
PCM	• 16-bit signed integers
A	• 8-bit unsigned integers
mu	• 8-bit unsigned integers

See Also

Quantizer	Simulink
Scalar Quantizer Decoder	Signal Processing Blockset
Scalar Quantizer Design	Signal Processing Blockset
Uniform Decoder	Signal Processing Blockset
Uniform Encoder	Signal Processing Blockset
Vector Quantizer Decoder	Signal Processing Blockset

Vector Quantizer Design
Vector Quantizer Encoder
S ignal Processing Blockset
Signal Processing Blockset

Halfband Filter

Purpose

Design halfband filter

Library

Filtering / Filter Design Toolbox

dspfdesign

Description

This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See “Halfband Filter Design Dialog Box — Main Pane” in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset product.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point

Port	Supported Data Types
	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Highpass Filter

Purpose

Design highpass filter

Library

Filtering / Filter Design Toolbox

dspfdesign

Description

This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See "Highpass Filter Design Dialog Box — Main Pane" in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset product.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point

Port	Supported Data Types
	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Hilbert Filter

Purpose

Design Hilbert filter

Library

Filtering / Filter Design Toolbox

dspfdesign

Description

This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See "Hilbert Filter Design Dialog Box — Main Pane" in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset product.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point

Port	Supported Data Types
	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Histogram

Purpose

Generate histogram of input or sequence of inputs

Library

Statistics

dspstat3

Description

The Histogram block computes the frequency distribution of the elements in a vector input, of the elements in each channel of a frame-based matrix input, or of the elements in a sample based N-D array. The Running histogram parameter selects between basic operation and running operation, de scribed below. The Histogram block accepts real and complex Þxed-point and ßoating-point inputs.

The block distributes the elements of the input into the number of discrete bins speci Þed by the Number of bins parameter, n.

```
y = hist(u,n)        % Equivalent MATLAB code
```

Complex Þxed-point inputs are distribute d according to their magnitude squared values; complex ßoating-point inputs are distributed by their normalized values.

The histogram value for a given bin represents the frequency of occurrence of the input values bra cketed by that bin. You specify the upper boundary of the highest-valued bin in the Upper limit of histogram parameter, B_M, and the lower boundary of the lowest-valued bin in the Lower limit of histogram parameter, B_m. The bins have equal width of

$$\Delta = \frac{B_M - B_m}{n}$$

and centers located at

$$B_m + \left(k + \frac{1}{2}\right)\Delta \qquad k = 0,1,2,\ldots,n-1$$

Input values that fall on the border between two bins are placed into the lower valued bin; that is, each bin includes its upper boundary. For example, a bin of width 4 centered on the value 5 contains the input value 7, but not the input value 3. Input values greater than the Upper limit of histogram parameter or less than Lower limit of histogram parameter are placed into the highest valued or lowest valued bin, respectively.

The values you enter for the Upper limit of histogram and Lower limit of histogram parameters must be real-valued scalars. NaN and inf are not valid values for the Upper limit of histogram and Lower limit of histogram parameters.

Basic Operation

When the Running histogram check box is not selected, the Histogram block computes the frequency distribution of the current input.

For frame-based M-by-N inputs, (including 1-by-N row vectors and M-by-1 column vectors), the Histogram block computes a histogram for each channel of the M-by-N matrix independently. The block outputs an n-by-N matrix, where n is the Number of bins specified in the Histogram block. The jth column of the output matrix contains the histogram for the data in the jth column of the M-by-N input matrix.

For all sample-based N-D input arrays, including length-M 1-D vectors and 1-by-N row vectors, the Histogram block computes the frequency distribution of the input data. The block outputs an n-by-1 vector, where n is the Number of bins specified in the Histogram block.

Running Operation

When you select the Running histogram check box, the Histogram block computes the frequency distribution of both the past and present data for successive inputs. The block resets the histogram (by emptying all of the bins) when it detects a reset event at the optional Rst port. See “Resetting the Running Histogram” on page 2-624 for more information on how to trigger a reset.

For frame-based M-by-N inputs (including 1-by-N row vectors and M-by-1 column vectors), the Histogram block computes a running

histogram for each channel of the M-by-N matrix. The block outputs an n-by-N matrix, where n is the Number of bins specified in the Histogram block. The jth column of the output matrix contains the running histogram for the jth column of the M-by-N input matrix.

For all sample-based N-D input arrays, including length-M 1-D vectors, the Histogram block computes a running histogram for the data in the first dimension of the input. The block outputs an n-by-1 vector, where n is the Number of bins specified in the Histogram block.

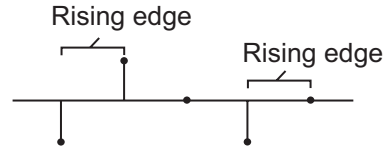
Note When the histogram block is used in running mode and the input data type is non-floating point, the output of the histogram is stored as a uint32 data type. The largest number that can be represented by this data type is 2^{32}-1. If the range of the uint32 data type is exceeded, the output data will wrap back to 0.

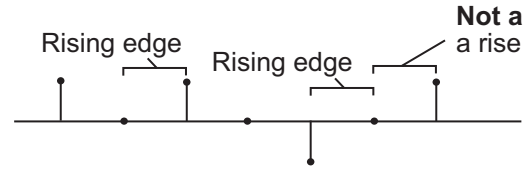
Resetting the Running Histogram

The block resets the running histogram whenever a reset event is detected at the optional Rst port. The reset signal and the input data signal must be the same rate.

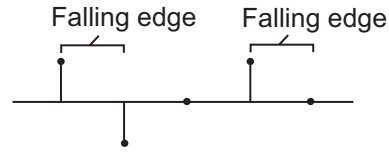
To enable the Rst port, select the Reset port parameter. You specify the reset event in the Trigger type parameter, and it can be one of the following:

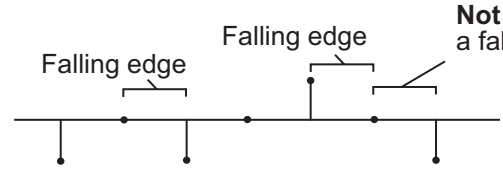
- Rising edge — Triggers a reset operation when the Rst input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following figure)





- Falling edge — Triggers a reset operation when the Rst input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)





- Either edge — Triggers a reset o peration when the Rst input is a Rising edge or Falling edge (as described earlier)
- Non-zero sample — Triggers a reset opera tion at each sample time that the Rst input is not zero

Examples

Real Input Data

The bin boundaries created by the Histogram block are determined by the data type of the input:

- Bin boundaries for real, double-precision input are cast into the data type double.
- Bin boundaries for real, fixed-point input are cast into the int8 data type.

The following example shows the differences in the output of the Histogram block based on the data type of the input.

To create this model you need the following blocks.

Block	Library	Quantity
Constant	Simulink / Sources library	2
Display	Signal Processing Sinks	2
Histogram	Statistics	2

The parameter settings for the Double Precision Input Constant block are:

- Constant value = double([1 2 3 4 5]')
- Interpret parameters as 1-D = Clear this check box.
- Sampling mode = Sample based
- Sample time = inf

The parameter settings for the Fixed-Point Input Constant block are:

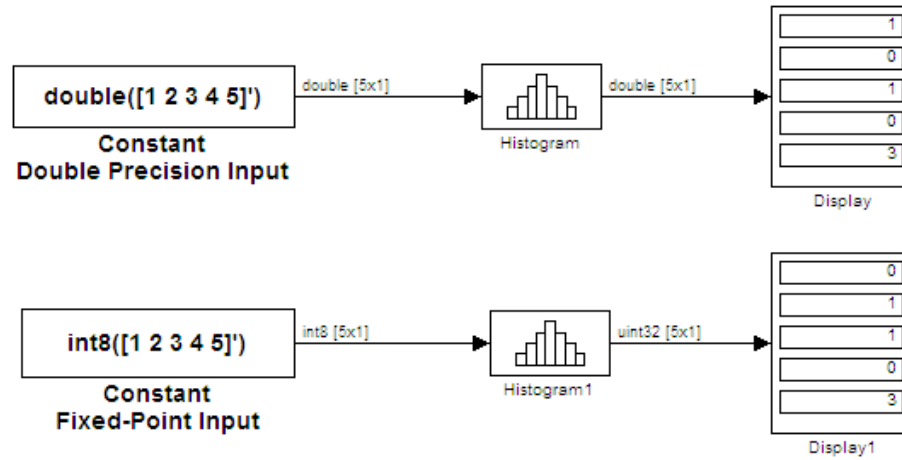
- Constant value = int8([1 2 3 4 5]')
- Interpret parameters as 1-D = Clear this check box.
- Sampling mode = Sample based

- Sample time = inf

The parameter settings for both Histogram blocks are:

- Lower limit of histogram = 1
- Upper limit of histogram = 3
- Number of bins = 5
- Normalized = Clear this check box.
- Running histogram = Clear this check box.

Connect the blocks as shown in the following Þgure, and run your model.

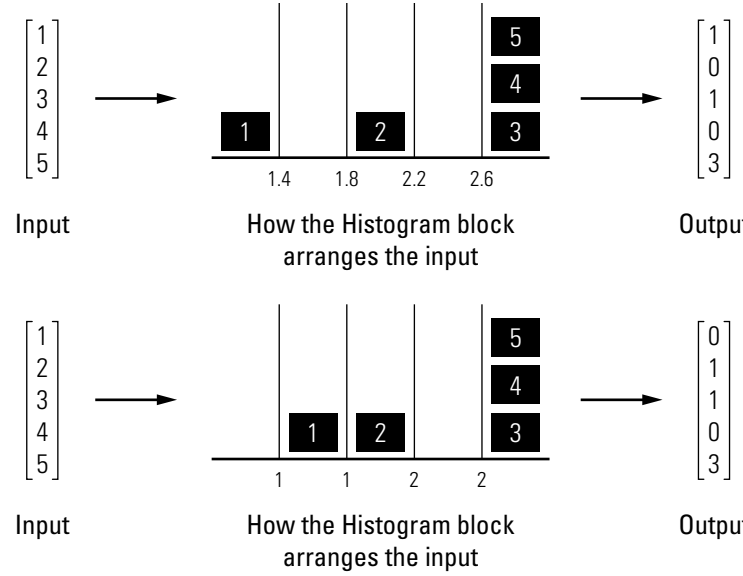


Running this model generates the following warning:

Warning: The bin width resulting from the specified parameters is less than the precision of the input data type. This might cause unexpected results. Since bin width is calculated by ((upper limit - lower limit)/number of bins), you could increase upper limit or decrease lower limit or number of bins.

This warning alerts you that it is not a good use case to have a histogram where 2 or more bin boundaries are the same. As the warning suggests, increasing the range of the limits of the histogram, or decreasing the number of bins, can correct this problem.

The following figures illustrate the different bins that are created by the Histogram block. The top figure shows the histogram for double-precision input, and the bottom figure shows the histogram for fixed-point input. The output of the histogram block differs based on the data type of the input, and the bin boundaries are duplicated in the histogram for the fixed-point input.



Complex Input Data

The bin boundaries created by the Histogram block are determined by the data type of the input.

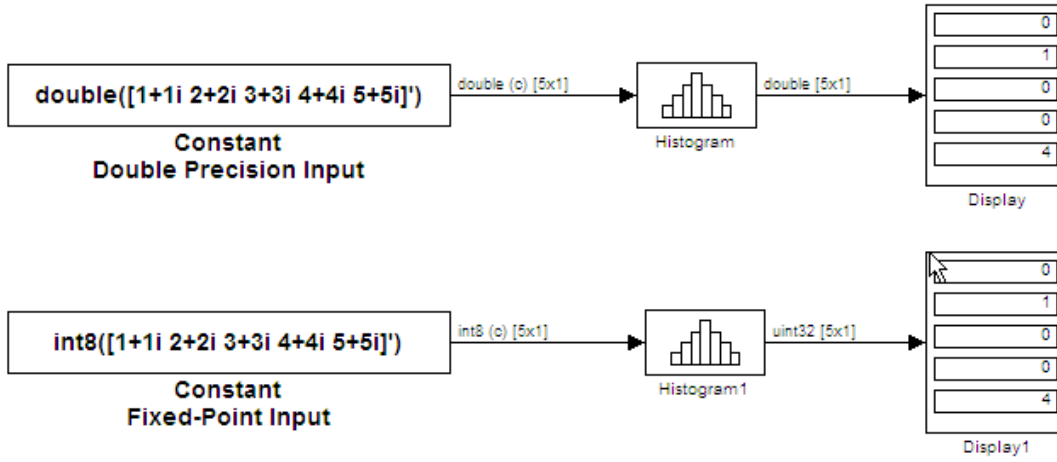
- Bin boundaries for complex, double -precision input are cast into the data type double . All complex, double-pre cision input values are placed in bins according to their normalized values.
- Bin boundaries for complex, Þxed-point input are cast into the data type double and squared. All complex, Þxed-point input values are placed in bins according to the ir magnitude-squared value.

The following example shows the differences in the bins created by the Histogram block based on the data type of the complex input.

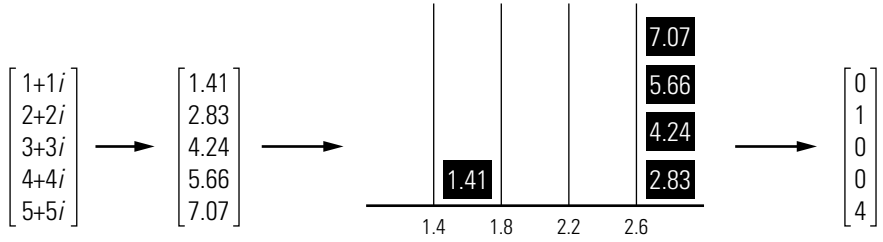
Using the same model you created for the example with real input data, modify the following parameters:

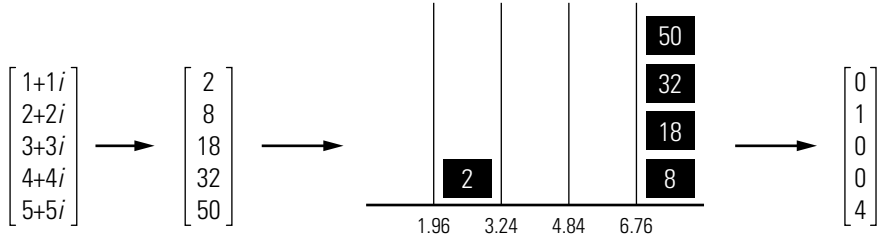
- In the Double Precision Input Constant block, set the Constant value parameter to double([1+1i 2+2i 3+3i 4+4i 5+5i]')
- In the Fixed-Point Input Constant block, set the Constant value parameter to int8([1+1i 2+2i 3+3i 4+4i 5+5i]')

Run your model. It should look similar to the following Þgure:



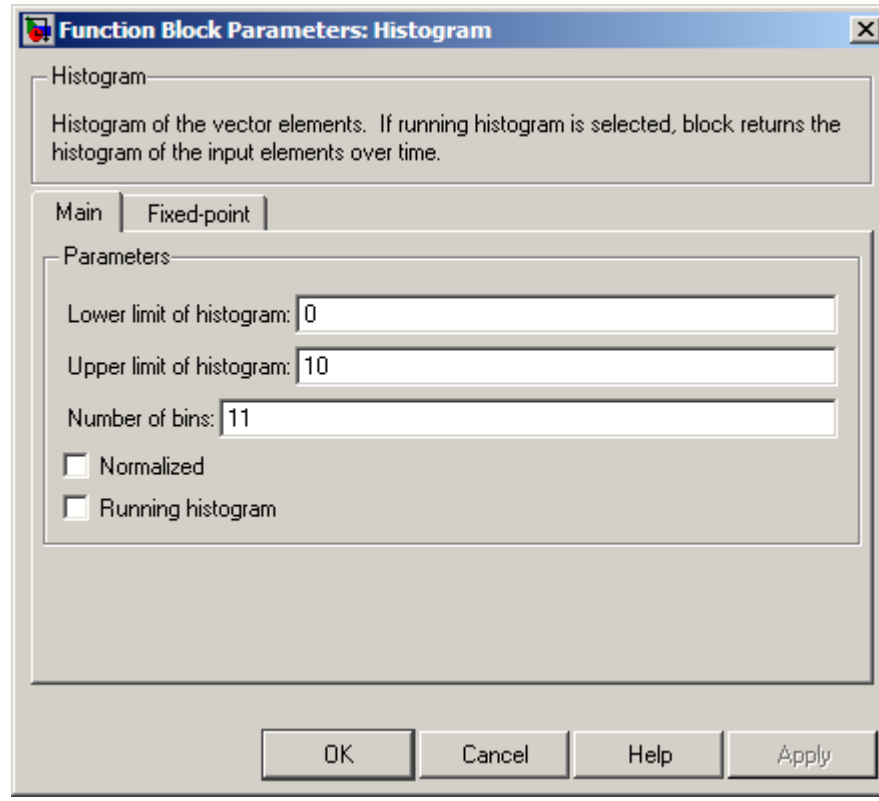
In this case, the Histogram block outputs the same result. The figures below illustrate how the Histogram block compares the input values to the bins it creates. The double-precision inputs are normalized for comparison, whereas the fixed-point inputs are placed using their magnitude squared value. The top figure shows the histogram for the double-precision input, and the bottom figure shows the histogram for the fixed-point input.





Dialog Box

The Main pane of the Histogram block dialog appears as follows.



Lower limit of histogram

Enter a real-valued scalar for the lower boundary, B_m, of the lowest-valued bin. NaN and inf are not valid values for B_m. Tunable.

Upper limit of histogram

Enter a real-valued scalar for the upper boundary, B_M, of the highest-valued bin. NaN and inf are not valid values for B_M. Tunable.

Number of bins
: The number of bins, n, in the histogram.

Normalized
: When selected, the output vector, v, is normalized such that sum(*v*) = 1 .

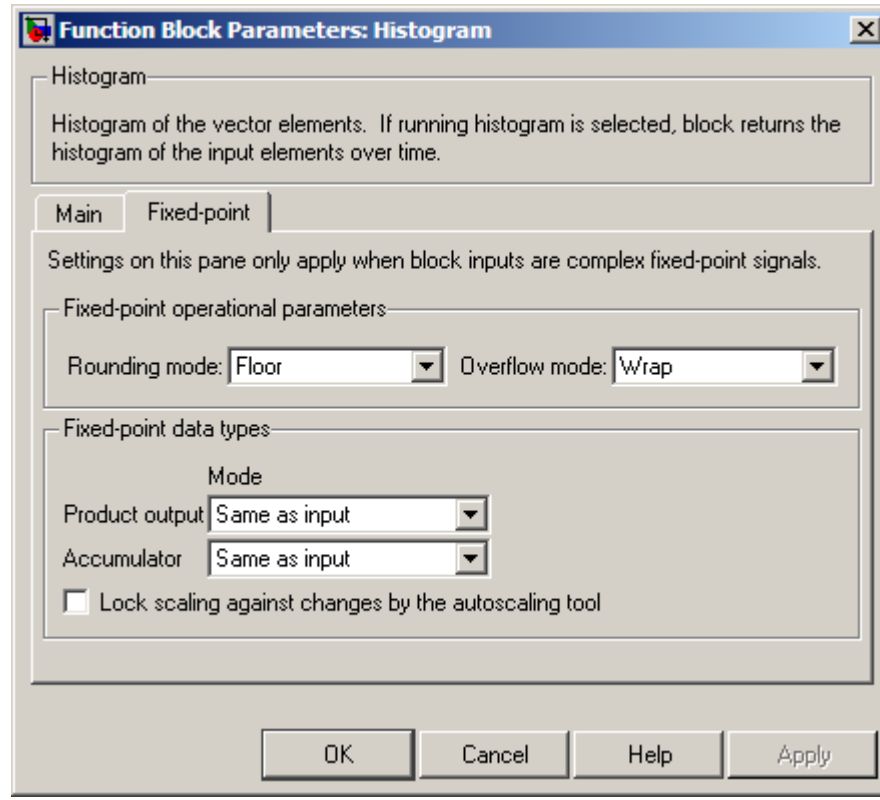
  Use of this parameter is not supported for fixed-point signals.

Running histogram
: Set to enable the running histogram operation, and clear to enable basic histogram operation. For more information, see “Basic Operation” on page 2-623 and “Running Operation” on page 2-623.

Reset port
: Enables the Rst input port when selected. The reset signal and the input data signal must be the same rate. This parameter is enabled only when you set the Running histogram parameter. For more information, see “Running Operation” on page 2-623.

Trigger type
: The type of event that resets the running histogram. For more information, see “Resetting the Running Histogram” on page 2-624. This parameter is enabled only when you set the Reset port parameter.

The Fixed-point pane of the Histogram block dialog appears as follows.



Note The fixed-point parameters listed are only used for fixed-point complex inputs, which are distributed by squared magnitude.

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths:

- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify the accumulator word and fraction lengths resulting from a complex -complex multiplication in the block:

- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any Þxed-point scaling you specify in this block mask f rom being overridden by the autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

Supported Data Types

Port	Supported Data Types
In	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • 32-bit unsigned integers
Rst	• Boolean

See Also

Sort	Signal Processing Blockset
hist	MATLAB

Purpose

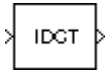
Compute inverse discrete cosi ne transform (IDCT) of input

Library

Transforms

dspxfrm3

Description



The IDCT block computes the inverse discrete cosine transform (IDCT) of each channel in the M-by-N input matrix, u.

```
y = idct(u)        % Equivalent MATLAB code
```

When the input is a sample-based row vector, the IDCT block computes the inverse discrete cosine transform across the vector dimension of the input. For all other sample-based N-D arrays, the block computes the IDCT across the Þrst dimension of the input.

For both sample-based and frame-based inputs, the block assumes that each input column is a frame containing M consecutive samples from an independent channel. The frame size, M, must be a power of two. To work with other frame sizes, use the Pad block to pad or truncate the frame size to a power of two length.

When the input is an M-by-N matrix, the output is an M-by-N matrix whose lth column contains the length- M IDCT of the corresponding input column.

$$y(m,l) = \sum_{k=1}^{M} w(k)u(k,l)\cos\frac{\pi(2m-1)(k-1)}{2M}, \qquad m = 1,\ldots,M$$

where

$$w(k) = \begin{cases} \dfrac{1}{\sqrt{M}}, & k = 1 \\ \sqrt{\dfrac{2}{M}}, & 2 \le k \le M \end{cases}$$

The Output sampling mode parameter allows you to select the sampling mode of the output. If the input is a sample-based N-D array with 3 or more dimensions, the Output sampling mode must be Sample based . The output sample rate and data type (real/complex) are the same as those of the input.

For convenience, length- M 1-D vector inputs and sample-based length- M row vector inputs are processed as single channels (that is, as M -by-1 column vectors), and the output has the same dimension as the input.

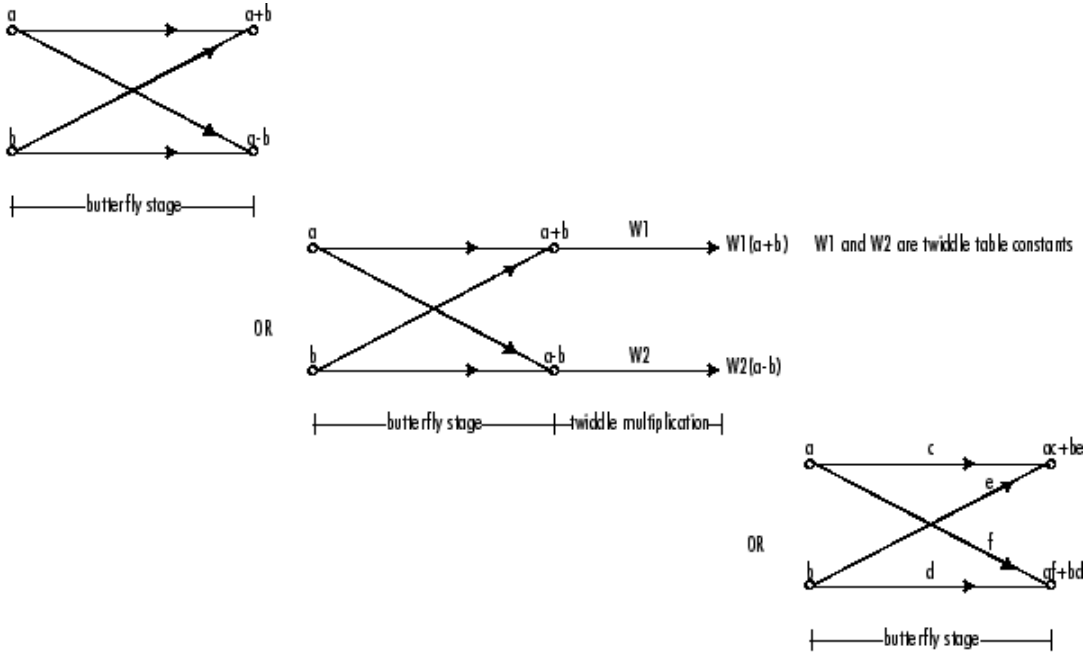
The Sine and cosine computation parameter determines how the block computes the necessary sine an d cosine values. This parameter has two settings, each with its adv antages and disadvantages, as described in the following table.

Sine and Cosine Computation Parameter Setting	Sine and Cosine Computation Method	Effect on Block Performance
Table lookup	The block computes and stores the trigonometric values before the simulation starts, and retrieves them during the simulation. When you generate code from the block, the processor running the generated code stores the trigonometric values computed by the block in a speed-optimized table, and retrieves the values during code execution.	The block usually runs much more quickly, but requires extra memory for storing the precomputed trigonometric values.
Trigonometric fcn	The block computes sine and cosine values during the simulation. When you generate code from the block, the processor running the generated code computes the sine and cosine values while the code runs.	The block usually runs more slowly, but does not need extra data memory. For code generation, the block requires a support library to emulate the trigonometric functions, increasing the size of the generated code.

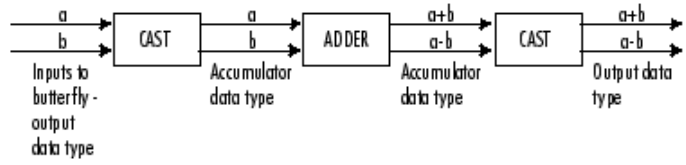
Fixed-Point Data Types

The following diagrams show the data types used within the IDCT block for fixed-point signals. You can set the sine table, accumulator, product output, and output data types displayed in the diagrams in the IDCT block dialog as discussed in “Dialog Box” on page 2-641.

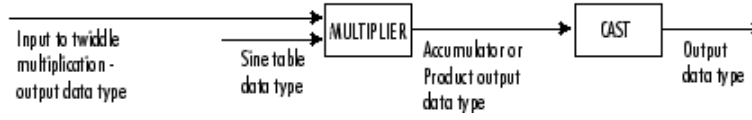
Inputs to the IDCT block are first cast to the output data type and stored in the output buffer. Each butterfly stage processes signals in the accumulator data type, with the final output of the butterfly being cast back into the output data type.



Butterfly Stage Data Types



Twiddle Multiplication Data Types

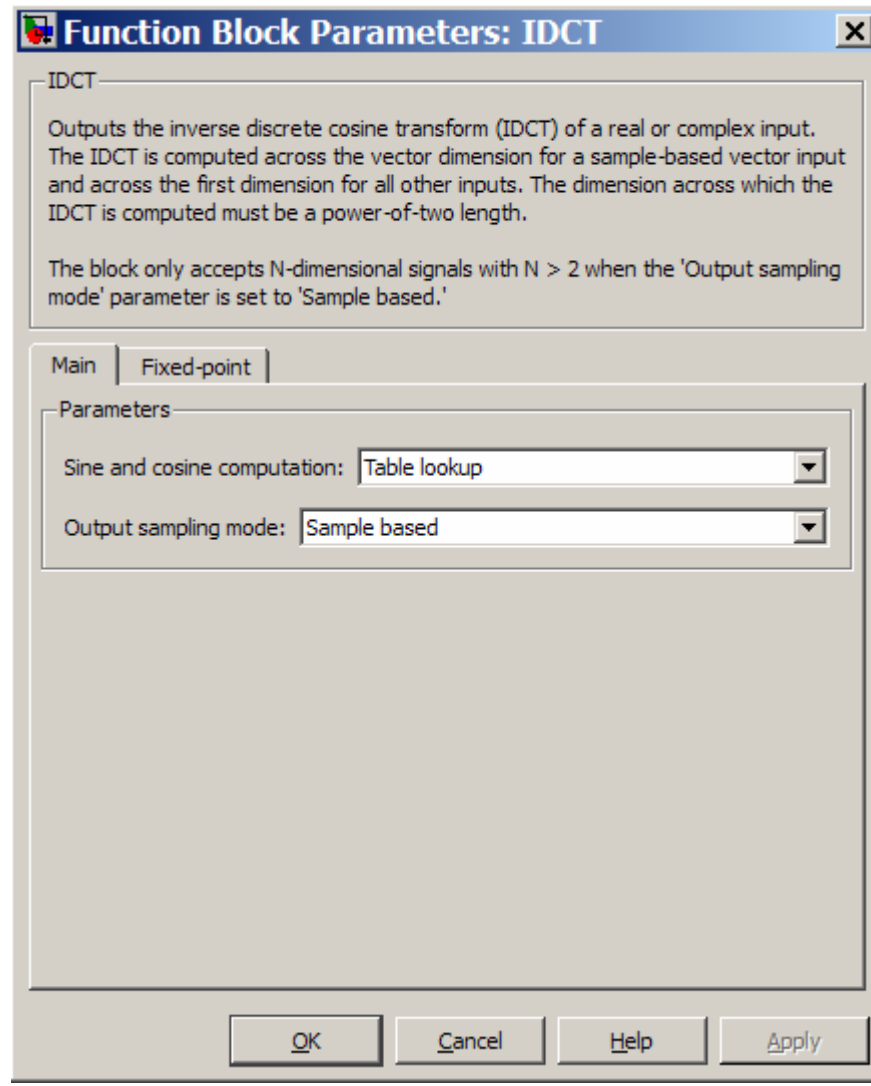


The output of the multiplier is in the product output data type when at least one of the inputs to the multiplier is real. When both of the

inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see “Multiplication Data Types”.

Dialog Box

The Main pane of the IDCT block dialog appears as follows.



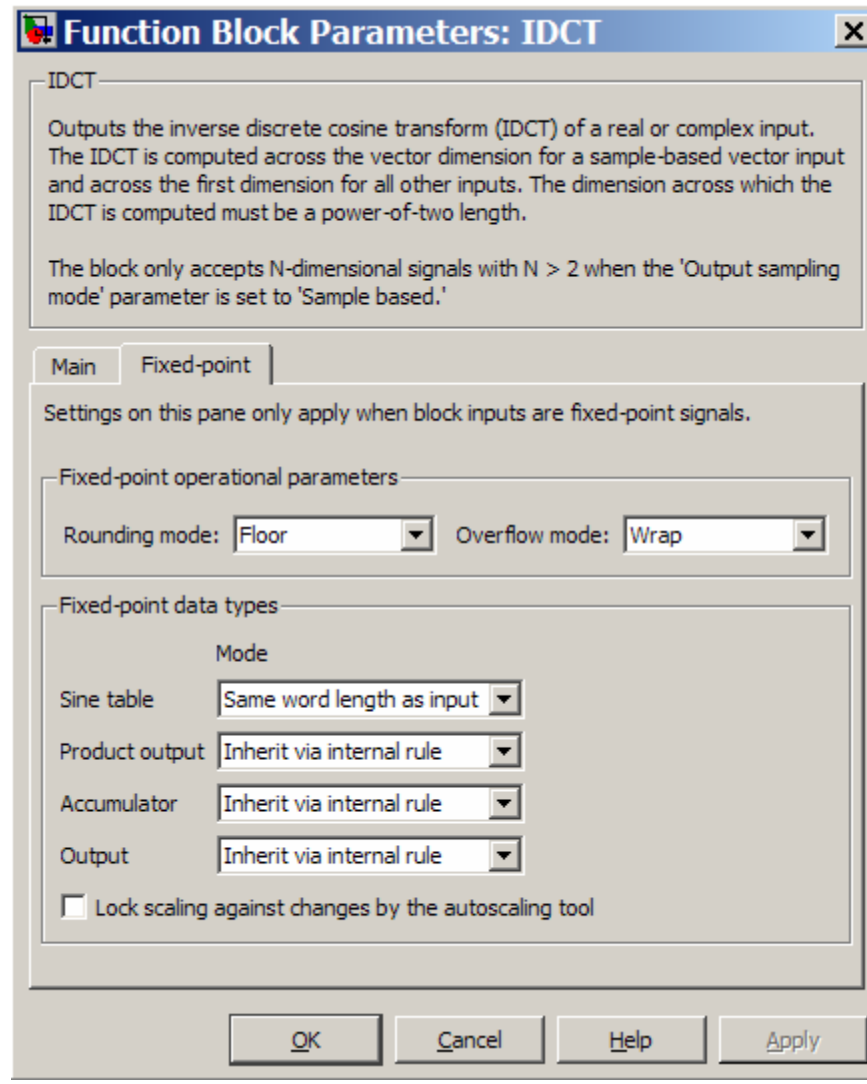
Sine and cosine computation

Sets the block to compute sines an d cosines by either looking up sine and cosine values in a speed-optimized table (Table lookup), or by making sine and cosine function calls (Trigonometric fcn). See the table above.

Output sampling mode

Select Sample based or Frame based output. If the input to the IDCT block has 3 or more dimensions, you must select Sample based output.

The Fixed-point pane of the IDCT block dialog appears as follows.


Function Block Parameters: IDCT
IDCT
Outputs the inverse discrete cosine transform (IDCT) of a real or complex input. The IDCT is computed across the vector dimension for a sample-based vector input and across the first dimension for all other inputs. The dimension across which the IDCT is computed must be a power-of-two length.
The block only accepts N-dimensional signals with N > 2 when the 'Output sampling mode' parameter is set to 'Sample based.'
Main
Fixed-point
Settings on this pane only apply when block inputs are fixed-point signals.
Fixed-point operational parameters
Rounding mode:
Floor
Overflow mode:
Wrap
Fixed-point data types
Mode
Sine table
Same word length as input
Product output
Inherit via internal rule
Accumulator
Inherit via internal rule
Output
Inherit via internal rule
Lock scaling against changes by the autoscaling tool
OK
Cancel
Help
Apply

Rounding mode

Select the rounding mode for fixed-point operations. The sine table values do not obey this parameter; they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations.

Sine table

Choose how you specify the word length of the values of the sine table. The fraction length of the sine table values is always equal to the word length minus one:

- When you select Same word length as input, the word length of the sine table values match that of the input to the block.
- When you select Specify word length, you can enter the word length of the sine table values, in bits.

The sine table values do not obey the Rounding mode and Overflow mode parameters; they are always saturated and rounded to Nearest.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-638 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Inherit via internal rule, the product output word length and fraction length are calculated automatically. For information on about how the product output word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.

- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths. See “Fixed-Point Data Types” on page 2-638 and “Multiplication Data Types” for illustrations depicting the use of the accumulator data type in this block:

- When you select Inherit via internal rule, the accumulator word length and fraction length are calculated automatically. For information about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as product output, these characteristics match those of the product output.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Inherit via internal rule, the output word length and fraction length are calculated automatically. The internal rule first calculates an ideal output word length and fraction length using the following equations:

$$WL_{ideal\ output} = WL_{input} + floor(\log_2(DCT\ length - 1)) + 1$$

$$FL_{ideal\ output} = FL_{input}$$

Using these ideal results, the internal rule then selects word lengths and fraction lengths that are appropriate for your hardware. For more informatio n, see “Inherit via Internal Rule”.

- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any Þxed-point scaling you specify in this block mask f rom being overridden by the autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

DCT	Signal Processing Blockset
IFFT	Signal Processing Blockset
idct	Signal Processing Toolbox

Identity Matrix

Purpose

Generate matrix with ones on main diagonal and zeros elsewhere

Library

- Signal Processing Sources

  dspsrcs4

- Math Functions / Matrices and Linear Algebra / Matrix Operations

  dspmtrx3

Description

The Identity Matrix block generates a rectangular matrix with ones on the main diagonal and zeros elsewhere.

When you select the Inherit output port attributes from input port check box, the input port is enabled, and an M-by-N matrix input generates a sample-based M-by-N matrix output with the same sample period. The values in the input matrix are ignored.

```
y = eye([M N])          % Equivalent MATLAB code
```

When you do not select the Inherit output port attributes from input port check box, the input port is disabled, and the dimensions of the output matrix are determined by the Matrix size parameter. A scalar value, M, specifies an M-by-M identity matrix, while a two-element vector, [M N], specifies an M-by-N unit-diagonal matrix. The output is sample based, and has the sample period specified by the Sample time parameter.

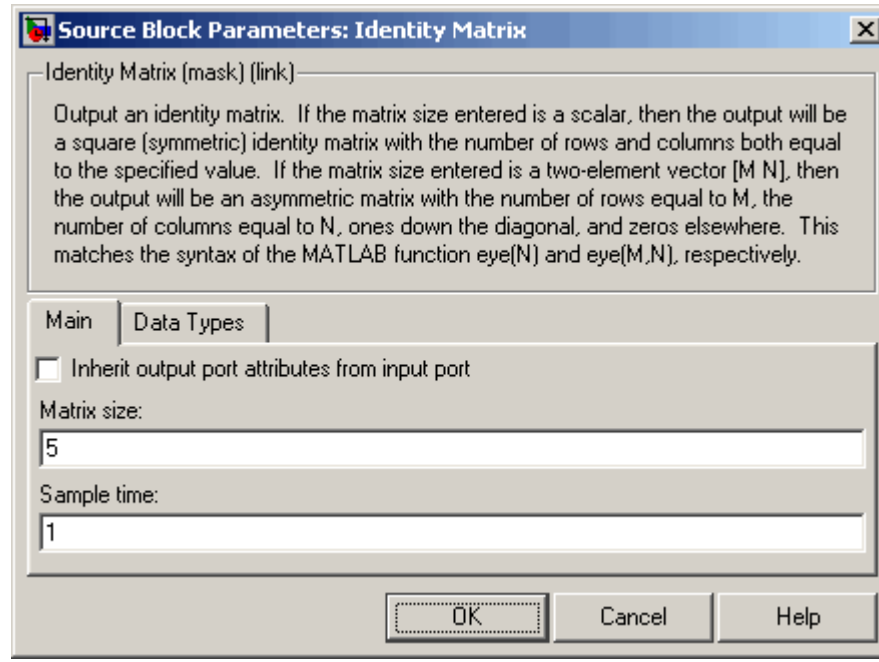
Examples

Set Matrix size to [3 6] to generate the 3-by-6 unit-diagonal matrix below.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Dialog Box

The Main pane of the Identity Matrix block dialog appears as follows.



Inherit output port attributes from input port

Enables the input port when selected. In this mode, the output inherits its dimensions, sample period, and data type from the input. The output is always real.

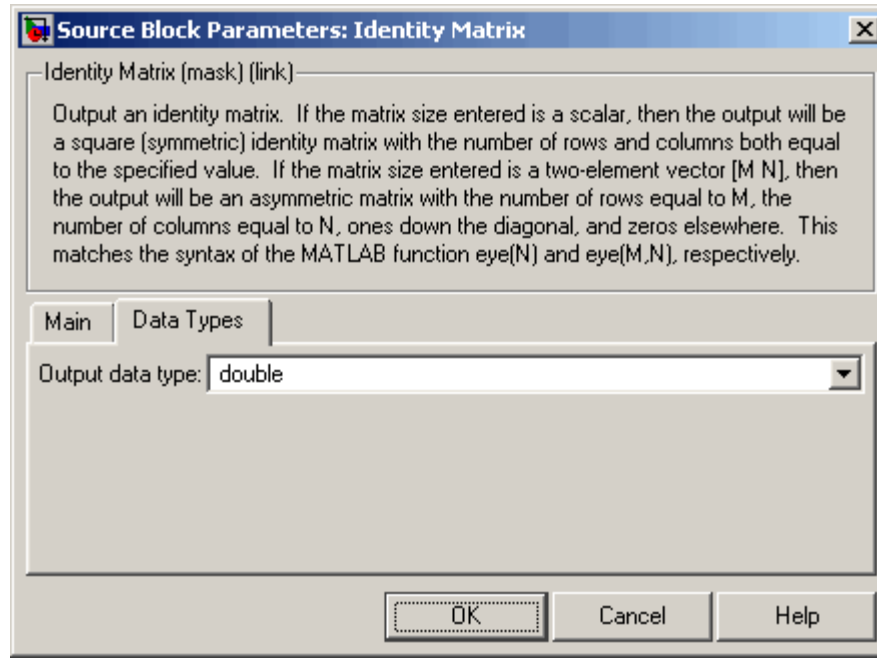
Matrix size

The number of rows and columns in the output matrix: a scalar M for a square M-by-M output, or a vector [M N] for an M-by-N output. This parameter is disabled when you select Inherit input port attributes from input port .

Sample time

The discrete sample period of the output. This parameter is disabled when you select Inherit input port attributes from input port .

The Data Types pane of the Identity Matrix block dialog appears as follows.



Output data type

Specify the output data type in one of the following ways:

- Choose one of the built-in data types from the list.
- Choose Fixed-point to specify the output data type and scaling in the Signed , Word length , Set fraction length in output to, and Fraction length parameters.

- Choose User-defined to specify the output data type and scaling in the User-defined data type, Set fraction length in output to, and Fraction length parameters.
- Choose Inherit via back propagation to set the output data type and scaling to match the following block

Signed

Select to output a signed fixed-point signal. Otherwise, the signal is unsigned. This parameter is visible only when you select Fixed-point for the Output data type parameter.

Word length

Specify the word length, in bits, of the fixed-point output data type. This parameter is visible only when you select Fixed-point for the Output data type parameter.

User-defined data type

Specify any built-in or fixed-point data type. You can specify fixed-point data types using the following Simulink Fixed Point functions: sfix, ufix, sint, uint, sfrac, and ufrac. This parameter is visible only when you select User-defined for the Output data type parameter.

Set fraction length in output to

Specify the scaling of the fixed-point output by either of the following two methods:

- Choose Best precision to have the output scaling automatically set such that the output signal has the best possible precision.
- Choose User-defined to specify the output scaling in the Fraction length parameter.

This parameter is visible only when you select Fixed-point for the Output data type parameter, or when you select User-defined and the specified output data type is a fixed-point data type.

Fraction length

For fixed-point output data types, specify the number of fractional bits, or bits to the right of the binary point. This parameter is visible only when you select Fixed-point or User-defined for the Output data type parameter and User-defined for the Set fraction length in output to parameter.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Constant Diagonal Matrix	Signal Processing Blockset
Constant	Simulink
eye	MATLAB

Purpose

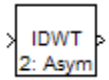
Compute inverse discrete wavelet transform (IDWT) of input

Library

Transforms

dspxfrm3

Description



Note The IDWT block is the same as the Dyadic Synthesis Filter Bank block in the Multirate Filters library, but with different default settings. See the Dyadic Synthesis F ilter Bank for more information on how to use the block.

The IDWT block computes the inverse discrete wavelet transform (IDWT) of the input subbands. By default, the block accepts a single sample-based vector or matrix of concatenated subbands. The output is frame based, and has the same dimensions as the input. Each column of the output is the IDWT of the corresponding input column.

You must install the Wavelet Toolbox product for the block to automatically design wavelet-based Þlters to compute the IDWT. Otherwise, you must specify yo ur own lowpass and highpass FIR Þlters by setting the Filter parameter to User defined .

For detailed information about how to use this block, see Dyadic Synthesis Filter Bank.

Examples

See the examples in the Dyadic Synthesis Filter Bank block reference.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Dyadic Synthesis Filter Bank

DWT

Signal Processing Blockset

Signal Processing Blockset

Purpose

Compute inverse fast Fourier transform (IFFT) of input

Library

Transforms

dspxfrm3

Description

The IFFT block computes the inverse fast Fourier transform (IFFT) of each row of a sample-based 1-by-P input vector, u, or across the first dimension (P) of an N-D input array, u. When you select the Inherit FFT length from input dimensions check box, P must be an integer power of two and the FFT length, M, gets set equal to P. If you do not select the check box, P can be any length, and the value of the FFT length parameter, M, must be a positive integer power of two. For user-specified FFT lengths, when M is not equal to P, zero padding or modulo-M data wrapping happens before the IFFT operation, as per Orfanidis [1].

```
y = ifft(u,M)   % P <= M

y(:,l) = ifft(datawrap(u(:,l),M))   % P > M; l = 1,...,N
```

When the input length, P, is greater than the FFT length, M, you may see magnitude increases in your IFFT output. These magnitude increases occur because the IFFT block uses modulo-M data wrapping to preserve all available input samples.

To avoid such magnitude increases, you can truncate the length of your input sample, P, to the FFT length, M. To do so, place a Pad block before the IFFT block in your model.

The kth entry of the lth output channel, y(k, l), is equal to the kth point of the M-point inverse discrete Fourier transform (IDFT) of the lth input channel:

$$y(k,l) = \frac{1}{M}\sum_{p=1}^{P} u(p,l)e^{j2\pi(p-1)(k-1)/M} \qquad k = 1,\ldots,M$$

This block supports real and complex ßoating-point and Þxed-point inputs.

Input and Output Characteristics

The following table describes the inpu t and output characteristics of the IFFT block. The table's columns pro vide the following information:

- Valid inputs to the IFFT block. They can be real- or complex-valued, and they can be in linear or bit-reversed order.
- The dimension along which the block computes the IDFT.
- The corresponding block output cha racteristics. The output port rate must equal the input port rate.

Valid Block Inputs	Dimension Along Which Block Computes IDFT	Corresponding Block Output Characteristics
N-D array	First dimension	The following output charac teristics apply to all valid block inputs: • Frame based • Complex valued. If you input conjugate symmetric data and select the Input is conjugate symmetric check box, the block outputs a real-valued result. • Same dimension as input (for 1-D inputs, output is a length- M column). • Each output column (each row for sample-based row inputs) contains the M-point IDFT of the corresponding input channel in linear order. If you do not select the Divide output by FFT length check box, the block computes a modi Þed version of
Sample-based 1-by-P row vector	Row	
1-D length- P vector	Vector	

Valid Block Inputs	Dimension Along Which Block Computes IDFT	Corresponding Block Output Characteristics
		the IDFT that does not include the multiplication factor of 1/M.

Selecting the Twiddle Factor Computation Method

The Twiddle factor computation parameter determines how the block computes the necessary sine and cosine terms to calculate the term $e^{j2\pi(p-1)(k-1)/M}$, shown in the first equation of this block reference page. This parameter has two settings, each with its advantages and disadvantages, as described in the following table. For fixed-point signals, only Table lookup mode is supported.

Twiddle Factor Computation Parameter Setting	Sine and Cosine Computation Method	Effect on Block Performance
Table lookup	The block computes and stores the trigonometric values before the simulation starts and retrieves them during the simulation. When you generate code from the block, the processor running the generated code stores the trigonometric values computed by the block and retrieves the values during code execution.	The block usually runs much more quickly, but requires extra memory for storing the precomputed trigonometric values. You can optimize the table for memory consumption or speed, as described in "Optimizing the Table of Trigonometric Values" on page 2-658.
Trigonometric fcn	The block computes sine and cosine values during the simulation. When you generate code from the	The block usually runs more slowly, but does not need extra data memory. For code generation,

Twiddle Factor Computation Parameter Setting	Sine and Cosine Computation Method	Effect on Block Performance
	block, the processor running the generated code computes the sine and cosine values while the code runs.	the block requires a support library to emulate the trigonometric functions, increasing the size of the generated code.

Optimizing the Table of Trigonometric Values

When you set the Twiddle factor computation parameter to Table lookup, you also need to set the Optimize table for parameter. This parameter optimizes the table of trigonometric values for speed or memory by varying the number of table entries as summarized in the following table.

Optimize Table for Parameter Setting	Number of Table Entries for N-Point IFFT	Memory Required for Single-Precision 512-Point IFFT
Speed	3N/4 — floating point N — fixed point	$\left(\frac{3\times 512}{4}\text{table entries}\right)\times\left(4\frac{\text{bytes}}{\text{table entry}}\right)=1536\text{ bytes}$
Memory	N/4 — floating point Not supported for fixed point	$\left(\frac{512}{4}\text{ table entries}\right)\times\left(4\frac{\text{bytes}}{\text{table entry}}\right)=512\text{ bytes}$

Input Order

Select or clear the Input is in bit-reversed order check box to designate the ordering of the column elements of the block input. If you select the Input is in bit-reversed order check box, the block assumes the input is in bit-reversed order. If you clear the Input is in bit-reversed order check box, the block assumes the input is in linear order. For more information on ordering of the output, see “Linear and Bit-Reversed Output Order”.

Conjugate Symmetric Input

The FFT block yields conjugate symmetric output when you input real-valued data. Taking the IFFT of a conjugate symmetric input matrix produces real-valued output. Therefore, if you input conjugate symmetric data and select the Input is conjugate symmetric check box, the block produces real-valued outputs. Selecting this check box optimizes the block’s computation method.

If you input conjugate symmetric data to the IFFT block and do not select the Input is conjugate symmetric check box, the IFFT block outputs a complex-valued signal with small imaginary parts. If you select this check box and the input is not conjugate symmetric, the block output is invalid.

Scaled Output

When you select the Divide output by FFT length check box, the block computes its output according to the IDFT equation, discussed in the Description section.

If you clear the Divide output by FFT length check box, the block computes a modified version of the IDFT, $M \cdot y(k,l)$, which is defined by the following equation:

$$M \cdot y(k,l) = \sum_{p=1}^{P} u(p,l) e^{j2\pi(p-1)(k-1)/M} \quad k = 1,\ldots,M$$

Algorithms Used for IFFT Computation

Depending on whether the block input is in bit-reversed order, conjugate symmetric order, or both, the block uses one or more of the following algorithms as summarized in the subsequent table:

- Bit-reversal operation
- Double-signal algorithm
- Half-length algorithm
- Radix-2 decimation-in-time (DIT) algorithm

Input Complexity	Other Parameter Settings	Algorithms Used for IFFT Computation
Real or complex	☐ Input is in bit-reversed order ☐ Input is conjugate symmetric	Bit-reversal operation and radix-2 DIT
Real or complex	☑ Input is in bit-reversed order ☐ Input is conjugate symmetric	Radix-2 DIF
Real or complex	☐ Input is in bit-reversed order ☑ Input is conjugate symmetric	Bit-reversal operation and radix-2 DIT in conjunction with the half-length and double-signal algorithms
Real or complex	☑ Input is in bit-reversed order ☑ Input is conjugate symmetric	Radix-2 DIF in conjunction with the half-length and double-signal algorithms

When the Input is conjugate symmetric check box is selected, the efficiency of the IFFT algorithm can be enhanced in two ways:

- By forming one length- M complex-valued sequence from two length- M conjugate symmetric input sequences
- By forming one length- M complex-valued sequence from one length-$2M$ conjugate symmetric input sequence

These optimizations correspond to the optimizations used in the FFT computation of real input signals.

When there are $2N+1$ conjugate symmetric input channels and the Input is conjugate symmetric check box is selected, the IFFT block applies the double-signal algorithm to the first $2N$ input channels and the half-length algorithm to the last odd-numbered channel. If the input signals have fixed-point data types, it is possible to see different numerical results in the output of the last odd channel, even if all input channels are identical. This numerical difference results from differences in the double-signal algorithm and the half-length algorithm.

You can eliminate this numerical difference in two ways:

- Using full precision arithmetic for fixed-point input signals
- Changing the input data type to floating point

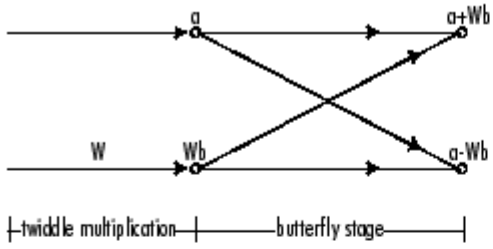
For more information on the double-signal and half-length algorithms, see Proakis [2]. "Efficient Computation of the DFT of Two Real Sequences" on page 475 describes the double signal algorithm. "Efficient Computation of the DFT of a 2N-Point Real Sequence" on page 476 describes the half-length algorithm.

Fixed-Point Data Types

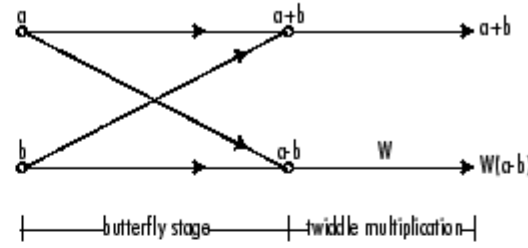
The following diagrams show the data types used within the IFFT block for fixed-point signals. You can set the sine table, accumulator, product output, and output data types displayed in the diagrams in the IFFT block dialog, as discussed in "Dialog Box" on page 2-664.

The IFFT block first casts input to the output data type and then stores it in the output buffer. Each butterfly stage then processes signals in the accumulator data type, with the final output of the butterfly being cast back into the output data type. The block multiplies in a twiddle factor before each butterfly stage in a decimation-in-time IFFT, and after each butterfly stage in a decimation-in-frequency IFFT.

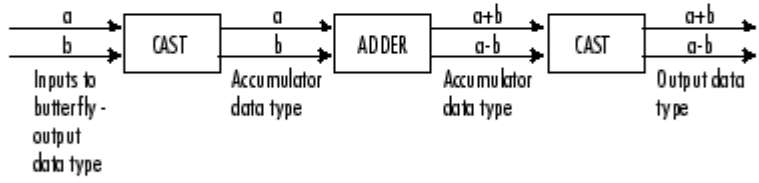
Decimation-in-time IFFT



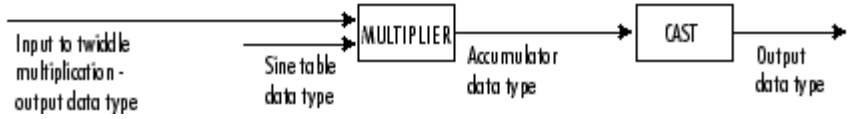
Decimation-in-frequency IFFT



Butterfly stage data types



Twiddle multiplication data types



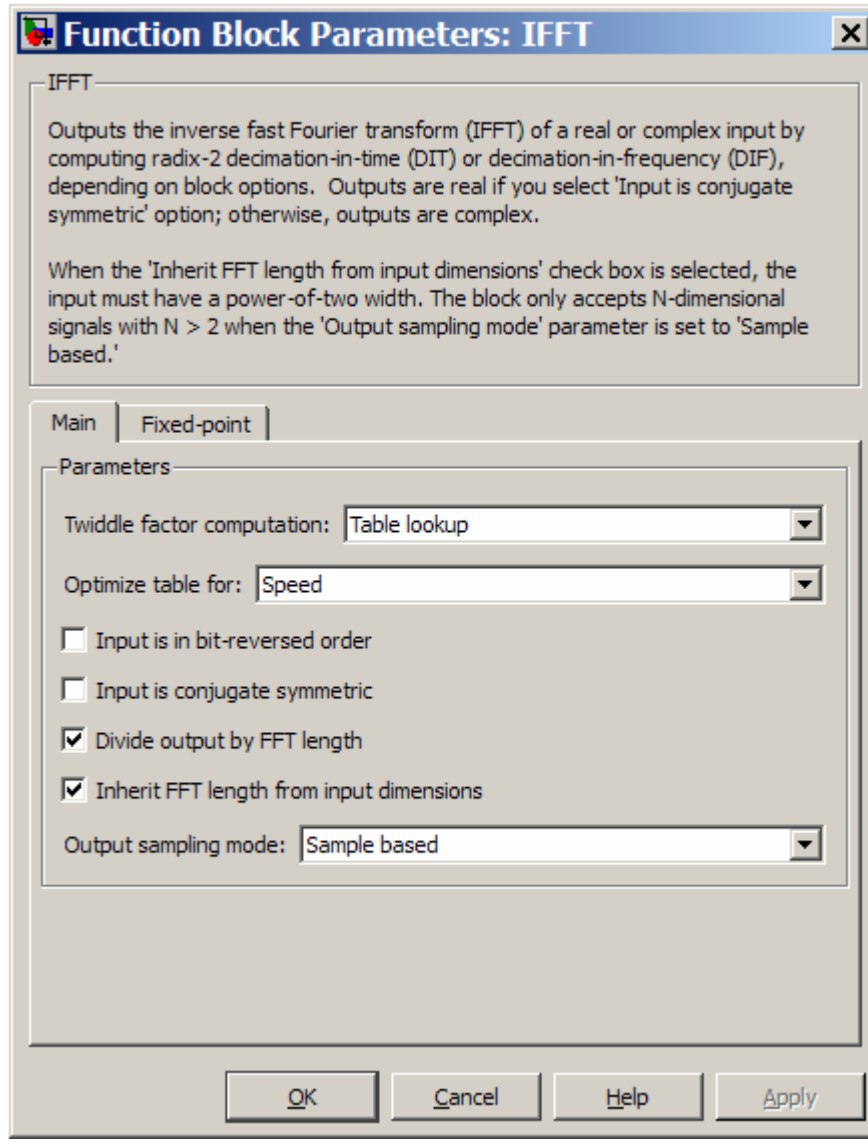
The output of the multiplier is in the accumulator data type because both of the inputs to the multiplier are complex. For details on the complex multiplication performed, see “Multiplication Data Types”.

Examples

See “Transforming Frequency-Do main Data into the Time Domain” in the Signal Processing Blockset User’s Guide .

Dialog Box

The Main pane of the IFFT block dialog appears as follows.



Twiddle factor computation

Specify the computation method of the term $e^{j2\pi(p-1)(l-1)/M}$ shown in the first equation of this block reference page.

In Table lookup mode, the block computes and stores the sine and cosine values before the simulation starts.

In Trigonometric fcn mode, the block computes the sine and cosine values during the simulation. See "Selecting the Twiddle Factor Computation Method" on page 2-657.

This parameter must be set to Table lookup for fixed-point signals.

Optimize table for

Select this option to optimize the table of sine and cosine values for either Speed or Memory. This parameter becomes available only when you set the Twiddle factor computation parameter to Table lookup. See "Optimizing the Table of Trigonometric Values" on page 2-658.

This parameter must be set to Speed for fixed-point signals.

Input is in bit-reversed order

Designate the order of the input channel elements. Select this check box when the input is in bit-reversed order, and clear this check box when the input is in linear order. The block yields invalid outputs when you do not set this parameter correctly. See "Input Order" on page 2-659.

You cannot select this parameter if you have cleared the Inherit FFT length from input dimensions parameter, and you are specifying the FFT length using the FFT length parameter.

Input is conjugate symmetric

Select this option when the input to the block is conjugate symmetric and you want real-valued outputs. If you select this

option and the input is not conjugate symmetric, the block output is invalid.

You cannot select this parameter if you have cleared the Inherit FFT length from input dimensions parameter, and you are specifying the FFT length using the FFT length parameter.

Divide output by FFT length

When you select this check box, the block computes the output according to the IDFT equation discussed in the Description section.

When you clear this check box, the block computes the output using a modified version of the IDFT. The modified IDFT equation does not include the multiplication factor of 1/M. For the full equation, see “Scaled Output” on page 2-659.

Inherit FFT length from input dimensions

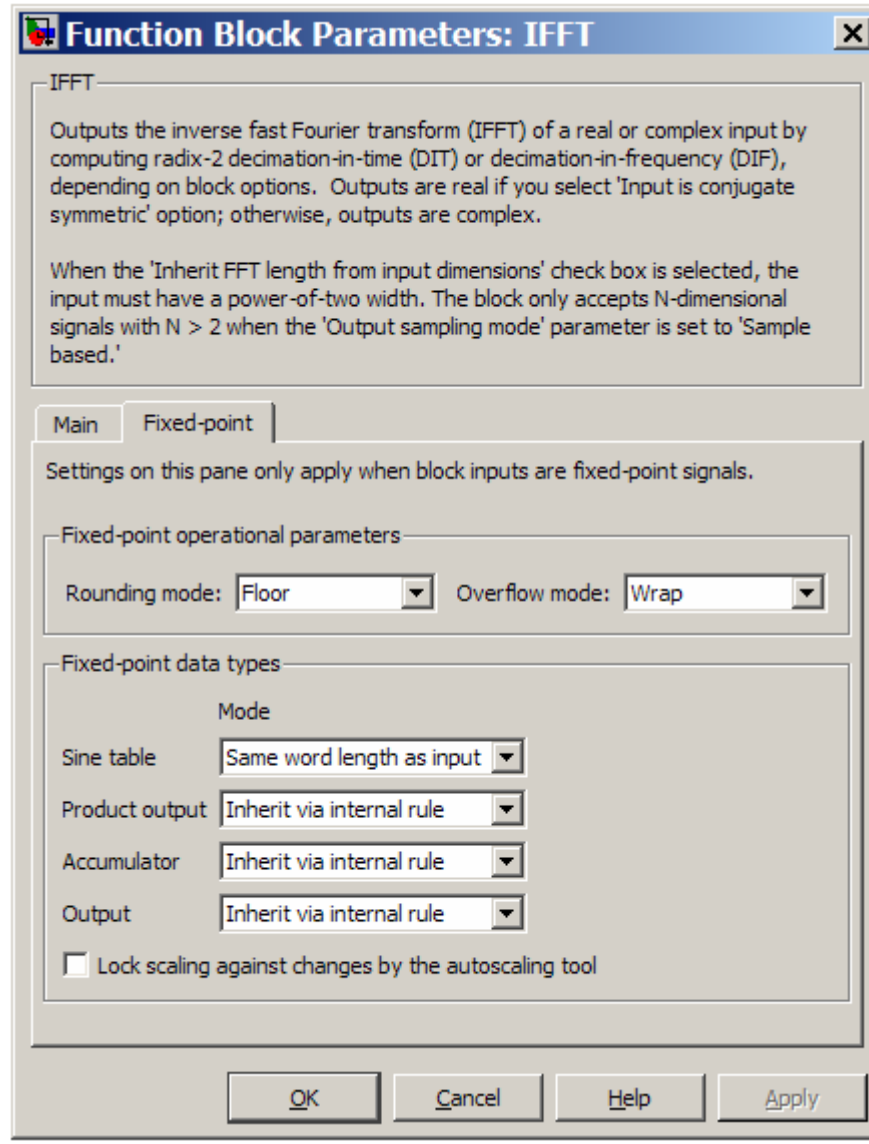
Select to inherit the FFT length from the input dimensions. When you select this parameter, the input length P must be a power of two. If you do not select this parameter, the FFT length parameter becomes available.

You cannot clear this parameter when you select either the Input is in bit-reversed order or the Input is conjugate symmetric parameter.

FFT length

Specify a power-of-two FFT length. This parameter only becomes available if you do not select the Inherit FFT length from input dimensions parameter.

The Fixed-point pane of the IFFT block dialog appears as follows.


Function Block Parameters: IFFT
IFFT
Outputs the inverse fast Fourier transform (IFFT) of a real or complex input by computing radix-2 decimation-in-time (DIT) or decimation-in-frequency (DIF), depending on block options. Outputs are real if you select 'Input is conjugate symmetric' option; otherwise, outputs are complex.
When the 'Inherit FFT length from input dimensions' check box is selected, the input must have a power-of-two width. The block only accepts N-dimensional signals with N > 2 when the 'Output sampling mode' parameter is set to 'Sample based.'
Main
Fixed-point
Settings on this pane only apply when block inputs are fixed-point signals.
Fixed-point operational parameters
Rounding mode: Floor
Overflow mode: Wrap
Fixed-point data types
Mode
Sine table
Same word length as input
Product output
Inherit via internal rule
Accumulator
Inherit via internal rule
Output
Inherit via internal rule
Lock scaling against changes by the autoscaling tool
OK
Cancel
Help
Apply

Rounding mode

Select the rounding mode for fixed-point operations. The sine table values do not obey this parameter; instead, they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations. The sine table values do not obey this parameter; instead, they are always saturated.

Sine table

Choose how you specify the word length of the values of the sine table. The fraction length of the sine table values always equals the word length minus one:

- When you select Same word length as input, the word length of the sine table values match that of the input to the block.
- When you select Specify word length, you can enter the word length of the sine table values, in bits.

The sine table values do not obey the Rounding mode and Overflow mode parameters; instead, they are always saturated and rounded to Nearest.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-661 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Inherit via internal rule, the product output word length and fraction length are calculated automatically. For information about how the block calculates the product output word and fraction lengths when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as input, these characteristics match those of the input to the block.

- When you select Binary point scaling , you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths. See “Fixed-Point Data Types” on page 2-661 and “Multiplication Data Types” for illustrations depicting the use of the accumulator data type in this block:

- When you select Inherit via internal rule , the block calculates the accumulator word length and fraction length automatically. For information about how block calculates the accumulator word and fraction lengths when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Inherit via internal rule , the block calculates the output word length and fraction length

automatically. The internal rule first calculates an ideal output word length and fraction length using the following equations:

$$WL_{ideal\ output} = WL_{input} + floor(\log_2(FFT\ length - 1)) + 1$$

$$FL_{ideal\ output} = FL_{input}$$

Using these ideal results, the internal rule then selects word lengths and fraction lengths that are appropriate for your hardware. For more information, see “Inherit via Internal Rule”.

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

References

[1] Orfanidis, S. J. Introduction to Signal Processing. Upper Saddle River, NJ: Prentice Hall, 1996, p. 497.

[2] Proakis, John G. and Dimitris G. Manolakis. Digital Signal Processing, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 1996.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

FFT	Signal Processing Blockset
IDCT	Signal Processing Blockset
Pad	Signal Processing Blockset
bitrevorder	Signal Processing Toolbox
fft	MATLAB
ifft	MATLAB

Inherit Complexity

Purpose

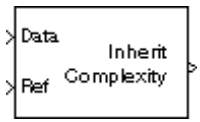
Change complexity of input to match reference signal

Library

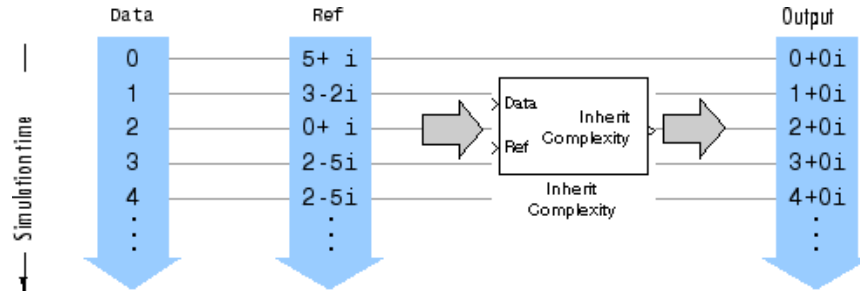
Signal Management / Signal Attributes

dspsigattribs

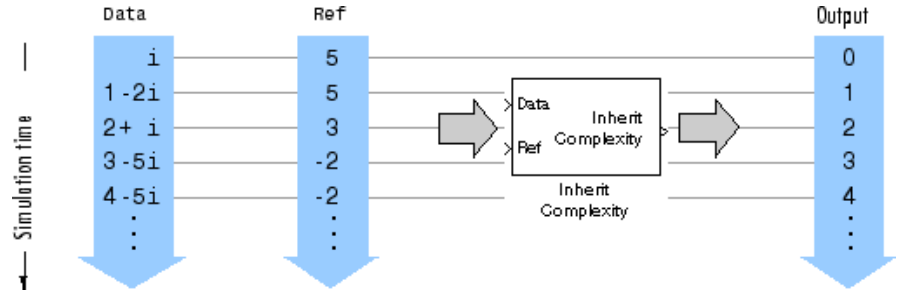
Description



The Inherit Complexity block alters the input data at the Data port to match the complexity of the referen ce input at the Ref port. When the Data input is real, and the Ref input is complex, the block appends a zero-valued imaginary component, 0i , to each element of the Data input.

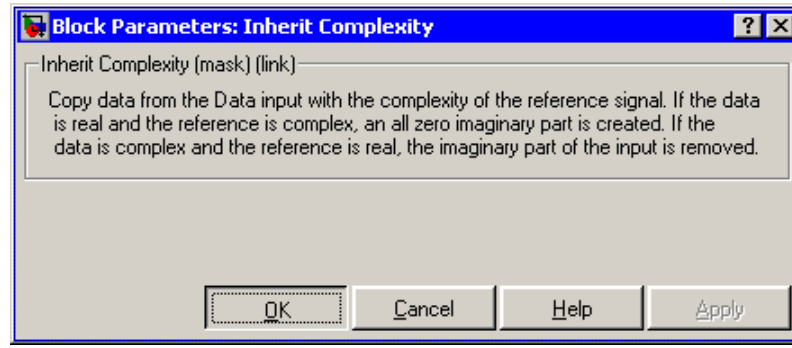


When the Data input is complex, and the Ref input is real, the block outputs the real component of the Data input.



When both the Data input and Ref input are real, or when both the Data input and Ref input are complex, the block propagates the Data input with no change.

Dialog Box



Supported Data Types

Port	Supported Data Types
Data	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Ref	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned)

Port	Supported Data Types
	• Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Check Signal Attributes	Signal Processing Blockset
Complex to Magnitude-Angle	Simulink
Complex to Real-Imag	Simulink
Magnitude-Angle to Complex	Simulink
Real-Imag to Complex	Simulink

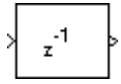
Purpose

Delay input by integer number of sample periods

Library

dspobslib

Description



Note The Integer Delay block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the Delay block.

The Integer Delay block delays a discrete-time input by the number of sample intervals specified in the Delay parameter. Noninteger delay values are rounded to the nearest integer, and negative delays are clipped at 0.

Sample-Based Operation

When the input is a sample-based M-by-N matrix, the block treats each of the M*N matrix elements as an independent channel. The Delay parameter, v, can be an M-by-N matrix of positive integers that specifies the number of sample intervals to delay each channel of the input, or a scalar integer by which to equally delay all channels.

For example, when the input is M-by-1 and v is the matrix [v(1) v(2) ... v(M)]', the first channel is delayed by v(1) sample intervals, the second channel is delayed by v(2) sample intervals, and so on. Note that when a channel is delayed for Δ sample-time units, the output sample at time t is the input sample at time $t - \Delta$. When $t - \Delta$ is negative, then the output is the corresponding value specified by the Initial conditions parameter.

A 1-D vector of length M is treated as an M-by-1 matrix, and the output is 1-D.

The Initial conditions parameter specifies the output of the block during the initial delay in each channel. The initial delay for a particular channel is the time elapsed from the start of the simulation until the first input in that channel is propagated to the output. Both

Fixed and time-varying initial conditions can be specified in a variety of ways to suit the dimensions of the input.

Fixed Initial Conditions

A fixed initial condition in sample-based mode can be specified as one of the following:

- Scalar value to be repeated at each sample time of the initial delay (for every channel). For a 2-by-2 input with the parameter settings below,

  the block generates the following sequence of matrices at the start of the simulation,

  $$\begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}, \begin{bmatrix} u_{11}^{1} & -1 \\ -1 & -1 \end{bmatrix}, \begin{bmatrix} u_{11}^{2} & u_{12}^{1} \\ -1 & -1 \end{bmatrix}, \begin{bmatrix} u_{11}^{3} & u_{12}^{2} \\ u_{21}^{1} & -1 \end{bmatrix}, \begin{bmatrix} u_{11}^{4} & u_{12}^{3} \\ u_{21}^{2} & u_{22}^{1} \end{bmatrix}, \ldots$$

  where u_{ij}^{k} is the i,jth element of the kth matrix in the input sequence.

- Array of size M-by-N-by-d. In this case, you can set different fixed initial conditions for each element of a sample-based input. This setting is explained further in the Array bullet in “Time-Varying Initial Conditions” on page 2-676.

Initial conditions cannot be specified by full matrices.

Time-Varying Initial Conditions

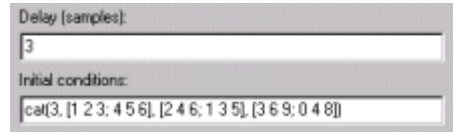
A time-varying initial condition in sample-based mode can be specified in one of the following ways:

- Vector of length d, where d is the maximum value specified for any channel in the Delay parameter. The vector can be a L-by-d, 1-by-d, or 1-by-1-by-d. The d elements of the vector are output in sequence, one at each sample time of the initial delay.

  For a scalar input and the parameters shown below,

  the block outputs the sequence -1, -1, -1, 0, 1,... at the start of the simulation.

- Array of dimension M-by-N-by-d, where d is the value specified for the Delay parameter (the maximum value when the Delay is a vector) and M and N are the number of rows and columns, respectively, in the input matrix. The d pages of the array are output in sequence, one at each sample time of the initial delay. For a 2-by-3 input, and the parameters below,

  the block outputs the matrix sequence

  $$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \begin{bmatrix} 2 & 4 & 6 \\ 1 & 3 & 5 \end{bmatrix}, \begin{bmatrix} 3 & 6 & 9 \\ 0 & 4 & 8 \end{bmatrix}$$

  at the start of the simulation. Note that setting Initial conditions to an array with the same matrix for each entry implements constant initial conditions; a different constant initial condition for each input matrix element (channel).

Initial conditions cannot be specified by full matrices.

Frame-Based Operation

When the input is a frame-based M-by-N matrix, the block treats each of the N columns as an independent channel, and delays each channel as specified by the Delay parameter.

For frame-based inputs, the Delay parameter can be a scalar integer by which to equally delay all channels. It can also be a 1-by-N row vector, each element of which serves as the delay for the corresponding channel of the N-channel input. Likewise, it can also be an M-by-1 column vector, each element of which serves as the delay for one of the corresponding M samples for each channel. The Delay parameter can be an M-by-N matrix of positive integers as well; in this case, each element of each channel is delayed by the corresponding element in the delay matrix. For instance, if the fifth element of the third column of the delay matrix was 3, then the fifth element of the third channel of the input matrix is always delayed by three sample-time units.

When a channel is delayed for Δ sample-time units, the output sample at time t is the input sample at time $t - \Delta$. When $t - \Delta$ is negative, then the output is the corresponding value specified in the Initial conditions parameter.

The Initial conditions parameter specifies the output during the initial delay. Both fixed and time-varying initial conditions can be specified. The initial delay for a particular channel is the time elapsed from the start of the simulation until the first input in that channel is propagated to the output.

Fixed Initial Conditions

The settings shown below specify fixed initial conditions. The value entered in the Initial conditions parameter is repeated at the output for each sample time of the initial delay. A fixed initial condition in frame-based mode can be one of the following:

- Scalar value to be repeated for all channels of the output at each sample time of the initial delay. For a general M-by-N input with the parameter settings below,

the first five samples in each of the N channels are zero. Notice that when the frame size is larger than the delay, all of these zeros are all included in the first output from the block.

- Array of size 1-by- N-by-D. In this case, you can also specify different fixed initial conditions for each channel. See "Time-Varying Initial Conditions" on page 2-679 for details.

Initial conditions cannot be specified by full matrices.

Time-Varying Initial Conditions

The following settings specify time-varying initial conditions. For time-varying initial conditions, the values specified in the Initial conditions parameter are output in sequence during the initial delay. A time-varying initial condition in frame-based mode can be specified in the following ways:

- Vector of length D, where each of the N channels have the same initial conditions sequence specified in the vector. D is defined as follows:
  - When an element of the delay entry is less than the frame size,

    $D = d + 1$

    where d is the maximum delay.
  - When the all elements of the delay entry are greater than the input frame size,

    $D = d + \text{input frame size} - 1$

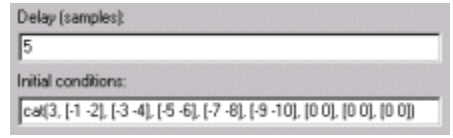
  Only the first d entries of the initial condition vector are used; the rest of the values are ignored, but you must include them nonetheless. For a two-channel ramp input [1:100; 1:100]' with a frame size of 4 and the parameter settings below,

the block outputs the following sequence of frames at the start of the simulation.

$$\begin{bmatrix} -4 & -1 \\ -5 & -2 \\ 1 & -3 \\ 2 & -4 \end{bmatrix}, \begin{bmatrix} 3 & -5 \\ 4 & 1 \\ 5 & 2 \\ 6 & 3 \end{bmatrix}, \begin{bmatrix} 7 & 4 \\ 8 & 5 \\ 9 & 6 \\ 10 & 7 \end{bmatrix}, \ldots$$

Note that since one of the delays, 2, is less than the frame size of the input, 4, the length of the Initial conditions vector is the sum of the maximum delay and 1 (5+1), which is 6. The first five entries of the initial conditions vector are used by the channel with the maximum delay, and the rest of the entries are ignored. Since the first channel is delayed for less than the maximum delay (2 sample time units), it only makes use of two of the initial condition entries.

- Array of size 1-by-N-by-D, where D is defined in "Time-Varying Initial Conditions" on page 2-679. In this case, the kth entry of each 1-by-N entry in the array corresponds to an initial condition for the kth channel of the input matrix. Thus, a 1-by-N-by-D initial conditions input allows you to specify different initial conditions for each channel. For instance, for a two-channel ramp input [1:100; 1:100]' with a frame size of 4 and the parameter settings below,



the block outputs the following sequence of frames at the start of the simulation.

$$\begin{bmatrix} -1 & -2 \\ -3 & -4 \\ -5 & -6 \\ -7 & -8 \end{bmatrix}, \begin{bmatrix} -9 & -10 \\ 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 4 \\ 5 & 5 \\ 6 & 6 \\ 7 & 7 \end{bmatrix}, \ldots$$

Note that the channels have distinct time varying initial conditions; the initial conditions for channel 1 correspond to the first entry of each length-2 row vector in the initial conditions array, and the initial conditions for channel 2 correspond to the second entry of each row vector in the initial conditions array. Only the first five entries in the initial conditions array are used; the rest are ignored.

The 1-by-N-by-D array entry can also specify different fixed initial conditions for every channel; in this case, every 1-by-N entry in the array would be identical, so that the initial conditions for each column are fixed over time.

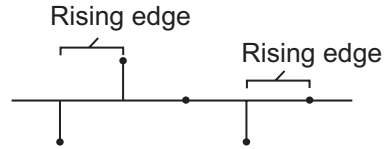
Initial conditions cannot be specified by full matrices.

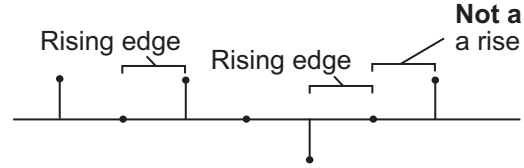
Resetting the Delay

The block resets the delay whenever it detects a reset event at the optional Rst port. The reset sample time must be a positive integer multiple of the input sample time.

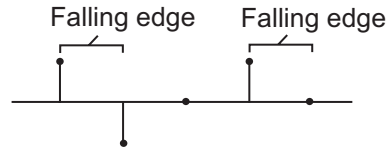
You specify the reset event in the Reset port parameter:

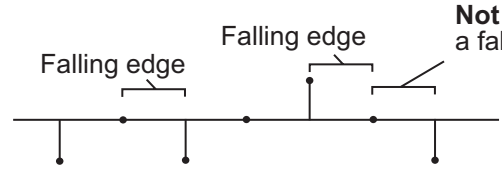
- None disables the Rst port.
- Rising edge — Triggers a reset operation when the Rst input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following figure)





- Falling edge — Triggers a reset operation when the Rst input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)





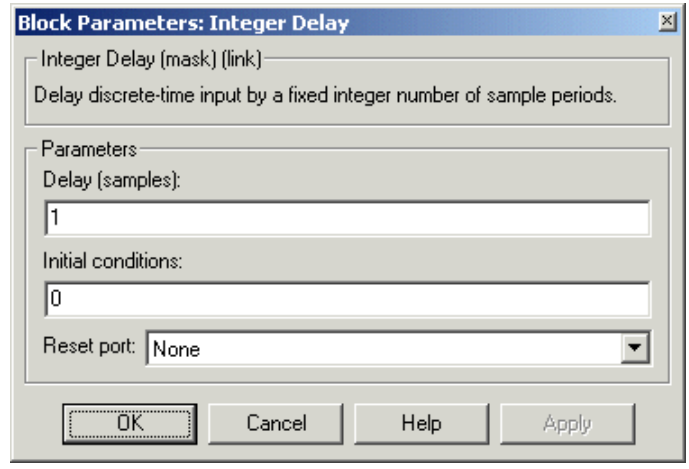
- Either edge — Triggers a reset o peration when the Rst input is a Rising edge or Falling edge (as described above).
- Non-zero sample — Triggers a reset opera tion at each sample time that the Rst input is not zero.

Note When running simulations in the Simulink MultiTasking mode, sample-based reset signals have a one-sample latency, and frame-based reset signals have one frame of latency. Thus, there is a one-sample or one-frame delay between the time the block detects a reset event, and when it applies the reset. For more information on latency and the Simulink tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Mo dels with Multiple Sample Rates” in the Real-Time Workshop User’s Guide .

Examples

The dspafxr demo illustrates an audio reverberation system built around the Integer Delay block.

Dialog Box



Delay

The numb er of sample periods to delay the input signal.

Initial conditions

The valu e of the block’s output during the initial delay.

Reset port

Determines the reset event that causes the block to reset the delay. For more information, see “Resetting the Delay” on page 2-681.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Fixed point (signed only)
- Boolean — The block accepts Boolean inputs to the Rst port, which is enabled by the Reset port parameter.
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

See Also

Unit Delay	Simulink
Variable Fractional Delay	Signal Processing Blockset
Variable Integer Delay	Signal Processing Blockset

Purpose

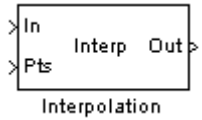
Interpolate values of real input samples

Library

Signal Operations

dspsigops

Description



Interpolation

The Interpolation block interpolates discrete, real, inputs using linear or FIR interpolation. The block acce pts both sample- and frame-based input data in the form of a vector, matrix, or sample-based N-D array. The block outputs a scalar, vector, matrix, or N-D array of the interpolated values, which has the same frame status as the input data.

You must specify the interpolation po ints (times at which to interpolate values) in a one-based interpolation array, I_{Pts}. An entry of 1 in I_{Pts} refers to the Þrst sample of the input data, an entry of 2.5 refers to the sample half-way between the second and third input sample, and so on. Depending on the dimensions of the input data, I_{Pts} can be a scalar, a length- P row or column vector, a P-by-N frame-based matrix, or a sample-based N-D array where P is the size of the Þrst dimension of the N-D array. In most cases, P can be any positive integer. For more information about valid interpolation arrays, refer to the tables in “How the Block Applies Interpolation Arrays to Inputs” on page 2-686.

In most cases, the block applies I_{Pts} across the Þrst dimension of an N-D input array, or to each input vector. You can set the block to apply the same interpolation array for all input data (static interpolation points entered on the block mask) or to use a different interpolation array for each N-D array, matrix, or vector inp ut (time-varying interpolation points received via the Pts input port).

Sections of This Reference Page

- “Specifying Static Interpo lation Points” on page 2-686
- “Specifying Time-Varying Interpolation Points” on page 2-686
- “How the Block Applies Interpolati on Arrays to Inputs” on page 2-686
- “Handling Out-of-Range Inter polation Points” on page 2-692
- “Linear Interpolation Mode” on page 2-693

- “FIR Interpolation Mode” on page 2-694
- “Dialog Box” on page 2-695
- “Supported Data Types” on page 2-697

Specifying Static Interpolation Points

To supply the block with a static interpolation array (an interpolation array applied to every vector or N-D array of input data), perform the following steps:

- Set the Source of interpolation points parameter to Specify via dialog.
- Enter the interpolation array in the Interpolation points parameter. To learn about interpolation arrays, see “How the Block Applies Interpolation Arrays to Inputs” on page 2-686.

Specifying Time-Varying Interpolation Points

To supply the block with time-varying interpolation arrays (where the block uses a different interpolation array for each vector or N-D array input), perform the following steps:

1. Set the Source of interpolation points parameter to Input port, the Pts port appears on the block.
2. Generate a signal of interpolation arrays, and supply it to the Pts port. The block uses the input to this port as the interpolation points. To learn about interpolation arrays, see “How the Block Applies Interpolation Arrays to Inputs” on page 2-686.

How the Block Applies Interpolation Arrays to Inputs

The interpolation array I_{Pts} represents the points in time at which to interpolate values of the input signal. An entry of 1 in I_{Pts} refers to the first sample of the input, an entry of 2.5 refers to the sample half-way between the second and third input sample, and so on. In most cases, when I_{Pts} is a vector, it can be of any length.

Valid values in the interpolation array, I_{Pts}, range from 1 to the number of samples in each channel of the input. To learn how the block handles out of range interpolation values, see “Handling Out-of-Range Interpolation Points” on page 2-692.

Depending on the dimension and frame status of the input and the dimension of I_{Pts}, the block usually applies I_{Pts} to the input in one of the following ways:

- Applies the I_{Pts} array across the first dimension of a sample-based N-D array or frame-based matrix input, resulting in a sample-based N-D array or frame-based matrix output.
- Applies the vector I_{Pts} to each input vector (as if the input vector were a single channel), resulting in a vector output with the same orientation as the input (row or column).

The following tables summarize how the block applies the interpolation array I_{Pts} to all the possible types of sample- and frame-based inputs, and show the resulting output dimensions.

The first table describes the block’s behavior when the Source of interpolation points is Specify via dialog and the input is sample based.

Input Dimensions (Sample Based)	Valid Dimensions of Interpolation Array I_{Pts}	How Block Applies I_{Pts} to Input	Output Dimensions (Sample Based)
N-D Array (ex. M-by-N-by-Q)	1-by-P row	Applies I_{Pts} to the first dimension of the input array	P-by-N-by-Q array
	P-by-1 column		
	P-by-N-by-Q array	Applies the columns of I_{Pts} to the corresponding	

Input Dimensions (Sample Based)	Valid Dimensions of Interpolation Array I_{Pts}	How Block Applies I_{Pts} to Input	Output Dimensions (Sample Based)
		columns of the input array	
M-by-1 column	1-by-P row (block treats I_n as a column)	Applies I_{Pts} to the input column	P-by-1 column
	P-by-1 column		
1-by-N row	1-by-P row	Applies I_n to the input row	1-by-P row
	P-by-1 column (block treats I_{Pts} as a row)		

The next table describes the block's behavior when the Source of interpolation points is Specify via dialog and the input is frame based.

Input Dimensions (Frame Based)	Valid Dimensions of Interpolation Array I_{Pts}	How Block Applies I_{Pts} to Input	Output Dimensions (Frame Based)
M-by-N matrix	1-by-N row	Applies each column of I_{Pts} (each element of I_{Pts}) to the corresponding column of the input matrix	1-by-N row

Input Dimensions (Frame Based)	Valid Dimensions of Interpolation Array I_{Pts}	How Block Applies I_{Pts} to Input	Output Dimensions (Frame Based)
	P-by-1 column	Applies I_{Pts} to each input column	P-by-N matrix
	P-by-N matrix	Applies the columns of I_{Pts} to the corresponding columns of the input matrix	
M-by-1 column	1-by-P row (block treats I_{Pts} as a column)	Applies I_{Pts} to the input column	P-by-1 column
	P-by-1 column		
1-by-N row (not recommended)	1-by-N row	Not Applicable. Block copies input vector	1-by-N row, a copy of the input vector
	P-by-1 column		P-by-N matrix where each row is a copy of the input vector
	P-by-N matrix		

The next table describes the block's behavior when the Source of interpolation points is Input port and the input is sample based.

Input Dimensions (Sample Based)	Valid Dimensions of Interpolation Array I_{Pts}	How Block Applies I_{Pts} to Input	Output Dimensions (Sample Based)
N-D Array (ex. M-by-N-by-Q)	Sample-based 1-by-P row	Applies I_{Pts} to the first dimension of the input array	P-by-N-by-Q array
	Sample- or frame-based P-by-1 column		
	Sample-based P-by-N-by-Q array	Applies the columns of I_{Pts} to the corresponding columns of the input array	
M-by-1 column	Sample-based 1-by-P row	Applies I_{Pts} to the input column	P-by-1 column
	Sample- or frame-based P-by-1 column		
1-by-N row	Sample-based 1-by-P row	Applies I_{Pts} to the input row	1-by-P row
	Sample or frame-based P-by-1 column		

The next table describes the block's behavior when the Source of interpolation points is Input port and the input is frame based.

Input Dimensions (Frame Based)	Valid Dimensions of Interpolation Array I_{Pts}	How Block Applies I_{Pts} to Input	Output Dimensions (Frame Based)
M-by-N matrix	Frame-based 1-by-N row	Applies each column of I_{Pts} (each element of I_{Pts}) to the corresponding column of the input matrix	1-by-N row
	Sample-based 1-by-P row	Applies I_{Pts} to each input column	P-by-N matrix
	Frame- or sample-based P-by-1 column		
	Frame- or sample-based P-by-N matrix	Applies the columns of I_{Pts} to the corresponding columns of the input matrix	
M-by-1 column	Sample-based 1-by-P row	Applies I_{Pts} to the input column	P-by-1 column
	Frame- or sample-based P-by-1 column		

Input Dimensions (Frame Based)	Valid Dimensions of Interpolation Array I_{Pts}	How Block Applies I_{Pts} to Input	Output Dimensions (Frame Based)
1-by-N row (not recommended)	Frame-based 1-by-N row	Not Applicable. Block copies input vector	1-by-N row, a copy of the input vector
	Sample-based 1-by-P row		P-by-N matrix where each row is a copy of the input vector
	Frame- or sample-based P-by-1 column		
	Frame- or sample-based P-by-N matrix		

Handling Out-of-Range Interpolation Points

Valid values in the interpolation array I_{Pts} range from 1 to the number of samples in each channel of the input. For instance, given a length-5 input vector D, all entries of I_{Pts} must range from 1 to 5. I_{Pts} cannot contain entries such as 7 or -9, since there is no 7th or -9th entry in D.

The Out of range interpolation points parameter sets how the block handles interpolation points that are fall outside the valid range, and has the following settings:

- Clip — The block replaces any out-of-range values in I_{Pts} with the closest value in the valid range (from 1 to the number of input samples), and then proceeds with computations using the clipped version of I_{Pts}.
- Clip and warn — In addition to Clip, the block issues a warning at the MATLAB command line every time clipping occurs.

- Error — When the block encounters an out-of-range value in I_{Pts}, the simulation stops, and the block issues an error at the MATLAB command line.

Example of Clipping

Suppose the block is set to clip out-of-range interpolation points, and gets the following input vector and interpolation points:

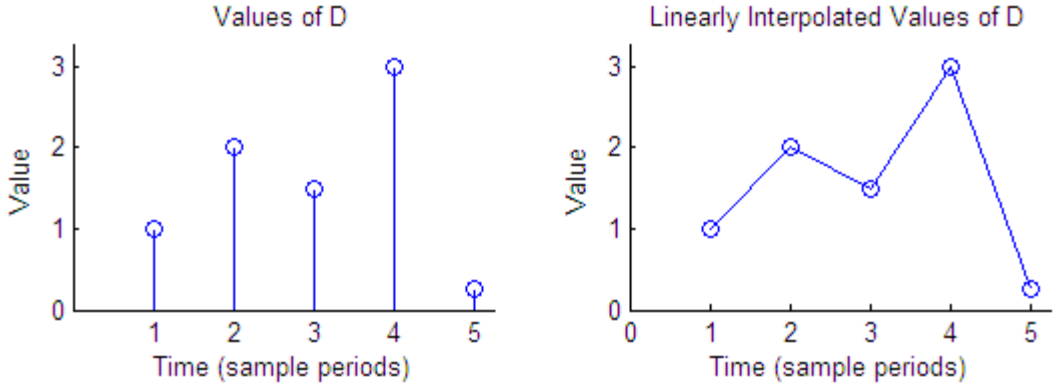
- D = [11 22 33 44]'
- I_{Pts} = [10 2.6 -3]'

Because D has four samples, valid interpolation points range from 1 to 4. The block clips the interpolation point 10 to 4 and the point -3 to 1, resulting in the clipped interpolation vector $I_{PtsClipped}$ = [4 2.6 1]'.

Linear Interpolation Mode

When Interpolation Mode is set to Linear, the block interpolates data values by assuming that the data varies linearly between samples taken at adjacent sample times.

For instance, if the input signal D = [1 2 1.5 3 0.25]', the following plot on the left shows the samples in D, and the plot on the right shows the linearly interpolated values between the samples in D.

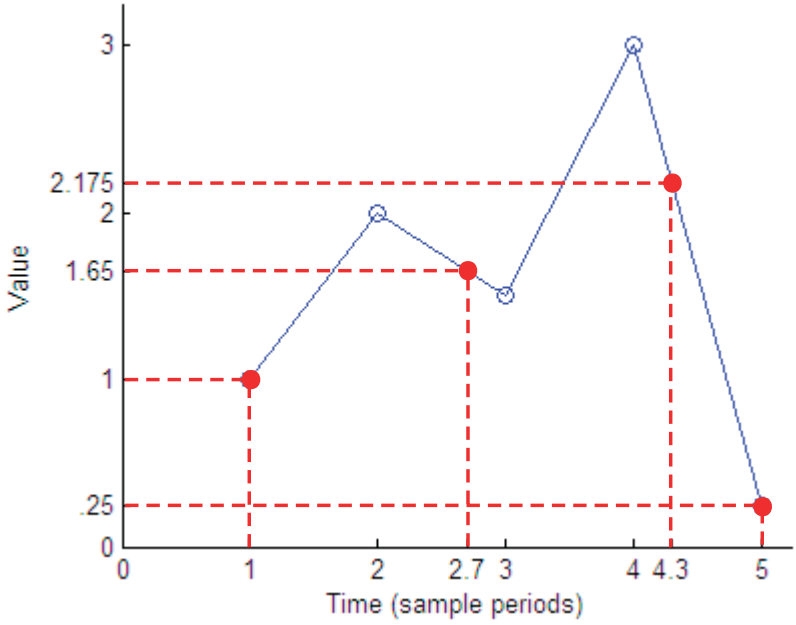


The following figure illustrates the case of a block in linear interpolation mode that is set to clip out-of-range interpolation points. The vector D supplies the input data and the vector I_{Pts} supplies the interpolation points:

- D = [1 2 1.5 3 0.25]'
- I_{Pts} = [-4 2.7 4.3 10]'

The block clips the invalid interpolation points, and outputs the linearly interpolated values in a vector, [1 1.65 2.175 0.25]'.

Interpolated Values of D at Clipped Interpolation Points



D = [1 2 1.5 3 0.25]'
I_{pts} = [-4 2.7 4.3 10]'

Valid interpolation points range from 1 to 5, so -4 is clipped to 1 and 10 is clipped to 5.

Clipped I_{pts} = [1 2.7 4.3 5]'

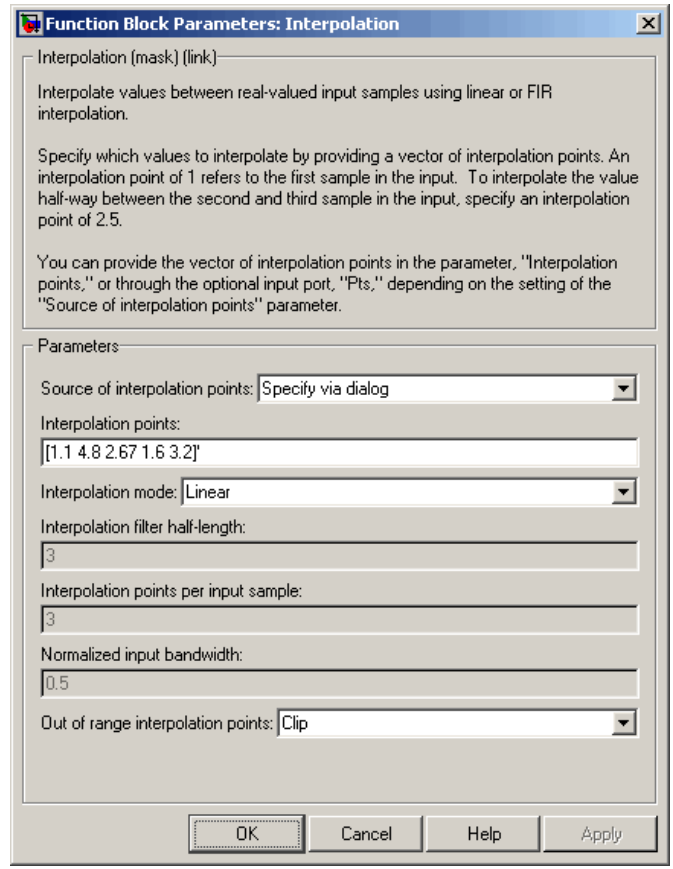
Output = [1 1.65 2.175 0.25]'

FIR Interpolation Mode

When Interpolation Mode is set to FIR, the block interpolates data values using an FIR interpolation filter, specified by various block

parameters. See FIR Interpolati on Mode on page 1350 in the Variable Fractional Delay block refe rence for more information.

Dialog Box



Source of interpolation points

Choose how you want to specify the interpolation points. If you select Specify via dialog , the Interpolation points parameter become available . Use this option for static interpolation points. If you select Input port , the Pts port

appears on the block. The block uses the input to this port as the interpolation points. Use this option for time-varying interpolation points. For more information, see “Specifying Static Interpolation Points” on page 2-686 and “Specifying Time-Varying Interpolation Points” on page 2-686.

Interpolation points

The array, I_{Pts}, of points in time at which to interpolate the input signal. An entry of 1 in I_{Pts} refers to the first sample of the input, an entry of 2.5 refers to the sample half-way between the second and third input sample, and so on. See “How the Block Applies Interpolation Arrays to Inputs” on page 2-686. Tunable.

Interpolation mode

Sets the block to interpolate by either linear or FIR interpolation. For more information, see “Linear Interpolation Mode” on page 2-693 and “FIR Interpolation Mode” on page 2-694.

Interpolator filter half-length

Half the length of the FIR interpolation filter. For more information, see “FIR Interpolation Mode” on page 2-694.

Interpolation points per input sample

The number Q, where the FIR interpolation filter uses the nearest 2*Q points in the signal to interpolate the value at an interpolation point. When there are less than 2*Q neighboring points, the block uses linear interpolation in place of FIR interpolation. For more information, see “FIR Interpolation Mode” on page 2-694. and “Linear Interpolation Mode” on page 2-693.

Normalized input bandwidth

The bandwidth of the input divided by Fs/2 (half the input sample frequency). For more information, see “FIR Interpolation Mode” on page 2-694.

Out of range interpolation points

When an interpolation point is out of range, this parameter sets the block to either clip the interpolation point, clip the value and issue a warning at the MATLAB command line, or stop the simulation and issue an error at the MATLAB command line.

For more information, see “Handling Out-of-Range Interpolation Points” on page 2-692.

Supported Data Types

Port	Supported Data Types
In	• Double-precision ßoating point • Single-precision ßoating point
Pts	• Double-precision ßoating point • Single-precision ßoating point
Out	• Double-precision ßoating point • Single-precision ßoating point

Purpose

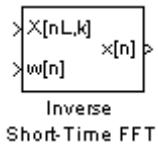
Recover time-domain signals by performing inverse short-time, fast Fourier transform (FFT)

Library

Transforms

dspxfrm3

Description



The Inverse Short-Time FFT block reconstructs the time-domain signal from the frequency-domain output of the Short-Time FFT block using a two-step process. First, the block performs the overlap add algorithm shown below.

$$x[n] = \frac{L}{W(0)} \sum_{p=-\infty}^{\infty} \left[\frac{1}{N} \sum_{k=0}^{N-1} X[pL,k] e^{j2\pi kn/N} \right]$$

Then, the block rebuffers the signal in order to reconstruct the frame-based time-domain signal. Depending on the analysis window used by the Short-Time FFT block, the Inverse Short-Time FFT block might or might not achieve perfect reconstruction of the time domain signal.

Connect your complex-valued, sample-based, single-channel or multichannel input signal to the X(n,k) port. The block uses the Overlap between consecutive STFFT frames (in samples) and Samples per output frame parameters as well as the Input is conjugate symmetric check box to reconstruct the original time-domain signal. The real or complex-valued, frame-based, single-channel or multichannel inverse short-time FFT is output at port x(n).

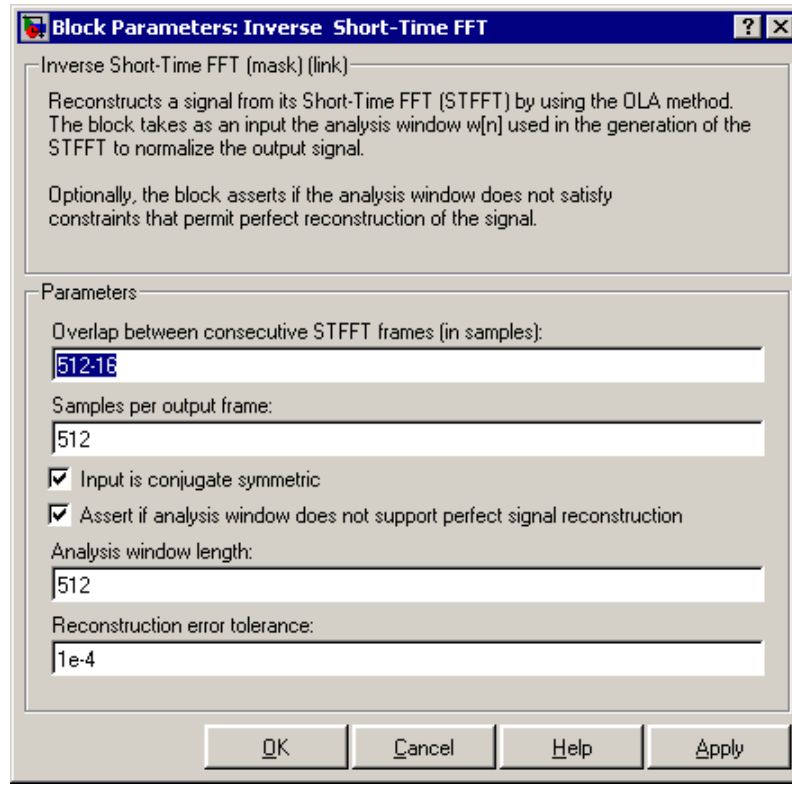
Connect your complex-valued, sample-based or frame-based, single-channel analysis window to the w(n) port. When you select the Assert if analysis window does not support perfect signal reconstruction check box, the block displays an error when the input signal cannot be perfectly reconstructed. The block uses the values you enter for the Analysis window length (W) and Reconstruction error tolerance, or maximum amount of allowable error in the

reconstruction process, to determine if the signal can be perfectly reconstructed.

Examples

The dspstsa demo illustrates how to use the Short-Time FFT and Inverse Short-Time FFT blocks to remove the background noise from a speech signal.

Dialog Box



Overlap between consecutive STFFT frames (in samples)

Enter the number of samples of overlap for each frame of the Short-Time FFT block's input signal. This value should be the same as the Overlap between consecutive windows (in

samples) parameter in the Short-Time FFT block parameters dialog.

Samples per output frame
: Enter the desired frame length of the frame-based output signal.

Input is conjugate symmetric
: Select this check box when th e input to the block is both ßoating point and conjugate symmetric, and you want real-valued outputs. When you select this check box when the input is not conjugate symmetric, the output of the block is invalid. This parameter cannot be used for Þxed-point signals.

Assert if analysis window does not support perfect signal reconstruction
: Select this check box to disp lay an error when the analysis window used by the Short-Time FFT block does not support perfect signal reconstruction.

Analysis window length
: Enter the length of the analysis window. This parameter is visible when you select the Assert if analysis window does not support perfect signal reconstruction check box.

Reconstruction error tolerance
: Enter the amount of acceptable e rror in the reconstruction of the original signal. This parameter is visible when you select the Assert if analysis window does not support perfect signal reconstruction check box.

References

Quatieri, Thomas E. Discrete-Time Speech Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 2001.

Supported Data Types

Port	Supported Data Types
X(n,k)	• Double-precision ßoating point • Single-precision ßoating point

Port	Supported Data Types
w(n)	• Double-precision ßoating point • Single-precision ßoating point
x(n)	• Double-precision ßoating point • Single-precision ßoating point

See Also

Burg Method	Signal Processing Blockset
Magnitude FFT	Signal Processing Blockset
Periodogram	Signal Processing Blockset
Short-Time FFT	Signal Processing Blockset
Spectrum Scope	Signal Processing Blockset
Window Function	Signal Processing Blockset
Yule-Walker Method	Signal Processing Blockset
pwelch	Signal Processing Toolbox

Inverse Sinc Filter

Purpose

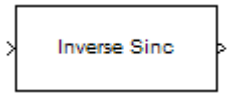
Design inverse sinc filter

Library

Filtering / Filter Design Toolbox

dspfdesign

Description



This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See “Inverse Sinc Filter Design Dialog Box — Main Pane” in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset product.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point

Port	Supported Data Types
	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Kalman Adaptive Filter (Obsolete)

Purpose

Compute filter estimates for inputs using Kalman adaptive filter algorithm

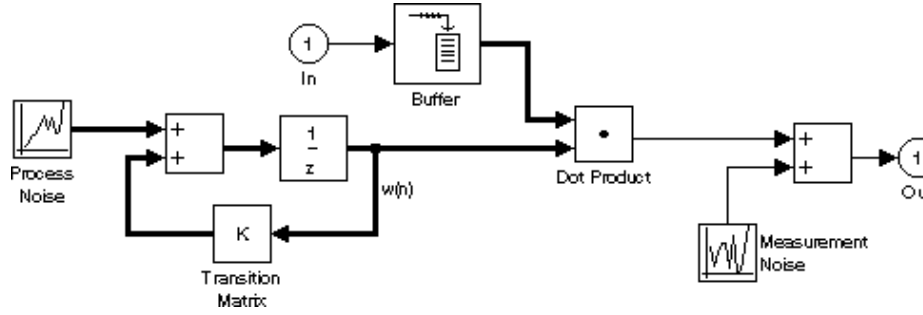
Library

dspobslib

Description

Note The Kalman Adaptive Filter block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the Kalman Filter block.

The Kalman Adaptive Filter block computes the optimal linear minimum mean-square estimate (MMSE) of the FIR filter coefficients using a one-step predictor algorithm. This Kalman filter algorithm is based on the following physical realization of a dynamic system.



The Kalman filter assumes that there are no deterministic changes to the filter taps over time (that is, the transition matrix is identity), and that the only observable output from the system is the filter output with additive noise. The corresponding Kalman filter is expressed in matrix form as

$$g(n) = \frac{K(n-1)u(n)}{u^H(n)K(n-1)u(n) + Q_M}$$
$$y(n) = u^H(n)\hat{w}(n)$$
$$e(n) = d(n) - y(n)$$
$$\hat{w}(n+1) = \hat{w}(n) + e(n)g(n)$$
$$K(n) = K(n-1) - g(n)u^H(n)K(n-1) + Q_p$$

The variables are as follows

Variable	Description
n	The current algorithm iteration
u(n)	The buffered input samples at step n
K(n)	The correlation matrix of the state estimation error
g(n)	The vector of Kalman gains at step n
$\hat{w}(n)$	The vector of filter-tap estimates at step n
y(n)	The filtered output at step n
e(n)	The estimation error at step n
d(n)	The desired response at step n
Q_M	The correlation matrix of the measurement noise
Q_P	The correlation matrix of the process noise

The correlation matrices, Q_M and Q_P, are specified in the parameter dialog by scalar variance terms to be placed along the matrix diagonals, thus ensuring that these matrices are symmetric. The filter algorithm based on this constraint is also known as the random-walk Kalman filter.

The implementation of the algorithm in the block is optimized by exploiting the symmetry of the input covariance matrix K(n) . This decreases the total number of computations by a factor of two.

The block icon has port labels corresponding to the inputs and outputs of the Kalman algorithm. Note that inputs to the In and Err ports must be sample-based scalars with the same complexity. The signal at the Out port is a scalar, while the signal at the Taps port is a sample-based vector.

Block Ports	Corresponding Variables
In	u, the scalar input, which is internally buffered into the vector u(n)
Out	y(n), the filtered scalar output
Err	e(n), the scalar estimation error
Taps	$\hat{w}(n)$, the vector of filter-tap estimates

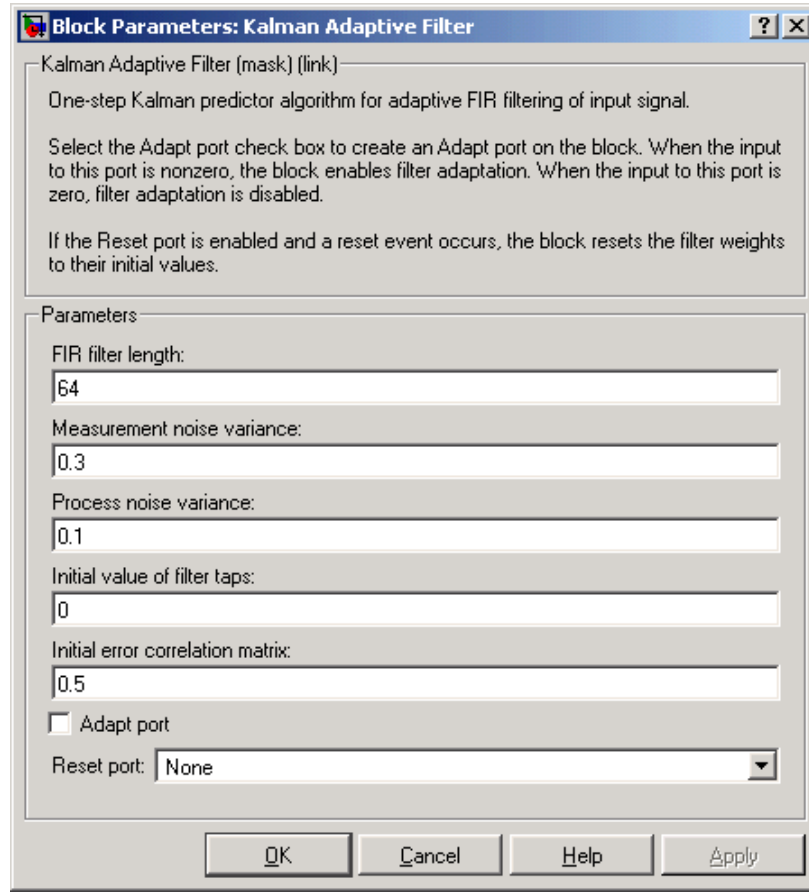
An optional Adapt input port is added when you select the Adapt port check box in the dialog. When this port is enabled, the block continuously adapts the filter coefficients while the Adapt input is nonzero. A zero-valued input to the Adapt port causes the block to stop adapting, and to hold the filter coefficients at their current values until the next nonzero Adapt input.

The FIR filter length parameter specifies the length of the filter that the Kalman algorithm estimates. The Measurement noise variance and the Process noise variance parameters specify the correlation matrices of the measurement and process noise, respectively. The Measurement noise variance must be a scalar, while the Process noise variance can be a vector of values to be placed along the diagonal, or a scalar to be repeated for the diagonal elements.

The Initial value of filter taps specifies the initial value $\hat{w}(0)$ as a vector, or as a scalar to be repeated for all vector elements. The Initial

error correlation matrix specifies the initial value K(0), and can be a diagonal matrix, a vector of values to be placed along the diagonal, or a scalar to be repeated for the diagonal elements.

Dialog Box



FIR filter length

The length of the FIR filter.

Measurement noise variance
: The value to appear along the diagonal of the measurement noise correlation matrix. Tunable.

Process noise variance
: The value to appear along the diagonal of the process noise correlation matrix. Tunable.

Initial value of filter taps
: The initial FIR filter coefficients.

Initial error correlation matrix
: The initial value of the error correlation matrix.

Adapt port
: Enables the Adapt port.

References

Haykin, S. Adaptive Filter Theory . 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1996.

Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

LMS Adaptive Filter (Obsolete)	Signal Processing Blockset
RLS Adaptive Filter (Obsolete)	Signal Processing Blockset

See “Adaptive Filters” for related information.

Purpose

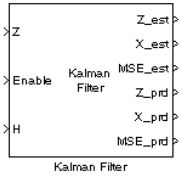
Predict or estimate states of dynamic systems

Library

Filtering/Adaptive Filters

dspadpt3

Description



Kalman Filter

Use the Kalman Filter block to predict or estimate the state of a dynamic system from a series of incomplete and/or noisy measurements. Suppose you have a noisy linear system that is defined by the following equations:

$$x_k = Ax_{k-1} + w_{k-1}$$
$$z_k = Hx_k + v_k$$

This block can use the previously estimated state, $\hat{x}_{k-1}$, to predict the current state at time k, x_k^-, as shown by the following equation:

$$x_k^- = A\hat{x}_{k-1}$$
$$P_k^- = A\hat{P}_{k-1}A^T + Q$$

The block can also use the current measurement, z_k, and the predicted state, x_k^-, to estimate the current state value at time k, $\hat{x}_k$, so that it is a more accurate approximation:

$$K_k = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1}$$
$$\hat{x}_k = x_k^- + K_k\left(z_k - Hx_k^-\right)$$
$$\hat{P}_k = (I - K_k H)P_k^-$$

The variables in the previous equations are defined in the following table.

Variable	Definition	Default Value or Initial Condition
x	State	N/A
$\hat{x}$	Estimated state	zeros([6, 1])
x^-	Predicted state	N/A
A	State transition matrix	$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$
w	Process noise	N/A
z	Measurement	N/A
H	Measurement matrix	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$
v	Measurement noise	N/A
$\hat{P}$	Estimated error covariance	10*eye(6)
P^-	Predicted error covariance	N/A
Q	Process noise covariance	0.05*eye(6)
K	Kalman gain	N/A

Variable	Definition	Default Value or Initial Condition
R	Measurement noise covariance	eye(4)
I	Identity matrix	N/A

In the previous equations, z is a vector of measurement values. Most of the time, the block processes Z , an M-by-N matrix, where M is the number of measurement values and N is the number of filters.

Use the Number of filters parameter to specify the number of filters to use to predict or estimate the current value.

Use the Enable filters parameter to specify which filters are enabled or disabled at each time step. If you select Always , the filters are always enabled. If you choose Specify via input port <Enable> , the Enable port appears on the block. The input to this port must be a row vector of 1s and 0s whose length is equal to the number of filters. For example, if there are 3 filters and the input to the Enable port is [1 0 1], only the first and third filter are enabled at this time step. If you select the Reset the estimated state and estimated error covariance when filters are disabled check box, the estimated and predicted states as well as the estimated error covariance that correspond to the disabled filters are reset to their initial values.

Note All filters have the same state transition matrix, measurement matrix, initial conditions, and noise covariance, but their state, measurement, enable, and MSE signals are unique. Within the state, measurement, enable, and MSE signals, each column corresponds to a filter.

Use the Measurement matrix source parameter to specify how to enter the measurement matrix values. If you select Specify via dialog , the Measurement matrix parameter appears in the dialog

box. If you select Input port <H> , the H port appears on the block. Use this port to specify your measurement matrix.

See the Radar Tracking demo for an example of how to use this block. You can open this demo by typing

```
aero_radmod_dsp
```

at the MATLAB command prompt.

If you have the Video and Image Processing Blockset product installed, you can also explore the People Tracking demo. Open this demo by typing

```
viptrackpeople
```

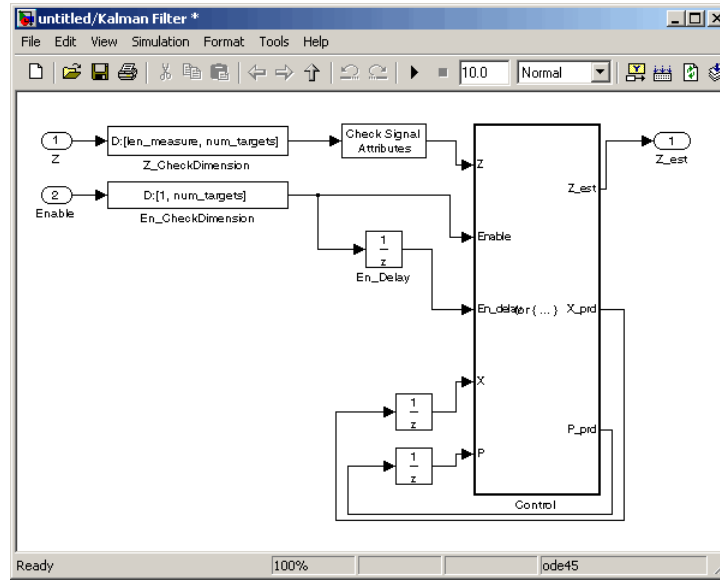
at the MATLAB command prompt.

Fixed-Point Data Types

This block can process fixed-point data types if you make several modifications to its subsystem:

1. Add a Kalman Filter block to a Simulink model.
2. Right-click the block and select Link Options > Disable Link .
3. Right-click the block and select Look Under Mask .

   The subsystem that implements the Kalman filter opens.

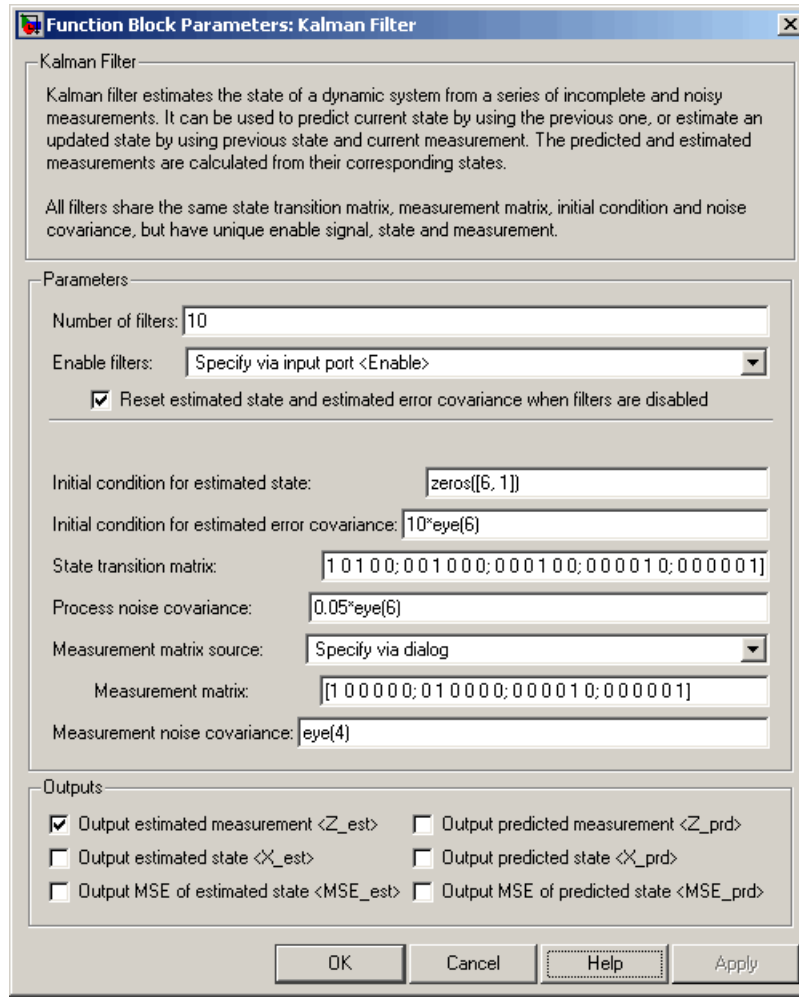


4 Double-click the Check Signal A ttributes block and change the Data type parameter to Ignore .

If you have the Video and Image Processing Blockset product installed, you can see an example by typing viptrackpeople_fixpt_all at the MATLAB command prompt. This opens the Þxed-point version of the People Tracking demo. The Kalman Filter block is located inside the Enabled Subsystem block called Tracking.

Dialog Box

The Kalman Filter dialog box appears as shown in the following figure.



Number of filters

Specify the number of filters to use to predict or estimate the current value.

Enable filters

Specify which filters are enabled or disabled at each time step. If you select Always, the filters are always enabled. If you choose Specify via input port <Enable>, the Enable port appears on the block.

Reset the estimated state and estimated error covariance when filters are disabled

If you select this check box, the estimated and predicted states as well as the estimated error covariance that correspond to the disabled filters are reset to their initial values. This parameter is visible if, for the Enable filters parameter, you select Specify via input port <Enable>.

Initial condition for estimated state

Enter the initial condition for the estimated state.

Initial condition for estimated error covariance

Enter the initial condition for the estimated error covariance.

State transition matrix

Enter the state transition matrix.

Process noise covariance

Enter the process noise covariance.

Measurement matrix source

Specify how to enter the measurement matrix values. If you select Specify via dialog, the Measurement matrix parameter appears in the dialog box. If you select Input port <H>, the H port appears on the block.

Measurement matrix

Enter the measurement matrix values. This parameter is visible if you select Specify via dialog for the Measurement matrix source parameter.

Measurement noise covariance

Enter the measurement noise covariance.

Output estimated measurement <Z_est>
: Select this check box if you want the block to output the estimated measurement.

Output estimated state <X_est>
: Select this check box if you want the block to output the estimated state.

Output MSE of estimated state <MSE_est>
: Select this check box if you want the block to output the mean-squared error of t he estimated state.

Output predicted measurement <Z_prd>
: Select this check box if you want the block to output the predicted measurement.

Output predicted state <X_prd>
: Select this check box if you want the block to output the predicted state.

Output MSE of predicted state <MSE_prb>
: Select this check box if you want the block to output the mean-squared error of the predicted state.

References

[1] Haykin, Simon. Adaptive Filter Theory . Upper Saddle River, NJ: Prentice Hall, 1996.

[2] Welch, Greg and Gary Bishop, “An Introduction to the Kalman Filter,” TR 95–041, Department of Computer Science, University of North Carolina.

Supported Data Types

Port	Input/Output	Supported Data Types
Z	M-by-N measurement where M is the length of the measurement vector and N is the number of Þlters.	• Double-precision ßoating point • Single-precision ßoating point

Port	Input/Output	Supported Data Types
Enable	1-by-N vector of 1s and 0s where N is the number of Þlters.	• Double-precision ßoating point • Single-precision ßoating point • Boolean
H	M-by-P measurement matrix where M is the length of the measurement vector and P is the length of the Þlter state vectors.	Same as Z port
Z_est	M-by-N estimated measurement matrix where M is the length of the measurement vector and N is the number of Þlters.	Same as Z port
X_est	P-by-N estimated state matrix where P is the length of the Þlter state vectors and N is the number of Þlters.	Same as Z port
MSE_est	1-by-N vector that represents the mean-squared-error of the estimated state. N is the number of Þlters.	Same as Z port
Z_prd	M-by-N predicted measurement matrix where M is the length of the measurement vector and N is the number of Þlters.	Same as Z port

Port	Input/Output	Supported Data Types
X_prd	P-by-N predicted state matrix where P is the length of the Þlter state vectors and N is the number of Þlters.	Same as Z port
MSE_prd	1-by-N vector that represents the mean-squared-error of the predicted state. N is the number of Þlters.	Same as Z port

See Also

LDL Solver

Signal Processing Blockset

Purpose

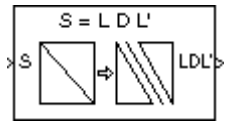
Factor square Hermitian positive definite matrices into lower, upper, and diagonal components

Library

Math Functions / Matrices and Linear Algebra / Matrix Factorizations

dspfactors

Description



The LDL Factorization block uniquely factors the square Hermitian positive definite input matrix S as

$$S = LDL^*$$

where L is a lower triangular square matrix with unity diagonal elements, D is a diagonal matrix, and L^* is the Hermitian (complex conjugate) transpose of L. Only the diagonal and lower triangle of the input matrix are used, and any imaginary component of the diagonal entries is disregarded.

The block's output is a composite matrix with lower triangle elements l_{ij} from L, diagonal elements d_{ij} from D, and upper triangle elements u_{ij} from L^*. It is always sample based. The output format is shown below for a 5-by-5 matrix.

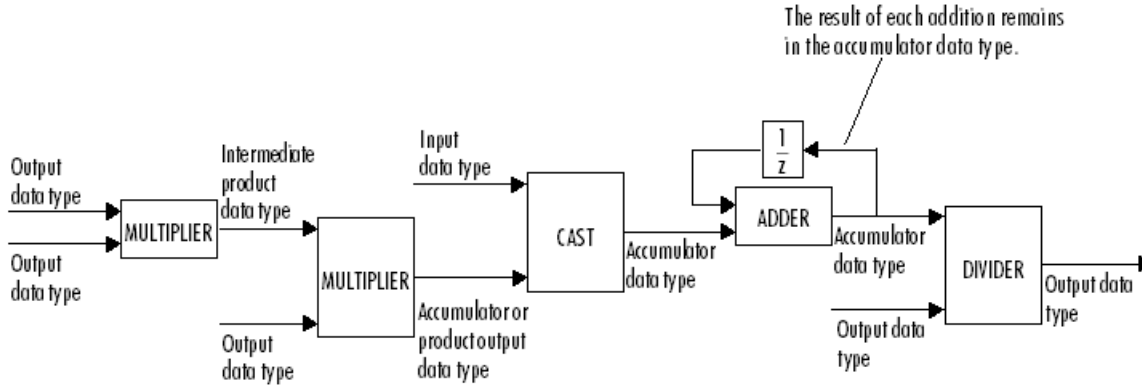
$$\begin{bmatrix} d_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ l_{21} & d_{22} & u_{23} & u_{24} & u_{25} \\ l_{31} & l_{32} & d_{33} & u_{34} & u_{35} \\ l_{41} & l_{42} & l_{43} & d_{44} & u_{45} \\ l_{51} & l_{52} & l_{53} & l_{54} & d_{55} \end{bmatrix} \qquad u_{ij} = l_{ji}^*$$

LDL factorization requires half the computation of Gaussian elimination (LU decomposition), and is always stable. It is more efficient than Cholesky factorization because it avoids computing the square roots of the diagonal elements.

The algorithm requires that the input be square and Hermitian positive definite. When the input is not positive definite, the block reacts with the behavior specified by the Non-positive definite input parameter.

Fixed-Point Data Types

The following diagram shows the data types used within the LDL Factorization block for fixed-point signals.

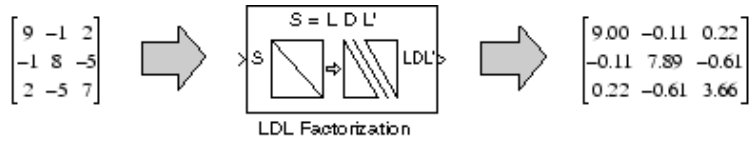


You can set the intermediate product, product output, accumulator, and output data types in the block dialog as discussed below.

The output of the second multiplier is in the product output data type when the input is real. When the input is complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see “Multiplication Data Types”.

Examples

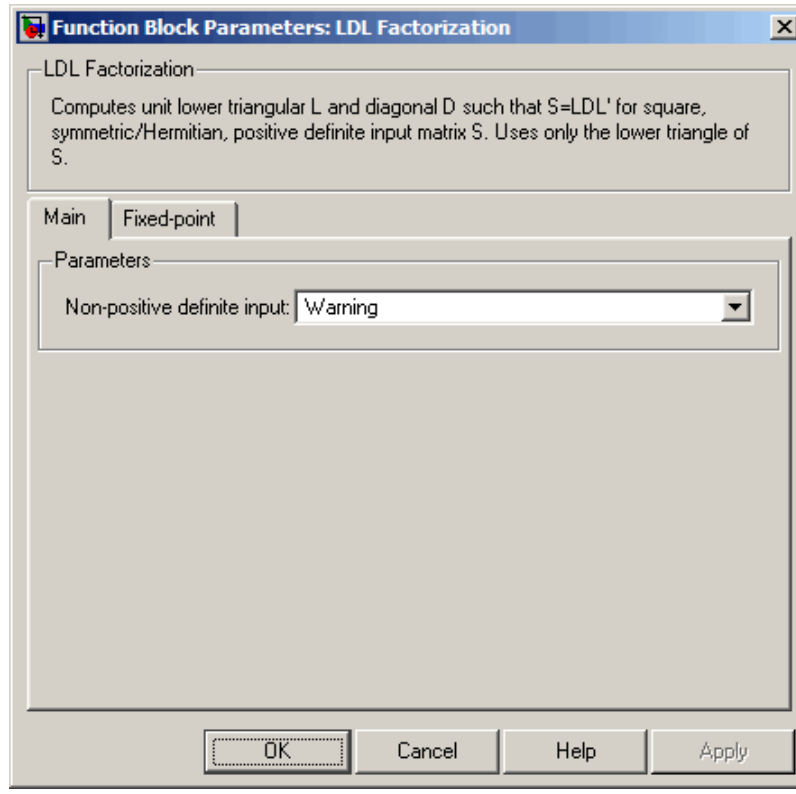
LDL decomposition of a 3-by-3 Hermitian positive de Þnite matrix:



$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.11 & 1 & 0 \\ 0.22 & -0.61 & 1 \end{bmatrix} \qquad D = \begin{bmatrix} 9.00 & 0 & 0 \\ 0 & 7.89 & 0 \\ 0 & 0 & 3.66 \end{bmatrix} \qquad L' = \begin{bmatrix} 1 & -0.11 & 0.22 \\ 0 & 1 & -0.61 \\ 0 & 0 & 1 \end{bmatrix}$$

Dialog Box

The Main pane of the LDL Factorization block dialog appears as follows.



Non-positive definite input

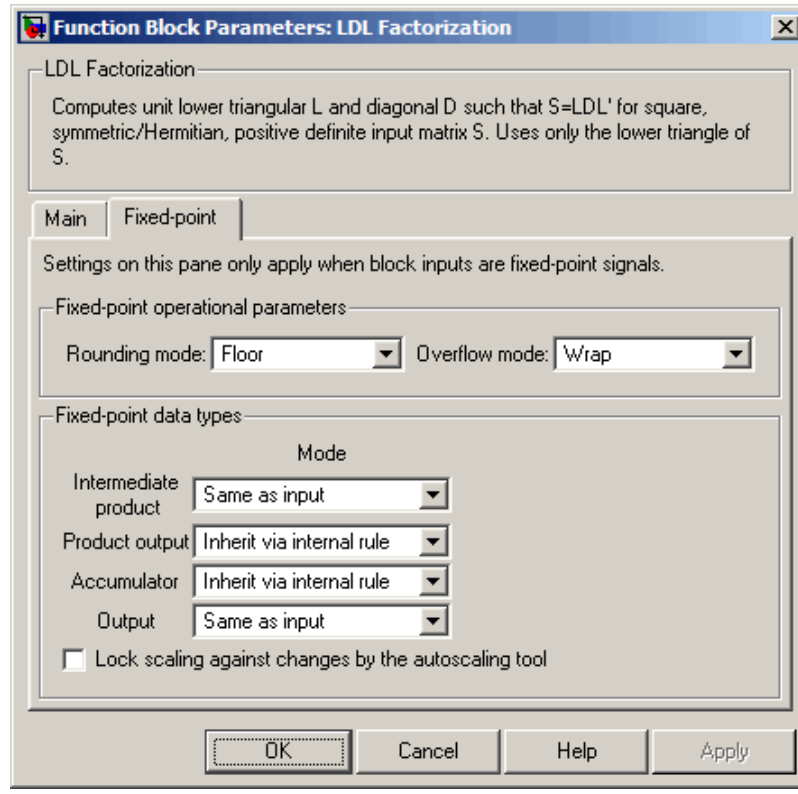
Specify the action when nonpositive definite matrix inputs occur:

- Ignore — Proceed with the computation and do not issue an alert. The output is not a valid factorization. A partial factorization is present in the upper left corner of the output.
- Warning — Display a warning message in the MATLAB Command Window, and continue the simulation. The output is

not a valid factorization. A partial factorization is present in the upper left corner of the output.

- Error — Display an error dialog and terminate the simulation.

The Fixed-point pane of the LDL Factorization block dialog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Intermediate product

Use this parameter to specify how you would like to designate the intermediate product word and fraction lengths. See “Fixed-Point Data Types” on page 2-720 for an illustration depicting the use of the intermediate product data type in this block:

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the intermediate product, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the intermediate product. This block requires power-of-two slope and a bias of zero.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-720 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Inherit via internal rule, the product output word length and fraction length are calculated automatically. For information about how the product output word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.

- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths. See “Fixed-Point Data Types” on page 2-720 and “Multiplication Data Types” for illustrations depicting the use of the accumulator data type in this block.

- When you select Inherit via internal rule, the accumulator word length and fraction length are calculated automatically. For information about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as product output, these characteristics match those of the product output.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Use this parameter to specify how you would like to designate the output word and fraction lengths. See “Fixed-Point Data Types” on page 2-720 for an illustration depicting the use of the output data type in this block:

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the output, in bits.

- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

References

Golub, G. H., and C. F. Van Loan. Matrix Computations. 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.

Supported Data Types

Port	Supported Data Types
S	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
LDL'	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

See Also

Cholesky Factorization	Signal Processing Blockset
LDL Inverse	Signal Processing Blockset
LDL Solver	Signal Processing Blockset
LU Factorization	Signal Processing Blockset
QR Factorization	Signal Processing Blockset

See “Matrix Factorizations” for related information.

LDL Inverse

Purpose

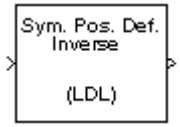
Compute inverse of Hermitian positive definite matrix using LDL factorization

Library

Math Functions / Matrices and Linear Algebra / Matrix Inverses

dspinverses

Description



The LDL Inverse block computes the inverse of the Hermitian positive definite input matrix S by performing an LDL factorization.

$$S^{-1} = \left(LDL^{*}\right)^{-1}$$

L is a lower triangular square matrix with unity diagonal elements, D is a diagonal matrix, and L^{*} is the Hermitian (complex conjugate) transpose of L. Only the diagonal and lower triangle of the input matrix are used, and any imaginary component of the diagonal entries is disregarded. The output is always sample based.

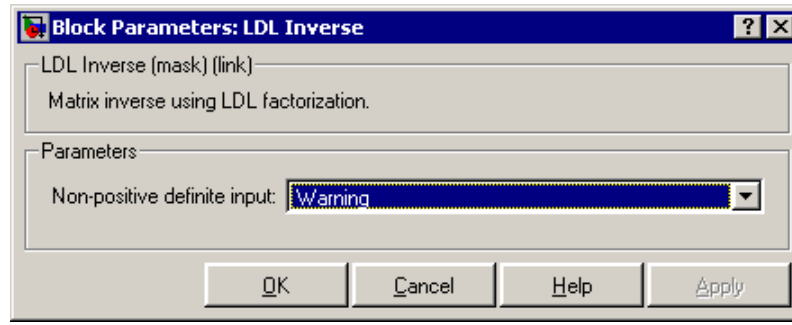
LDL factorization requires half the computation of Gaussian elimination (LU decomposition), and is always stable. It is more efficient than Cholesky factorization because it avoids computing the square roots of the diagonal elements.

The algorithm requires that the input be Hermitian positive definite. When the input is not positive definite, the block reacts with the behavior specified by the Non-positive definite input parameter. The following options are available:

- Ignore — Proceed with the computation and do not issue an alert. The output is not a valid inverse.
- Warning — Display a warning message in the MATLAB command window, and continue the simulation. The output is not a valid inverse.
- Error — Display an error dialog and terminate the simulation.

Note The Non-positive definite input parameter is a diagnostic parameter. Like all diagnostic parameters on the Configuration Parameters dialog, it is set to Ignore in the code generated for this block by Real-Time Workshop code generation software.

Dialog Box



Non-positive definite input
Response to nonpositive definite matrix inputs.

References

Golub, G. H., and C. F. Van Loan. Matrix Computations. 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.

Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

Cholesky Inverse	Signal Processing Blockset
LDL Factorization	Signal Processing Blockset
LDL Solver	Signal Processing Blockset
LU Inverse	Signal Processing Blockset

Pseudoinverse	Signal Processing Blockset
inv	MATLAB

See “Matrix Inverses” for related information.

Purpose

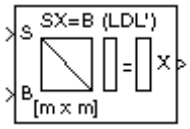
Solve SX=B for X when S is square Hermitian positive definite matrix

Library

Math Functions / Matrices and Linear Algebra / Linear System Solvers

dspsolvers

Description



The LDL Solver block solves the linear system SX=B by applying LDL factorization to the matrix at the S port, which must be square (M-by-M) and Hermitian positive definite. Only the diagonal and lower triangle of the matrix are used, and any imaginary component of the diagonal entries is disregarded. The input to the B port is the right side M-by-N matrix, B. The output is the unique solution of the equations, M-by-N matrix X, and is always sample based.

A length-M 1-D vector input for right side B is treated as an M-by-1 matrix.

When the input is not positive definite, the block reacts with the behavior specified by the Non-positive definite input parameter. The following options are available:

- Ignore — Proceed with the computation and do not issue an alert. The output is not a valid solution.
- Warning — Proceed with the computation and display a warning message in the MATLAB Command Window. The output is not a valid solution.
- Error — Display an error dialog and terminate the simulation.

Note The Non-positive definite input parameter is a diagnostic parameter. Like all diagnostic parameters on the Configuration Parameters dialog, it is set to Ignore in the code generated for this block by Real-Time Workshop code generation software.

Algorithm

The LDL algorithm uniquely factors the Hermitian positive definite input matrix S as

$$S = LDL^*$$

where L is a lower triangular square matrix with unity diagonal elements, D is a diagonal matrix, and L^* is the Hermitian (complex conjugate) transpose of L.

The equation

$$LDL^*X = B$$

is solved for X by the following steps:

1 Substitute

$$Y = DL^*X$$

2 Substitute

$$Z = L^*X$$

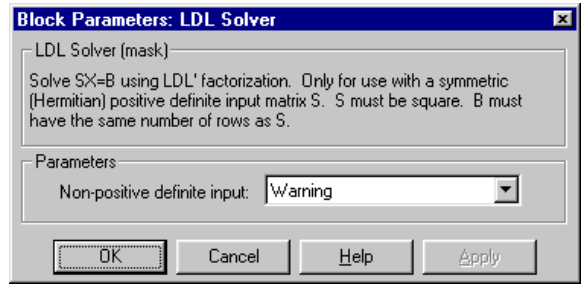
3 Solve one diagonal and two triangular systems.

$$LY = B$$

$$DZ = Y$$

$$L^*X = Z$$

Dialog Box



Non-positive definite input
Response to nonpositive definite matrix inputs.

Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

Autocorrelation LPC	Signal Processing Blockset
Cholesky Solver	Signal Processing Blockset
LDL Factorization	Signal Processing Blockset
LDL Inverse	Signal Processing Blockset
Levinson-Durbin	Signal Processing Blockset
LU Solver	Signal Processing Blockset
QR Solver	Signal Processing Blockset

See “Linear System Solvers” for related information.

Least Squares FIR Filter Design (Obsolete)

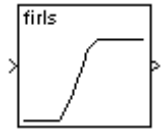
Purpose

Design and implement least-squares FIR filter

Library

dspobslib

Description



Note The Least Squares FIR Filter Design block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the Digital Filter block.

The Least Squares FIR Filter Design block designs an FIR filter and applies it to a discrete-time input using the Direct Form II Transpose Filter block. The filter design uses the Signal Processing Toolbox firls function to minimize the integral of the squared error between the desired frequency response and the actual frequency response.

An M-by-N sample-based matrix input is treated as M*N independent channels, and an M-by-N frame-based matrix input is treated as N independent channels. In both cases, the block filters each channel independently over time, and the output has the same size and frame status as the input.

The Filter type parameter allows you to specify one of the following filters:

- Multiband — The Multiband filter designs a linear-phase filter with an arbitrary magnitude response.
- Differentiator — The Differentiator filter approximates the ideal differentiator. Differentiators are antisymmetric FIR filters with approximately linear magnitude responses. To obtain the correct derivative, scale the Gains at these frequencies vector by π F_s rad/s, where F_s is the sample frequency in Hertz.

- Hilbert Transformer — The Hilbert Transformer filter approximates the ideal Hilbert transformer. Hilbert transformers are antisymmetric FIR filters with approximately constant magnitude.

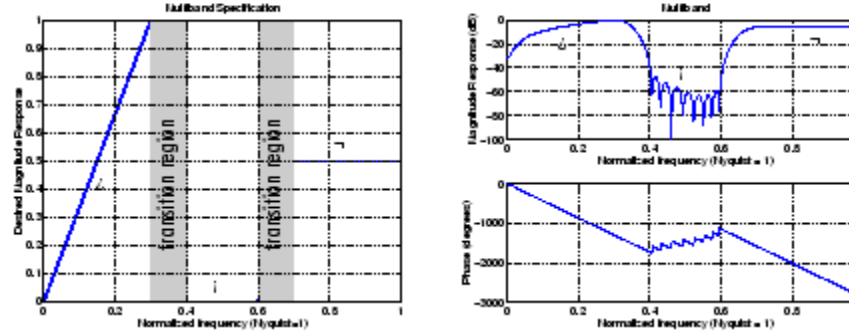
The Band-edge frequency vector parameter is a vector of frequency points in the range 0 to 1, where 1 corresponds to half the sample frequency. This vector must have even length, and intermediate points must appear in ascending order. The Gains at these frequencies parameter is a vector containing the desired magnitude response at the corresponding points in the Band-edge frequency vector .

Each odd-indexed frequency-amplitude pair defines the left endpoint of a line segment representing the desired magnitude response in that frequency band. The corresponding even-indexed frequency-amplitude pair defines the right endpoint. Between the frequency bands specified by these end-points, there may be undefined sections of the specified frequency response. These are called “don’t care” or “transition” regions, and the magnitude response in these areas is a result of the optimization in the other (specified) frequency ranges.

Band edge frequency = [0 0.3 0.4 0.6 0.7 1]

Gains = [0 1 0 0 0.5 0.5]

Band: ¿ ¡ ¬



The Weights parameter is a vector that specifies the emphasis to be placed on minimizing the error in certain frequency bands relative to others. This vector specifies one weight per band, so it is half the length of the Band-edge frequency vector and Gains at these frequencies vectors.

In most cases, differentiators and Hilbert transformers have only a single band, so the weight is a scalar value that does not affect the final filter. However, the Weights parameter is useful when using the block to design an antisymmetric multiband filter, such as a Hilbert transformer with stopbands.

For more information on the Band-edge frequency vector, Gains at these frequencies, and Weights parameters, see "Filter Designs and Implementation" in the Signal Processing Toolbox documentation. For more on the FIR filter algorithm, see the description of the firls function in the Signal Processing Toolbox documentation.

Examples

Example 1: Multiband

Consider a lowpass filter with a transition band in the normalized frequency range 0.4 to 0.5, and 10 times more error minimization in the stopband than the passband. In this case,

- Filter type = Multiband
- Band-edge frequency vector = [0 0.4 0.5 1]
- Gains at these frequencies = [1 1 0 0]
- Weights = [1 10]

Example 2: Differentiator

Assume the specifications for a differentiator filter require it to have order 21. The "ramp" response extends over the entire frequency range. In this case, specify:

- Filter type = Differentiator
- Filter order = 21

- Band-edge frequency vector = [0 1]
- Gains at these frequencies = [0 pi *Fs]

For a type III (even order) filter, the differentiation band should stop short of half the sample frequency. For example, if the filter order is 20, you could specify the block parameters as follows:

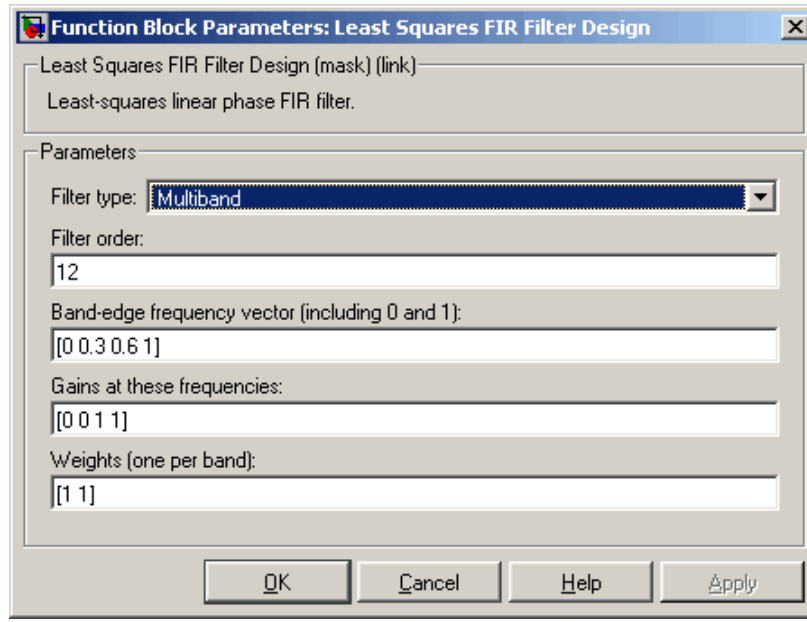
- Filter type = Differentiator
- Filter order = 20
- Band-edge frequency vector = [0 0.9]
- Gains at these frequencies = [0 0.9 *pi *Fs]

Example 3: Hilbert Transformer

Assume the specifications for a Hilbert transformer filter require it to have order 21. The passband extends over approximately the entire frequency range. In this case, specify:

- Filter type = Hilbert Transform
- Filter order = 21
- Band-edge frequency vector = [0.1 1]
- Gains at these frequencies = [1 1]

Dialog Box



Filter type
: The filter type. Tunable.

Filter order
: The filter order.

Band-edge frequency vector
: A vector of frequency points, in ascending order, in the range 0 to 1. The value 1 corresponds to half the sample frequency. This vector must have even length. Tunable.

Gains at these frequencies
: A vector of frequency-response amplitudes corresponding to the points in the Band-edge frequency vector. This vector must be the same length as the Band-edge frequency vector. Tunable.

Weights

A vector containing one weight for each frequency band. This vector must be half the length of the Band-edge frequency vector and Gains at these frequencies vectors. Tunable.

References

Oppenheim, A. V. and R. W. Schafer. Discrete-Time Signal Processing . Englewood Cliffs, NJ: Prentice Hall, 1989.

Proakis, J. and D. Manolakis. Digital Signal Processing. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

Least Squares Polynomial Fit

Purpose

Compute polynomial coefficients that best fit input data in least-squares sense

Library

Math Functions / Polynomial Functions

dsppolyfun

Description

The Least Squares Polynomial Fit block computes the coefficients of the nth order polynomial that best fits the input data in the least-squares sense, where you specify n in the Polynomial order parameter. A distinct set of n+1 coefficients is computed for each column of the M-by-N input, u.

For a given input column, the block computes the set of coefficients, c_1, c_2, ..., c_{n+1}, that minimizes the quantity

$$\sum_{i=1}^{M}(u_i - \hat{u}_i)^2$$

where u_i is the ith element in the input column, and

$$\hat{u}_i = f(x_i) = c_1 x_i^n + c_2 x_i^{n-1} + \ldots + c_{n+1}$$

The values of the independent variable, x_1, x_2, ..., x_M, are specified as a length-M vector by the Control points parameter. The same M control points are used for all N polynomial fits, and can be equally or unequally spaced. The equivalent MATLAB code is shown below.

```
c = polyfit(x,u,n)          % Equivalent MATLAB code
```

Inputs can be frame based or sample based. For convenience, a length-M 1-D vector input is treated as an M-by-1 matrix.

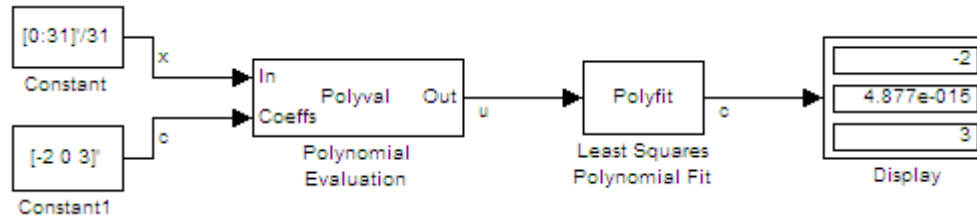
Each column of the (n+1)-by-N output matrix, c, represents a set of n+1 coefficients describing the best-fit polynomial for the corresponding column of the input. The coefficients in each column are arranged in order of descending exponents, c_1, c_2, ..., c_{n+1}. The output is always sample based.

Examples

In the model below, the Polynomial Evaluation block uses the second-order polynomial

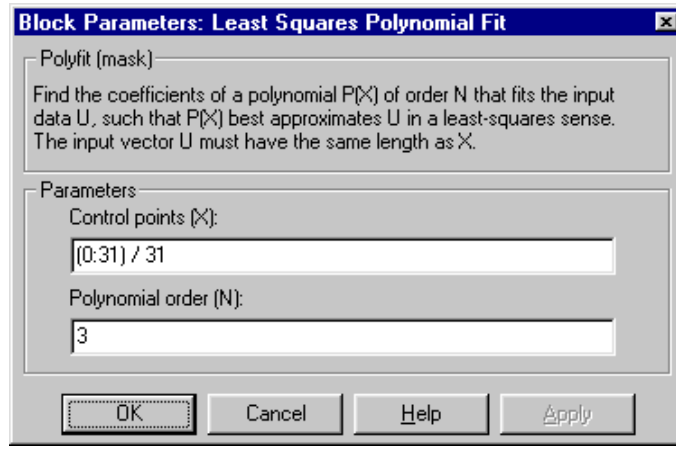
$$y = -2u^2 + 3$$

to generate four values of dependent variable y from four values of independent variable u, received at the top port. The polynomial coefficients are supplied in the vector [-2 0 3] at the bottom port. Note that the coefficient of the first-order term is zero.



The Control points parameter of the Least Squares Polynomial Fit block is configured with the same four values of independent variable u that are used as input to the Polynomial Evaluation block, [1 2 3 4]. The Least Squares Polynomial Fit block uses these values together with the input values of dependent variable y to reconstruct the original polynomial coefficients.

Dialog Box



Control points
: The values of the independent variable to which the data in each input column correspond. For an M-by-N input, this parameter must be a length- M vector. Tunable.

Polynomial order
: The order, n, of the polynomial to be used in constructing the best Þt. The number of coef Þcients is n+1.

Supported Data Types

- Double-pr ecision ßoating point
- Single-pr ecision ßoating point

See Also

Detrend	Signal Processing Blockset
Polynomi al Evaluation	Signal Processing Blockset
Polynomial Stability Test	S ignal Processing Blockset
polyfit	MATLAB

Purpose

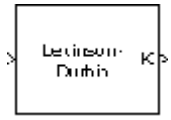
Solve linear system of equations using Levinson-Durbin recursion

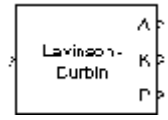
Library

Math Functions / Matrices and Linear Algebra / Linear System Solvers

dspsolvers

Description





The Levinson-Durbin block solves the nth-order system of linear equations

Ra = b

in the cases where:

- R is a Hermitian, positive-definite, Toeplitz matrix.
- b is identical to the first column of R shifted by one element and with the opposite sign.

$$\begin{bmatrix} r(1) & r^*(2) & \cdots & r^*(n) \\ r(2) & r(1) & \cdots & r^*(n-1) \\ \vdots & \vdots & \ddots & \vdots \\ r(n) & r(n-1) & \cdots & r(1) \end{bmatrix} \begin{bmatrix} a(2) \\ a(3) \\ \vdots \\ a(n+1) \end{bmatrix} = \begin{bmatrix} -r(2) \\ -r(3) \\ \vdots \\ -r(n+1) \end{bmatrix}$$

The input to the block, r = [r(1) r(2) ... r(n+1)], can be a 1-D or 2-D row or column vector or a sample- or frame-based matrix. If the input is a matrix, the block treats each column as an independent channel and solves it separately. Each channel of the input contains lags 0 through n of an autocorrelation sequence, which appear in the matrix R.

The block can output the polynomial coefficients, A, the reflection coefficients, K, and the prediction error power, P, in various combinations. The Output(s) parameter allows you to enable the A and K outputs by selecting one of the following settings:

- A — For each channel, port A outputs A=[1 a(2) a(3) ... a(n+1)], the solution to the Levinson-Durbin equation. A has the

same dimension as the input. You can also view the elements of each output channel as the coefficients of an nth-order autoregressive (AR) process.

- K — For each channel, port K outputs K=[k(1) k(2) ... k(n)], which contains n reflection coefficients and has the same dimension as the input, less one element. A scalar input channel causes an error when you select K. You can use reflection coefficients to realize a lattice representation of the AR process described later in this page.
- A and K — The block outputs both representations at their respective ports. A scalar input channel causes an error when you select A and K.

A and K are matrices if the input is a matrix. Otherwise, A and K are 1-D vectors.

Select the Output prediction error power (P) check box to output the prediction error power for each channel, P. For each channel, P represents the power of the output of an FIR filter with taps A and input autocorrelation described by r, where A represents a prediction error filter and r is the input to the block. In this case, A is a whitening filter. P has one element per input channel.

When you select the If the value of lag 0 is zero, A=[1 zeros], K=[zeros], P=0 check box (default), an input channel whose r(1) element is zero generates a zero-valued output. When you clear this check box, an input with r(1) = 0 generates NaNs in the output. In general, an input with r(1) = 0 is invalid because it does not construct a positive-definite matrix R. Often, however, blocks receive zero-valued inputs at the start of a simulation. The check box allows you to avoid propagating NaNs during this period.

Applications

One application of the Levinson-Durbin formulation implemented by this block is in the Yule-Walker AR problem, which concerns modeling an unknown system as an autoregressive process. You would model such a process as the output of an all-pole IIR filter with white Gaussian noise input. In the Yule-Walker problem, the use of the signal's

autocorrelation sequence to obtain an optimal estimate leads to an Ra = b equation of the type shown above, which is most efficiently solved by Levinson-Durbin recursion. In this case, the input to the block represents the autocorrelation sequence, with r(1) being the zero-lag value. The output at the block's A port then contains the coefficients of the autoregressive process that optimally models the system. The coefficients are ordered in descending powers of z, and the AR process is minimum phase. The prediction error, G, defines the gain for the unknown system, where $G = \sqrt{P}$:

$$H(z) = \frac{G}{A(z)} = \frac{G}{1 + a(2)z^{-1} + \ ... \ + a(n+1)z^{-n}}$$

The output at the block's K port contains the corresponding reflection coefficients, [k(1) k(2) ... k(n)], for the lattice realization of this IIR filter. The Yule-Walker AR Estimator block implements this autocorrelation-based method for AR model estimation, while the Yule-Walker Method block extends the method to spectral estimation.

Another common application of the Levinson-Durbin algorithm is in linear predictive coding, which is concerned with finding the coefficients of a moving average (MA) process (or FIR filter) that predicts the next value of a signal from the current signal sample and a finite number of past samples. In this case, the input to the block represents the signal's autocorrelation sequence, with r(1) being the zero-lag value, and the output at the block's A port contains the coefficients of the predictive MA process (in descending powers of z).

$$H(z) = A(z) = 1 + a(2)z^{-1} + \ ... \ a(n+1)z^{-n}$$

These coefficients solve the following optimization problem:

$$\min_{\{a_i\}}$$

$$E\left[\left|x_n - \sum_{i=1}^{N} a_i x_{n-i}\right|^2\right]$$

Again, the output at the block's K port contains the corresponding reßection coefÞcients, [k(1) k(2) ... k(n)] , for the lattice realization of this FIR Þlter. The Autocorrelation LPC block in the Linear Prediction library implements this autocorrelation-based prediction method.

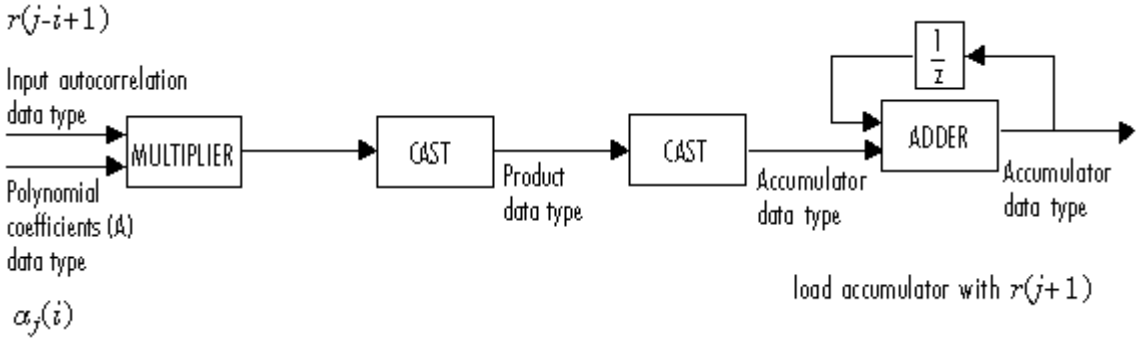
Fixed-Point Data Types

The diagrams in this section show the data types used within the Levinson-Durbin block for Þxed-point signals.

After initialization the block performs n updates. At the (j+1) update,

$$\text{value in accumulator} = r(j+1) + \sum a_j(i) \times r(j-i+1)$$

The following diagram displays the Þxed-point data types used in this calculation:



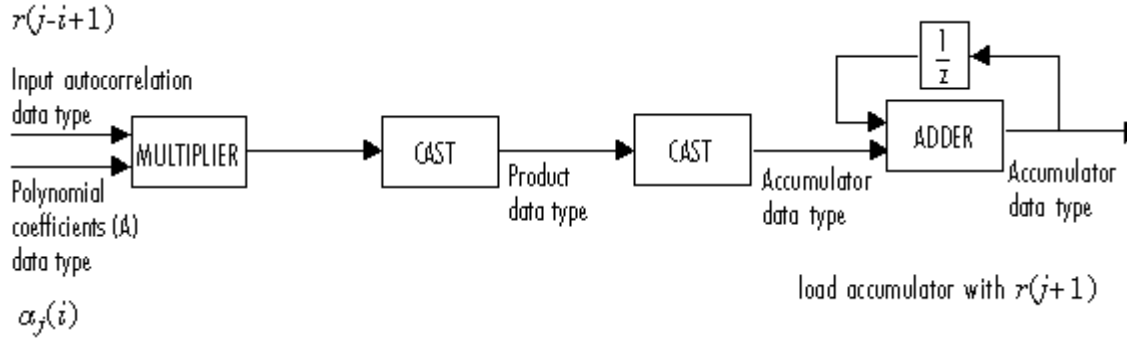
The block then updates the reßection coefÞcients K according to

$$K_{j+1} = \text{value in accumulator} / P_j$$

The block then updates the prediction error power P according to

$$P_{j+1} = P_j - P_j \times K_{j+1} \times \mathrm{conj}\left(K_{j+1}\right)$$

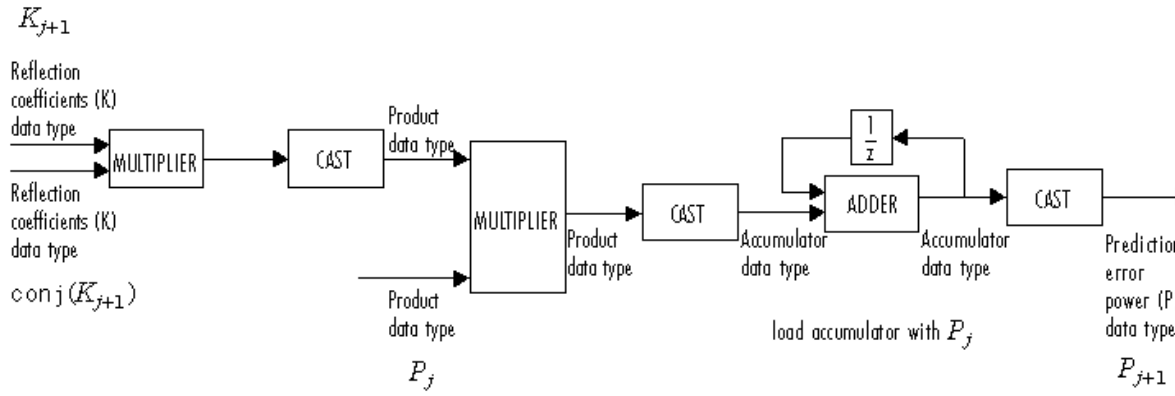
The next diagram displays the fixed-point data types used in this calculation:



The polynomial coefficients A are then updated according to

$$a_{j+1}(i) = a_j(i) + K_{j+1} \times \mathrm{conj}\left(\mathrm{a_j}(j-1+i)\right)$$

This diagram displays the fixed-point data types used in this calculation:

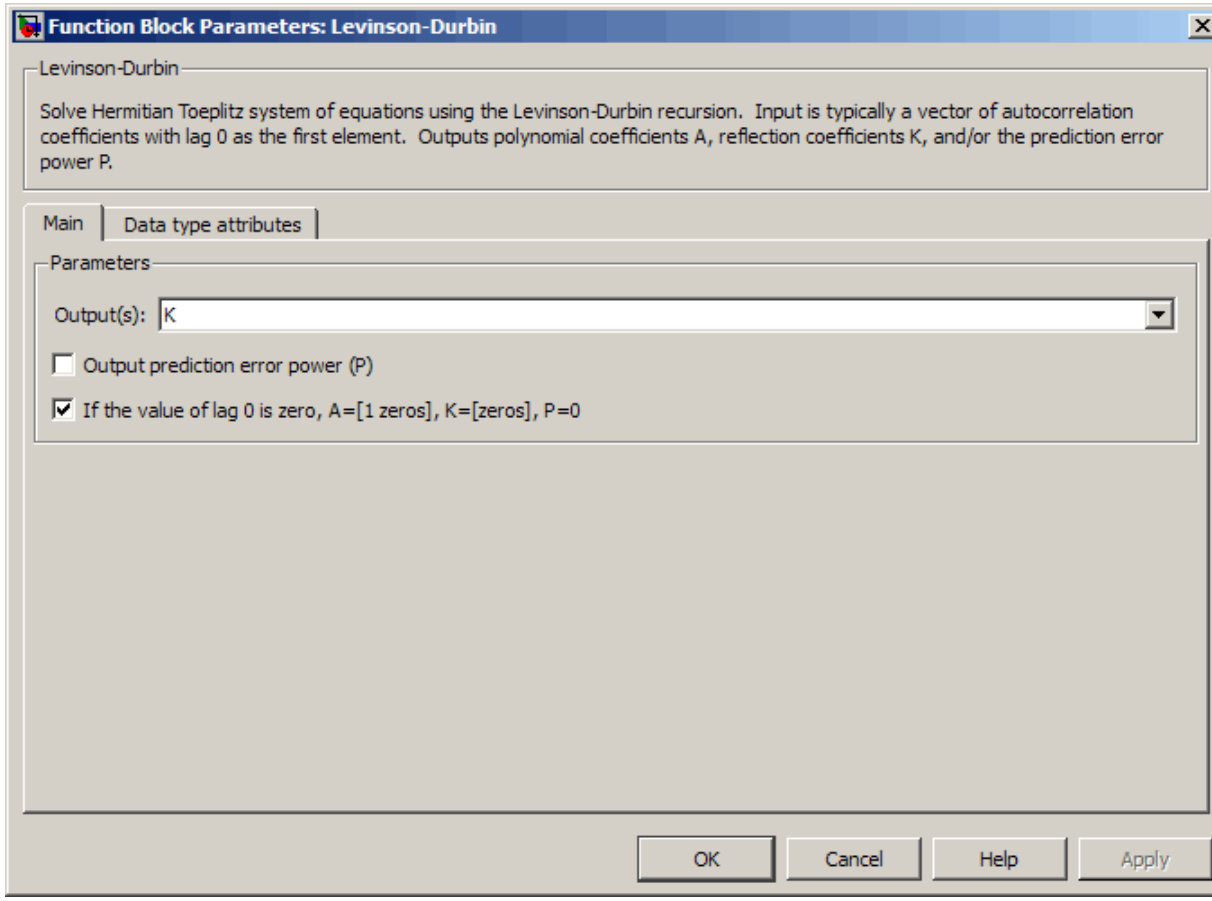


Algorithm

The algorithm requires $O(n^2)$ operations for each input channel. This implementation is therefore much more efficient for large n than standard Gaussian elimination, which requires $O(n^3)$ operations per channel.

Dialog Box

The Main pane of the Levinson-Durbin block dialog box appears as follows.



Output(s)

Specify the solution representation of Ra = b to output: model coefficients (A), reflection coefficients (K), or both (A and K). For scalar and frame-based row vector inputs, this parameter must be set to A.

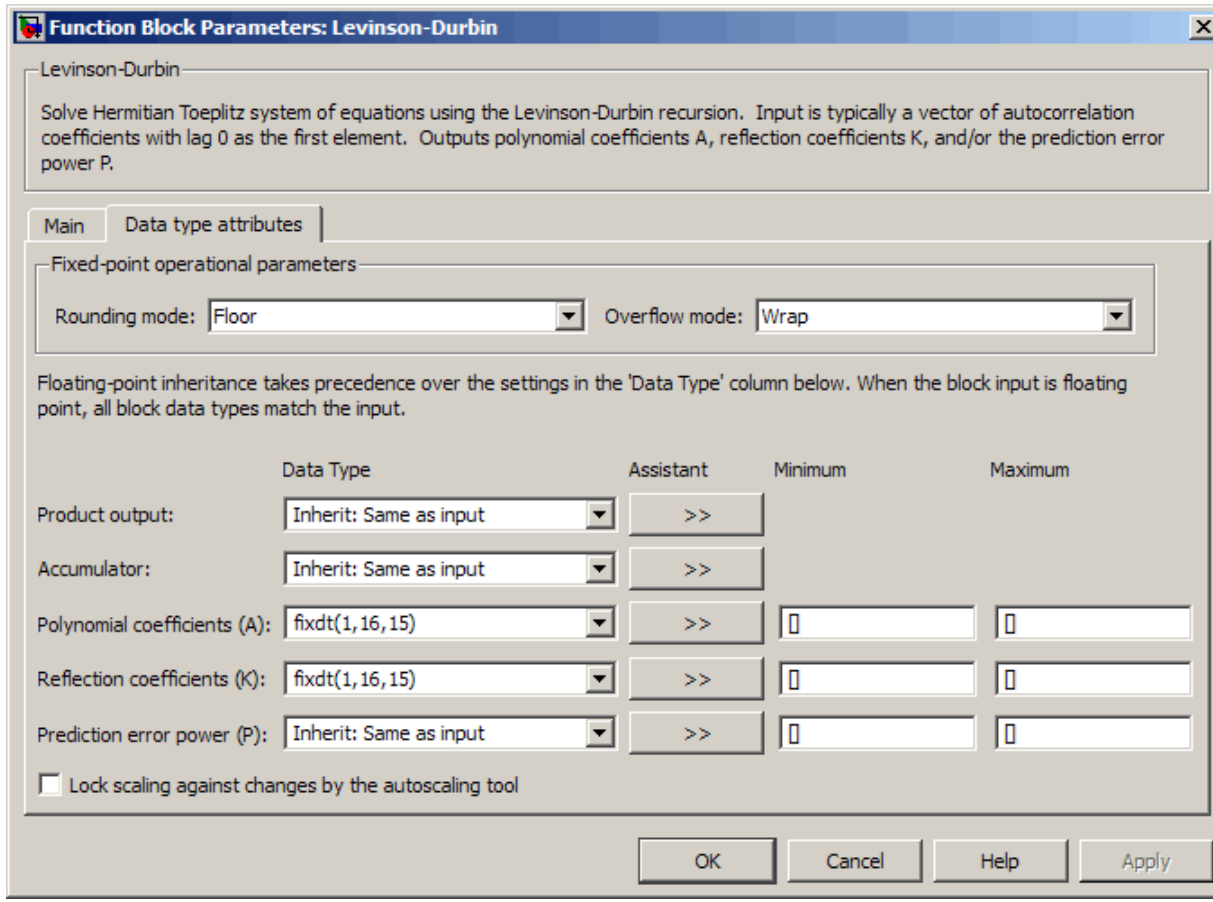
Output prediction error power (P)

Select to output the prediction error at port P.

If the value of lag 0 is zero, A=[1 zeros], K=[zeros], P=0

Set to output a zero-vector for inputs having r(1) = 0. Otherwise, the block outputs NaNs for these inputs.

The Data type attributes pane of the Levinson-Durbin block dialog box appears as follows.



Note Floating-point inheritance takes precedence over the data type settings de Þned on this pane. When inputs are ßoating point, the block ignores these settings, and a ll internal data types are ßoating point.

Rounding mode

Select the rounding mode for Þxed-point operations.

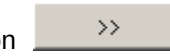
Overflow mode

Select the overflow mode for fixed-point operations.

Product output data type

Specify the product output data type. See “Fixed-Point Data Types” on page 2-746 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block. You can set it to:

- A rule that inherits a data type, for example, `Inherit: Same as input`
- An expression that evaluates to a data type, for example, `fixdt(1,16,0)`

Click the Show data type assistant button 

to display the Data Type Assistant, which helps you set the Product output data type parameter.

See “Using the Data Type Assistant” in Simulink User’s Guide for more information.

Accumulator data type

Specify the accumulator data type. See “Fixed-Point Data Types” on page 2-746 for illustrations depicting the use of the accumulator data type in this block. You can set it to:

- A rule that inherits a data type, for example, `Inherit: Same as input`
- An expression that evaluates to a data type, for example, `fixdt(1,16,0)`

Click the Show data type assistant button

to display the Data Type Assistant, which helps you set the Accumulator data type parameter.

See “Using the Data Type Assistant” in Simulink User’s Guide for more information.

Polynomial coefficients (A)

Specify the polynomial coefficients (A) data type. See "Fixed-Point Data Types" on page 2-746 for illustrations depicting the use of the A data type in this block. You can set it to an expression that evaluates to a data type, for example, fixdt(1,16,15).

Click the Show data type assistant button to display the Data Type Assistant, which helps you set the A parameter.

See "Using the Data Type Assistant" in Simulink User's Guide for more information.

Reflection coefficients (K)

Specify the polynomial coefficients (A) data type. See "Fixed-Point Data Types" on page 2-746 for illustrations depicting the use of the K data type in this block. You can set it to an expression that evaluates to a data type, for example, fixdt(1,16,15).

Click the Show data type assistant button to display the Data Type Assistant, which helps you set the K parameter.

See "Using the Data Type Assistant" in Simulink User's Guide for more information.

Prediction error power (P)

Specify the prediction error power (P) data type. See "Fixed-Point Data Types" on page 2-746 for illustrations depicting the use of the P data type in this block. You can set it to:

- A rule that inherits a data type, for example, Inherit: Same as input
- An expression that evaluates to a data type, for example, fixdt(1,16,0)

Click the Show data type assistant button to display the Data Type Assistant, which helps you set the P parameter.

See "Using the Data Type Assistant" in Simulink User's Guide for more information.

Minimum

Specify the minimum values that the polynomial coefficients, reflection coefficients, or prediction error power should have. The default value, [], is equivalent to -Inf. Simulink software uses this value to perform:

- Parameter range checking (see "Checking Parameter Values")
- Automatic scaling of fixed-point data types

Maximum

Specify the maximum values that the polynomial coefficients, reflection coefficients, or prediction error power should have. The default value, [], is equivalent to Inf. Simulink software uses this value to perform:

- Parameter range checking (see "Checking Parameter Values")
- Automatic scaling of fixed-point data types

References

Golub, G. H. and C. F. Van Loan. Sect. 4.7 in Matrix Computations. 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.

Ljung, L. System Identification: Theory for the User. Englewood Cliffs, NJ: Prentice Hall, 1987. Pgs. 278–280.

Kay, Steven M. Modern Spectral Estimation: Theory and Application. Englewood Cliffs, NJ: Prentice Hall, 1988.

Supported Data Types

- Double-precision floating point
- Single-precision floating point
- Fixed point (signed only)

- 8-, 16-, and 32-bit signed integers

See Also

Cholesky Solver	Signal Processing Blockset
LDL Solver	Signal Processing Blockset
Autocorrelation LPC	Signal Processing Blockset
LU Solver	Signal Processing Blockset
QR Solver	Signal Processing Blockset
Yule-Walker AR Estimator	Signal Processing Blockset
Yule-Walker Method	Signal Processing Blockset
levinson	Signal Processing Toolbox

See “Linear System Solvers” for related information.

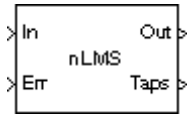
Purpose

Compute filter estimates for input using LMS adaptive filter algorithm

Library

dspobslib

Description



Note The LMS Adaptive Filter block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the LMS Filter block.

The LMS Adaptive Filter block implements an adaptive FIR filter using the stochastic gradient algorithm known as the normalized least mean-square (LMS) algorithm.

$$y(n) = \hat{w}^H(n-1)u(n)$$

$$e(n) = d(n) - y(n)$$

$$\hat{w}(n) = \hat{w}(n-1) + \frac{u(n)}{a + u^H(n)u(n)}\mu e^*(n)$$

The variables are as follows.

Variable	Description
n	The current algorithm iteration
u(n)	The buffered input samples at step n
$\hat{w}(n)$	The vector of filter-tap estimates at step n
y(n)	The filtered output at step n
e(n)	The estimation error at step n
d(n)	The desired response at step n
μ	The adaptation step size

To overcome potential numerical instability in the tap-weight update, a small positive constant (a = 1e-10) has been added in the denominator.

To turn off normalization, clear the Use normalization check box in the parameter dialog. The block then computes the filter-tap estimate as

$$\hat{w}(n) = \hat{w}(n-1) + u(n)\mu e^{*}(n)$$

The block icon has port labels corresponding to the inputs and outputs of the LMS algorithm. Note that inputs to the In and Err ports must be sample-based scalars. The signal at the Out port is a scalar, while the signal at the Taps port is a sample-based vector.

Block Ports	Corresponding Variables
In	u, the scalar input, which is internally buffered into the vector u(n)
Out	y(n), the filtered scalar output
Err	e(n), the scalar estimation error
Taps	$\hat{w}(n)$, the vector of filter-tap estimates

An optional Adapt input port is added when you select the Adapt input check box in the dialog. When this port is enabled, the block continuously adapts the filter coefficients while the Adapt input is nonzero. A zero-valued input to the Adapt port causes the block to stop adapting, and to hold the filter coefficients at their current values until the next nonzero Adapt input.

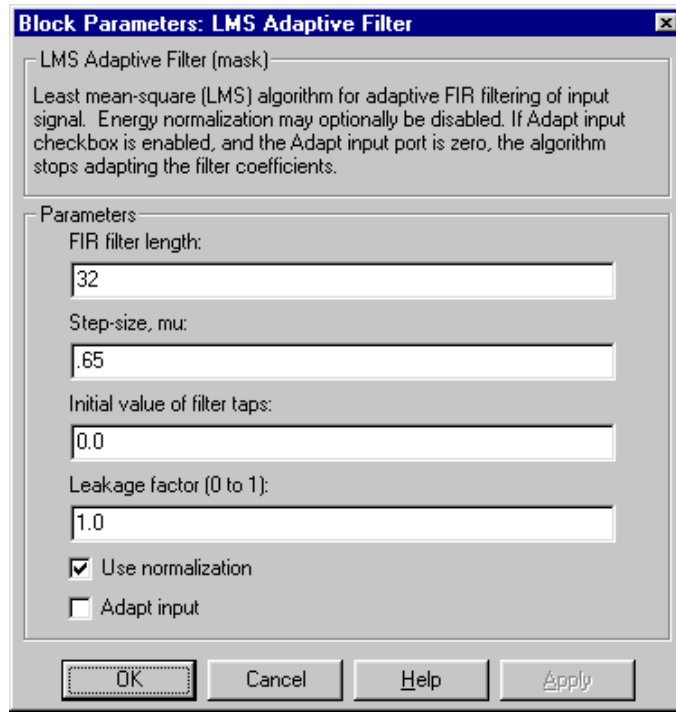
The FIR filter length parameter specifies the length of the filter that the LMS algorithm estimates. The Step size parameter corresponds to μ in the equations. Typically, for convergence in the mean square, μ must be greater than 0 and less than 2. The Initial value of filter taps specifies the initial value $\hat{w}(0)$ as a vector, or as a scalar to be repeated for all vector elements. The Leakage factor specifies the value of the leakage factor, $1 - \mu\alpha$, in the leaky LMS algorithm below. This parameter must be between 0 and 1.

$$\hat{w}(n+1) = (1-\mu\alpha)\hat{w}(n) + \frac{u(n)}{u^H(n)u(n)}\mu e^*(n)$$

Examples

See the lmsadeq and lmsadtde demos.

Dialog Box



FIR filter length
: The length of the FIR filter.

Step-size
: The step-size, usually in the range (0, 2). Tunable.

Initial value of filter taps
: The initial FIR filter coefficients.

Leakage factor
: The leakage factor, in the range [0, 1]. Tunable.

Use normalization
: Select this check box to compute the filter-tap estimate using the normalized equations.

Adapt input
Enables the Adapt port when selected.

References

Haykin, S. Adaptive Filter Theory . 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1996.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Kalman Adaptive Filter (Obsolete)	Signal Processing Blockset
RLS Adaptive Filter (Obsolete)	Signal Processing Blockset

See “Adaptive Filters” for related information.

LMS Filter

Purpose

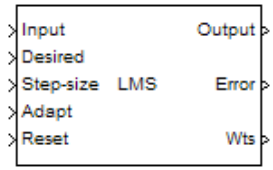
Compute output, error, and weigh ts using LMS adaptive algorithm

Library

Filtering / Adaptive Filters

dspadpt3

Description



The LMS Filter block can implement an adaptive FIR Þlter using Þve different algorithms. The block estimates the Þlter weights, or coefÞcients, needed to minimize the error, e(n), between the output signal y(n) and the desired signal, d(n). Connect the signal you want to Þlter to the Input port. This input signal can be a sample-based scalar or a single-channel frame-based signal. Connect the desired signal to the Desired port. The desired sig nal must have the same data type, frame status, complexity, and dimensions as the input signal. The Output port outputs the Þltered input signal, which is the estimate of the desired signal. The output of the Output port has the same frame status as the input signal. The Error port outputs the result of subtracting the output signal from the desired signal.

When you select LMSfor the Algorithm parameter, the block calculates the Þlter weights using the least mean-square (LMS) algorithm. This algorithm is deÞned by the following equations.

$$y(n) = \mathbf{w}^T(n-1)\mathbf{u}(n)$$
$$e(n) = d(n) - y(n)$$
$$\mathbf{w}(n) = \mathbf{w}(n-1) + f(\mathbf{u}(n), e(n), \mu)$$

The weight update function for the LMS adaptive Þlter algorithm is deÞned as

$$f(\mathbf{u}(n), e(n), \mu) = \mu e(n)\mathbf{u}^{*}(n)$$

The variables are as follows.

Variable	Description
n	The current time index
u(n)	The vector of buffered input samples at step n
u* (n)	The complex conjugate of the vector of buffered input samples at step n
w(n)	The vector of filter weight estimates at step n
y(n)	The filtered output at step n
e(n)	The estimation error at step n
d(n)	The desired response at step n
μ	The adaptation step size

When you select Normalized LMS for the Algorithm parameter, the block calculates the filter weights using the normalized LMS algorithm. The weight update function for the normalized LMS algorithm is defined as

$$f(\mathbf{u}(n), e(n), \mu) = \mu e(n)\frac{\mathbf{u}^{*}(n)}{\varepsilon + \mathbf{u}^{H}(n)\mathbf{u}(n)}$$

To overcome potential numerical instability in the update of the weights, a small positive constant, epsilon, has been added in the denominator. For double-precision floating-point input, epsilon is 2.2204460492503131e-016. For single-precision floating-point input, epsilon is 1.192092896e-07. For fixed-point input, epsilon is 0.

When you select Sign-Error LMS for the Algorithm parameter, the block calculates the filter weights using the LMS algorithm equations.

However, each time the block updates the weights, it replaces the error term e(n) with +1 when the error term is positive, -1 when it is negative, or 0 when it is zero.

When you select Sign-Data LMS for the Algorithm parameter, the block calculates the filter weights using the LMS algorithm equations. However, each time the block updates the weights, it replaces each sample of the input vector $\boldsymbol{u}$ (n) with +1 when the input sample is positive, -1 when it is negative, or 0 when it is zero.

When you select Sign-Sign LMS for the Algorithm parameter, the block calculates the filter weights using the LMS algorithm equations. However, each time the block updates the weights, it replaces the error term e(n) with +1 when the error term is positive, -1 when it is negative, or 0 when it is zero. It also replaces each sample of the input vector $\boldsymbol{u}$ (n) with +1 when the input sample is positive, -1 when it is negative, or 0 when it is zero.

Use the Filter length parameter to specify the length of the filter weights vector.

The Step size (mu) parameter corresponds to μ in the equations. For convergence of the normalized LMS equations, $0 < \mu < 2$. You can either specify a step size using the input port, Step-size, or by entering a value in the Block Parameters: LMS Filter dialog.

Use the Leakage factor (0 to 1) parameter to specify the leakage factor $1 - \mu\alpha$ where $0 < 1 - \mu\alpha \leq 1$ in the leaky LMS algorithm shown below.

$$\mathbf{w}(n) = (1 - \mu\alpha)\mathbf{w}(n-1) + f\left(\mathbf{u}(n), e(n), \mu\right)$$

When you select LMS from the Algorithm list, the weight update function in the above equation is the LMS weight update function. When you select Normalized LMS from the Algorithm list, the weight update function in the above equation is the normalized LMS weight update function.

Enter the initial filter weights w(0) as a vector or a scalar in the Initial value of filter weights text box. When you enter a scalar, the block uses the scalar value to create a vector of filter weights. This vector

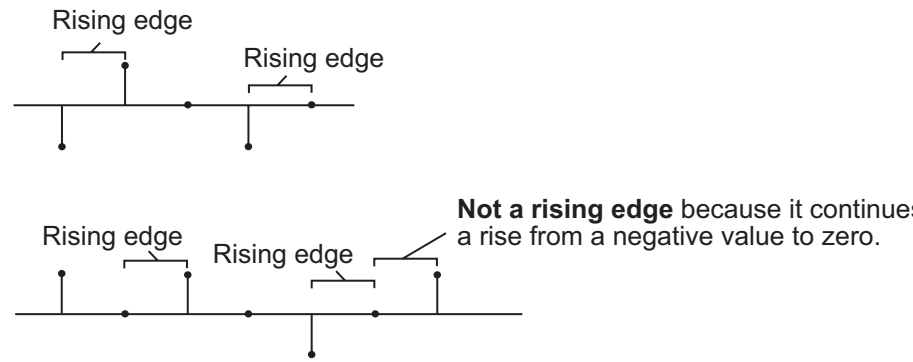
has length equal to the filter length and all of its values are equal to the scalar value.

When you select the **Adapt port** check box, an Adapt port appears on the block. When the input to this port is greater than zero, the block continuously updates the filter weights. When the input to this port is less than or equal to zero, the filter weights remain at their current values.

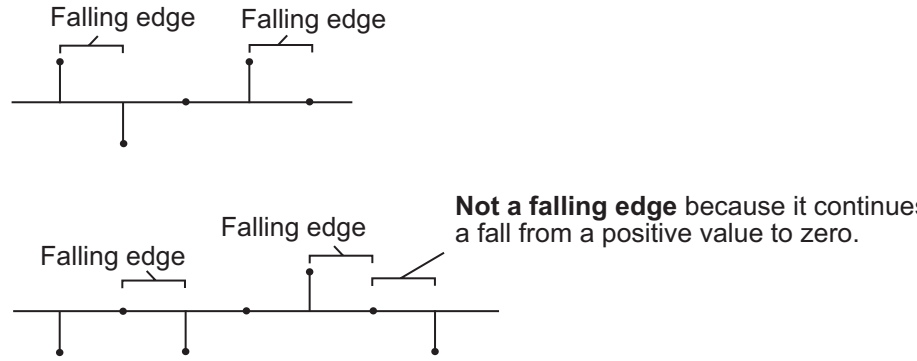
When you want to reset the value of the filter weights to their initial values, use the **Reset port** parameter. The block resets the filter weights whenever a reset event is detected at the Reset port. The reset signal rate must be the same rate as the data signal input.

From the **Reset port** list, select **None** to disable the Reset port. To enable the Reset port, select one of the following from the **Reset port** list:

- **Rising edge** — Triggers a reset operation when the Reset input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following figure)



- Falling edge — Triggers a reset operation when the Reset input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)



- Either edge — Triggers a reset operation when the Reset input is a Rising edge or Falling edge (as described above)
- Non-zero sample — Triggers a reset opera tion at each sample time that the Reset input is not zero

---

Note When running simulations in the Simulink MultiTasking mode, sample-based reset signals have a one-sample latency, and frame-based reset signals have one frame of latency. Thus, there is a one-sample or one-frame delay between the time the block detects a reset event, and when it applies the reset. For more information on latency and the Simulink tasking modes, see “Ex cess Algorithmic Delay (Tasking Latency)” and “Models with Multiple Sample Rates” in the Real-Time Workshop User’s Guide.

Select the Output filter weights check box to create a Wts port on the block. For each iteration, the block outputs the current updated filter weights from this port.

Fixed-Point Data Types

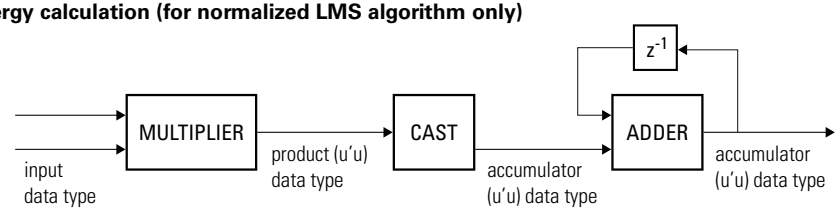
The following diagrams show the data types used within the LMS Filter block for fixed-point signals; the table summarizes the definitions of variables used in the diagrams:

Variable	Definition
u	Input vector
W	Vector of filter weights
μ	Step size
e	Error
Q	Quotient, $Q = \frac{\mu \cdot e}{\mathbf{u}'\mathbf{u}}$
Product u'u	Product data type in Energy calculation diagram
Accumulator u'u	Accumulator data type in Energy calculation diagram
Product W'u	Product data type in Convolution diagram
Accumulator W'u	Accumulator data type in Convolution diagram
Product $\mu \cdot e$	Product data type in Product of step size and error diagram
Product $Q \cdot \mathbf{u}$	Product and accumulator data type in Weight update diagram. [1]

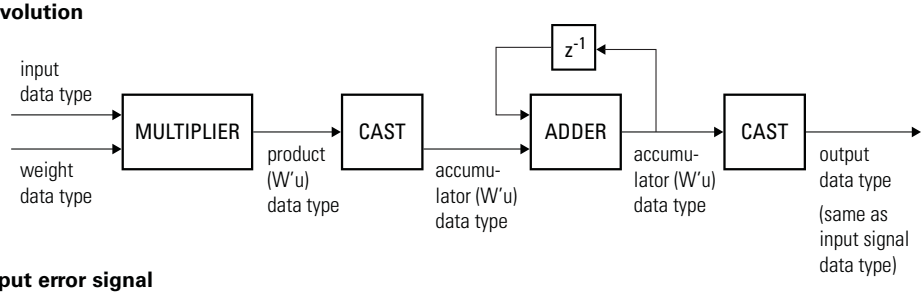
[1]The accumulator data type for this quantity is automatically set to be the same as the product data type. The minimum, maximum, and

overßow information for this accumulator is logged as part of the product information. Autoscaling tr eats this product and accumulator as one data type.

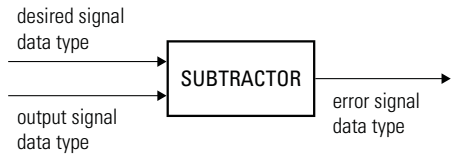
Energy calculation (for normalized LMS algorithm only)



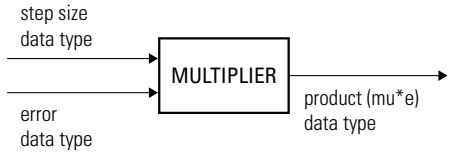
Convolution



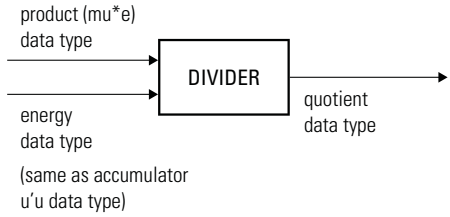
Output error signal



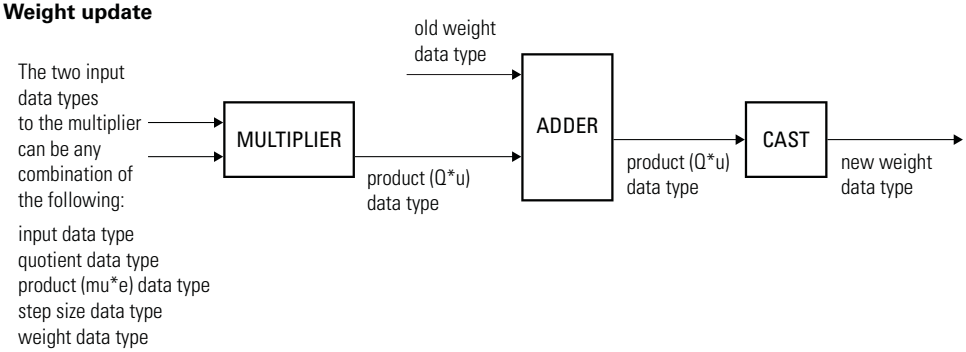
Product of step size and error (for LMS and Sign-Data LMS algorithms only)



Quotient (for normalized LMS only)



Weight update



You can set the data type of the parameters, weights, products, quotient, and accumulators in the block mask. F ixed-point inputs, outputs, and mask parameters of this block must hav e the following characteristics:

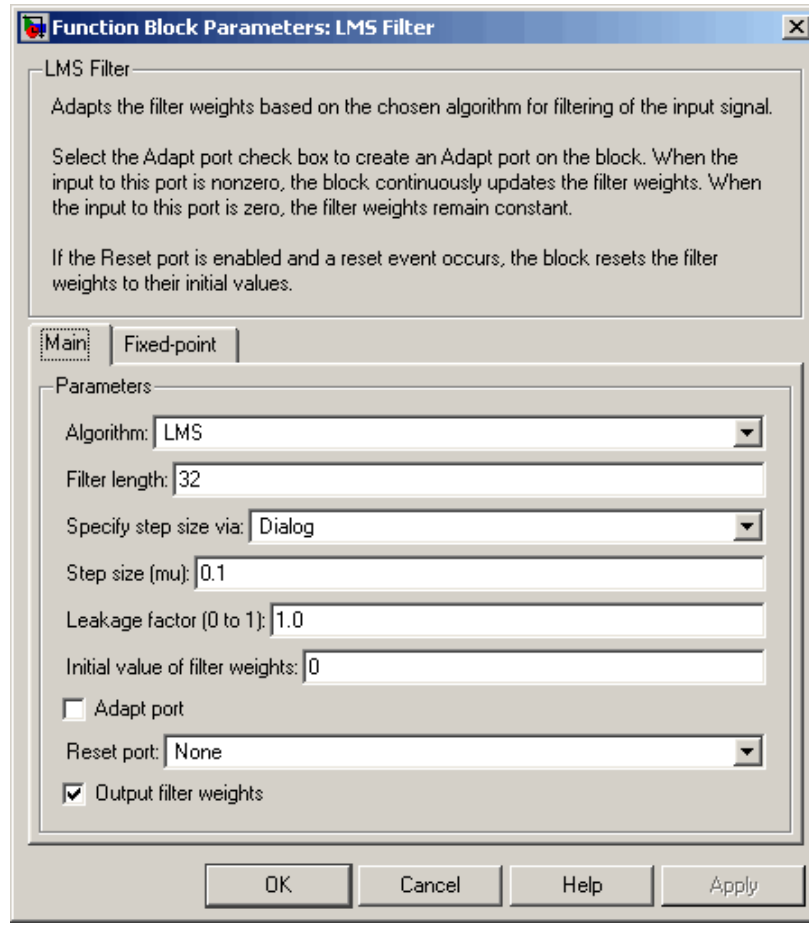
- The input signal and the desire d signal must have the same word length, but their fract ion lengths can differ.

- The step size and leakage factor must have the same word length, but their fraction lengths can differ.
- The output signal and the error signal have the same word length and the same fraction length as the desired signal.
- The quotient and the product output of the u'u, W'u, $\mu \cdot e$, and $Q \cdot \mathbf{u}$ operations must have the same word length, but their fraction lengths can differ.
- The accumulator data type of the u'u and W'u operations must have the same word length, but their fraction lengths can differ.

The output of the multiplier is in the product output data type if at least one of the inputs to the multiplier is real. If both of the inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see "Multiplication Data Types".

Dialog Box

The Main pane of the LMS Filter block dialog appears as follows.



Algorithm

Choose the algorithm used to calculate the filter weights.

Filter length

Enter the length of the FIR filter weights vector.

Specify step size via
: Select Dialog to enter a value for step size in the Block parameters: LMS Filter dialog. Select Input port to specify step size using the Step-size input port.

Step size (mu)
: Enter the step size μ. Tunable.

Leakage factor (0 to 1)
: Enter the leakage factor, $0 < 1 - \mu\alpha \leq 1$. Tunable.

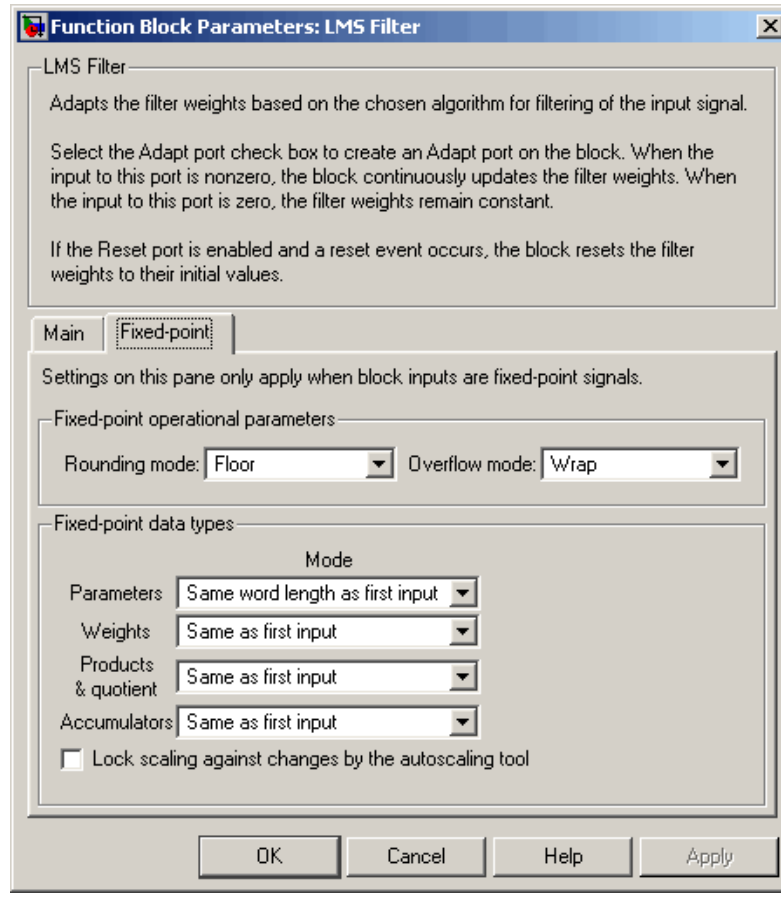
Initial value of filter weights
: Specify the initial values of the FIR filter weights.

Adapt port
: Select this check box to enable the Adapt input port.

Reset port
: Select this check box to enable the Reset input port.

Output filter weights
: Select this check box to export the filter weights from the Wts port.

The Fixed-point pane of the LMS Filter block dialog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Parameters

This parameter is visible if, for the Specify step size via parameter, you choose Dialog. Choose how you specify the word length and the fraction length of the leakage factor and step size:

- When you select Same word length as first input, the word length of the leakage factor and step size match that of the first input to the block. In this mode, the fraction length of the leakage factor and step size is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select Specify word length, you can enter the word length of the leakage factor and step size, in bits. In this mode, the fraction length of the leakage factor and step size is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select Binary point scaling, you can enter the word length and the fraction length of the leakage factor and step size, in bits. The leakage factor and the step size must have the same word length, but the fraction lengths can differ.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the leakage factor and step size. The leakage factor and the step size must have the same word length, but the slopes can differ. This block requires a power-of-two slope and a bias of zero.

If, for the Specify step size via parameter, you choose Input port, the word length of the leakage factor is the same as the word length of the step size input at the Step size port. The fraction length of the leakage factor is automatically set to the best precision possible based on the word length of the leakage factor.

Weights

Choose how you specify the word length and fraction length of the filter weights of the block:

- When you select **Same as first input**, the word length and fraction length of the filter weights match those of the first input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the filter weights, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the filter weights. This block requires a power-of-two slope and a bias of zero.

Products & quotient

Choose how you specify the word length and fraction length of u'u, W'u, $\mu \cdot e$, $Q \cdot \mathbf{u}$, and the quotient, Q. Here, u is the input vector, W is the vector of filter weights, μ is the step size, e is the error, and Q is the quotient, which is defined as $Q = \frac{\mu \cdot e}{\mathbf{u}'\mathbf{u}}$

- When you select **Same as first input**, the word length and fraction length of these quantities match those of the first input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of these quantities, in bits. The word length of the quantities must be the same, but the fraction lengths can differ.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of these quantities. The word length of the quantities must be the same, but the slopes can differ. This block requires a power-of-two slope and a bias of zero.

Accumulators

Use this parameter to specify how you would like to designate the word and fraction lengths of the accumulators for the u'u and W'u operations.

Note This parameter is not used to designate the word and fraction lengths of the accumulator for the $Q \cdot \mathbf{u}$ operation. The accumulator data type for this qu antity is automatically set to be the same as the product data type. The minimum, maximum, and overßow information for this accu mulator is logged as part of the product information. Auto scaling treats this product and accumulator as one data type.

See "Fixed-Point Data Types" on page 2-765 and "Multiplication Data Types" for illustrations depicting the use of the accumulator data type in this block:

- When you select Same as first input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulators, in bits. The word length of both the accumulators must be the same, but the fraction lengths can differ.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulators. The word length of both the accumulators must be the same, but the slopes can differ. This block requires a power-of-two slope and a bias of zero.

References

Hayes, M.H. Statistical Digital Signal Processing and Modeling. New York: John Wiley & Sons, 1996.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point
Desired	• Must be the same as Input for ßoating-point signals • Must be any Þxed-point data type when Input is Þxed point
Step-size	• Must be the same as Input for ßoating-point signals • Must be any Þxed-point data type when Input is Þxed point
Adapt	• Double-precision ßoating point • Single-precision ßoating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Reset	• Double-precision ßoating point • Single-precision ßoating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Must be the same as Input for ßoating-point signals • Must be the same as Desired for Þxed-point signals

Port	Supported Data Types
Error	• Must be the same as Input for ßoating-point signals • Must be the same as Desired for Þxed-point signals
Wts	• Must be the same as Input for ßoating-point signals • Obeys the Weights parameter for Þxed-point signals

See Also

Kalman Adaptive Filter (Obsolete)	Signal Processing Blockset
RLS Filter	Signal Processing Blockset
Block LMS Filter	Signal Processing Blockset
Fast Block LMS Filter	Signal Processing Blockset

See “Adaptive Filters” for related information.

Purpose

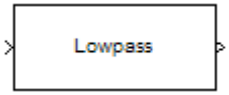
Design lowpass Filter

Library

Filtering / Filter Design Toolbox

dspfdesign

Description



This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design Þlters with this block. However, you can run models contain ing this block without a license. This allows you to run a model sent to you by a colleague who has designed a Þlter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See “Lowpass Filter Design Dialog Box — Main Pane” in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset product.

Parameters of this block that do not change Þlter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point

Port	Supported Data Types
	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Purpose

Convert linear prediction coefficients to line spectral pairs or line spectral frequencies

Library

Estimation / Linear Prediction

dsplp

Description

The LPC to LSF/LSP Conversion block takes a vector or matrix of linear prediction coefficients (LPCs) and converts it to a vector or matrix of line spectral pairs (LSPs) or line spectral frequencies (LSFs). When converting LPCs to LSFs, the block outputs match those of the poly2lsf function.

The block input must be a sample-based row vector, which is treated as a single channel, or a matrix, which is treated as a single channel per column.

The input LPCs for each channel, 1, a_1, a_2, ..., a_m, must be the denominator of the transfer function of a stable all-pole filter with the form given in the first equation of "Requirements for Valid Outputs" on page 2-779. A length-$M+1$ input channel yields a length-M output channel. Inputs can be sample based or frame based, but outputs are always sample based.

See other sections of this reference page to learn about how to ensure that you get valid outputs, how to detect invalid outputs, how the block computes the LSF/LSP values, and more.

Requirements for Valid Outputs

To get valid outputs, your inputs and the Root finding coarse grid points parameter value must meet these requirements:

- The input LPCs for each channel, 1, a_1, a_2, ..., a_m, must come from the denominator of the following transfer function, H(z), of a stable all-pole filter (all roots of H(z) must be inside the unit circle). Note that the first term in H(z)'s denominator must be 1. When the input LPCs do not come from a transfer function of the following form, the block outputs are invalid.

$$H(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2} + \ ... \ + a_m z^{-m}}$$

- The Root finding coarse grid points parameter value must be large enough so that the block can find all the LSP or LSF values. (The output LSFs and LSPs are roots of polynomials related to the input LPC polynomial; the block looks for these roots to produce the output. For details, see “LSF and LSP Computation Method: Chebyshev Polynomial Method for Root Finding” on page 2-787.) When you do not set Root finding coarse grid points to a high enough value relative to the number of LPCs, the block might not find all the LSPs or LSFs and yield invalid outputs as described in “Root Finding Method Limitations: Failure to Find Roots” on page 2-791.

To learn about recognizing invalid inputs and outputs and parameters for dealing with them, see “Handling and Recognizing Invalid Inputs and Outputs” on page 2-782.

Setting Outputs to LSFs or LSPs

Set the Output parameter to one of the following settings to determine whether the block outputs LSFs or LSPs:

- LSF in radians (0 to pi) — Block outputs the LSF values between 0 and π radians in increasing order. The block does not output the guaranteed LSF values, 0 and π.
- LSF normalized in range (0 to 0.5) — Block outputs normalized LSF values in increasing order, computed by dividing the LSF values between 0 and π radians by 2π. The block does not output the guaranteed normalized LSF values, 0 and 0.5.
- LSP in range (-1 to 1) — Block outputs LSP values in decreasing order, equal to the cosine of the LSF values between 0 and π radians. The block does not output the guaranteed LSP values, -1 and 1.

Adjusting Output Computation Time and Accuracy with Root Finding Parameters

The values n and k determine the block's out put computation time and accuracy, where

- n is the value of the Root finding coarse grid points parameter (choose this value with care; see the note below).
- k is the value of the Root finding bisection re finement parameter.
- Decreasing the values of n and k decreases the output computation time, but also decreas es output accuracy:
  - The upper bound of block's computation time is proportional to $k \cdot (n-1)$.
  - Each LSP output is within $1/(n \cdot 2^k)$ of the actual LSP value.
  - Each LSF output is within ΔLSF of the actual LSF value, LSF_{act}, where

$$\Delta LSF = \left| a\cos(LSF_{act}) - a\cos\left(LSF_{act} + 1/\left(n \cdot 2^k\right)\right)\right|$$

Note When the value of the Root finding coarse grid points parameter is too small relative to the number of LPCs, the block might output invalid data as described in "Requirements for Valid Outputs" on page 2-779. Also see "Handling and Recognizing Invalid Inputs and Outputs" on page 2-782.

Notable Input and Output Properties

- To get valid outputs, your input LPCs and the value of the Root finding coarse grid points parameter must meet the requirements described in "Requirements for Valid Outputs" on page 2-779.
- Length- L +1 input channel yields length- L output channel

- Output is always sample based
- Output parameter determines the output type (see “Setting Outputs to LSFs or LSPs” on page 2-780):
  - LSFs — frequencies, w_k, where $0 < w_k < \pi$ and $w_k < w_{k+1}$
  - Normalized LSFs — $w_k / 2\pi$
  - LSPs — $\cos(w_k)$

Handling and Recognizing Invalid Inputs and Outputs

The block outputs invalid data when your input LPCs and the value of the Root finding coarse grid points parameter do not meet the requirements described in “Requirements for Valid Outputs” on page 2-779. The following topics describe what invalid outputs look like, and how to set the block parameters provided for handling invalid inputs and outputs:

- “What Invalid Outputs Look Like” on page 2-782
- “Parameters for Handling Invalid Inputs and Outputs” on page 2-783

What Invalid Outputs Look Like

The channels of an invalid output have the same dimensions, sizes, and frame statues as the channels of a valid output. However, invalid output channels do not contain all the LSP or LSF values. Instead, they contain none or some of the LSP and LSF values and the rest of the output is filled with place holder values (-1, 0.5, or π) depending on the Output parameter setting).

In short, all invalid outputs in a channel end in one of the place holder values (-1, 0.5, or π) as illustrated in the following table. To learn how to use the block’s parameters for handling invalid inputs and outputs, see the next section.

Output Parameter Setting	Place Holder	Sample Invalid Outputs
LSF in radians (0 pi)	π	$[w_1 \quad w_2 \quad w_3 \quad \pi \quad \pi \quad \pi \quad \pi \quad \pi \ ]$

Output Parameter Setting	Place Holder	Sample Invalid Outputs
LSF normalized in range (0 0.5)	0.5	$\begin{bmatrix} w_1 \\ w_2 \\ 0.5 \end{bmatrix}$
LSP in range (-1 1) -1		$\begin{bmatrix} \cos(w_{13}) \\ \cos(w_{23}) \\ -1 \\ -1 \\ -1 \end{bmatrix}$

Parameters for Handling Invalid Inputs and Outputs

You must set how the block handles invalid inputs and outputs by setting these parameters:

- Show output validity status (1=valid, 0=invalid) — Set this parameter to activate a second output port that outputs a vector with one Boolean element per channel; 1 when the output of the corresponding chan nel is valid, and 0 when the output is invalid. The LSF and LSP outputs are invalid when the block fails to Þnd all the LSF or LSP values or when the input LPCs are unstable (for details, see “Requirements for Valid Outputs” on page 2-779). See the previous section to learn how to recognize invalid outputs.
- If current output is invalid, overwrite with previous output — Select this check box to cause the bl ock to overwrite invalid outputs with the previous output. When you set this parameter you also need to consider these parameters:
  - When Þrst output is invalid, overwrite with user-de Þned values — When the Þrst input is unstable, you can overwrite the invalid Þrst output with either

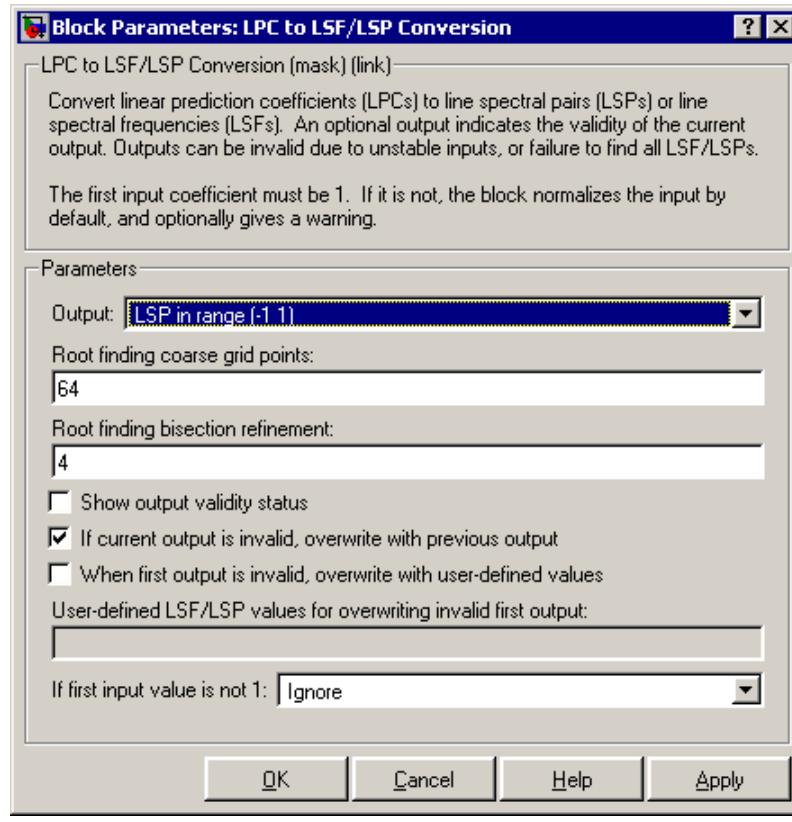
  - The default values, by clearing this check box
  - Values you specify, by selecting this check box

  The default initial overwrite values are the LSF or LSP representations of an all-pass filter. The vector that is used to overwrite invalid output is stored as an internal state.

- User-defined LSP/LSF values for overwriting invalid first output — Specify a vector of values for overwriting an invalid first output if you selected the When first output is invalid, overwrite with user-defined values parameter. For multichannel inputs, provide a matrix with the same number of channels as the input, or one vector that will be applied to every channel. The vector or matrix of LSP/LSF values you specify should have the same dimension, size, and frame status as the other outputs.

- If first input value is not 1 — The block output in any channel is invalid when the first coefficient in an LPC vector is not 1; this parameter determines what the block does when given such inputs:
  - Ignore — Proceed with computations as if the first coefficient is 1.
  - Normalize — Divide the input LPCs by the value of the first coefficient before computing the output.
  - Normalize and warn — In addition to Normalize, display a warning message at the MATLAB command line.
  - Error — Stop the simulation and display an error message at the MATLAB command line.

Dialog Box



Output

Specifies whether to convert the input linear prediction polynomial coefficients (LPCs) to LSP in range (-1 1), LSF in radians (0 pi), or LSF normalized in range (0 0.5). See "Setting Outputs to LSFs or LSPs" on page 2-780 for descriptions of the three settings.

Root finding coarse grid points

The value n, where the block divides the interval (-1, 1) into n subintervals of equal length, and looks for roots (LSP values) in each subinterval. You must pick n large enough or the block

output might be invalid as described in “Requirements for Valid Outputs” on page 2-779. To learn how the block uses this parameter to compute the output, see “LSF and LSP Computation Method: Chebyshev Polynomial Method for Root Finding” on page 2-787. Also see “Adjusting Output Computation Time and Accuracy with Root Finding Parameters” on page 2-781. Tunable.

Root finding bisection refinement

The value k, where each LSP output is within $1/(n \cdot 2^k)$ of the actual LSP value, where n is the value of the Root finding coarse grid points parameter. To learn how the block uses this parameter to compute the output, see “LSF and LSP Computation Method: Chebyshev Polynomial Method for Root Finding” on page 2-787. Also see “Adjusting Output Computation Time and Accuracy with Root Finding Parameters” on page 2-781. Tunable.

Show output validity status

Set this parameter to activate a second output port that outputs a vector with one Boolean element per channel; 1 when the output of the corresponding channel is valid, and 0 when the output is invalid. The LSF and LSP outputs are invalid when the block fails to find all the LSF or LSP values or when the input LPCs are unstable (for details, see “Requirements for Valid Outputs” on page 2-779).

If current output is invalid, overwrite with previous output

Selecting this check box causes t he block to overwrite invalid outputs with the previous output. Setting this parameter activates other parameters for taking care of initial overwrite values (when the very first output of the block is invalid). For more information, see “Paramete rs for Handling Invalid Inputs and Outputs” on page 2-783.

When first output is invalid, overwrite with user-defined values

When the first input is unstable, you can overwrite the invalid first output with either

- The default values, by clearing this check box

- Values you specify, by selecting this check box

The default initial overwrite values are the LSF or LSP representations of an all-pass filter. The vector that is used to overwrite invalid output is stored as an internal state. For more information, see "Parameters for Handling Invalid Inputs and Outputs" on page 2-783.

User-defined LSP/LSF values for overwriting invalid first output
: Specify a vector of values for overwriting an invalid first output if you selected the When first output is invalid, overwrite with user-defined values parameter. For multichannel inputs, provide a matrix with the same number of channels as the input, or one vector that will be applied to every channel. The vector or matrix of LSP/LSF values you specify should have the same dimension, size, and frame status as the other outputs.

If first input value is not 1
: Determines what the block does when the first coefficient of an input is not 1. The block can either proceed with computations as when the first coefficient is 1 (Ignore); divide the input LPCs by the value of the first coefficient before computing the output (Normalize); in addition to Normalize , display a warning message at the MATLAB command line (Normalize and warn); stop the simulation and display an error message at the MATLAB command line (Error). For more information, see "Parameters for Handling Invalid Inputs and Outputs" on page 2-783.

Theory

LSF and LSP Computation Method: Chebyshev Polynomial Method for Root Finding

---

Note To learn the principles on which the block's LSP and LSF computation method is based, see the reference listed in "References" on page 2-793.

To compute LSP outputs for each channel, the block relies on the fact that LSP values are the roots of two particular polynomials related

to the input LPC polynomial; the block finds these roots using the Chebyshev polynomial root finding method, described next. To compute LSF outputs, the block computes the arc cosine of the LSPs, outputting values ranging from 0 to π radians.

Root Finding Method

LSPs, which are the roots of two particular polynomials, always lie in the range (-1, 1). (The guaranteed roots at 1 and -1 are factored out.) The block finds the LSPs by looking for a sign change of the two polynomials' values between points in the range (-1, 1). The block searches a maximum of $k(n - 1)$ points, where

- n is the value of the Root finding coarse grid points parameter.
- k is the value of the Root finding bisection refinement parameter.

The block's method for choosing which points to check consists of the following two steps:

1 Coarse Root Finding —- The block divides the interval [-1, 1] into n intervals, each of length 2/n, and checks the signs of both polynomials' values at the endpoints of the intervals. The block starts checking signs at 1, and continues checking signs at $1 - 4/n$, $1 - 6/n$, and so on at steps of length 2/n, outputting any point if it is a root. The block stops searching in these situations:

   a The block finds a sign change of a polynomial's values between two adjacent points. An interval containing a sign change is guaranteed to contain a root, so the block further searches the interval as described in Step 2, Root Finding Refinement.

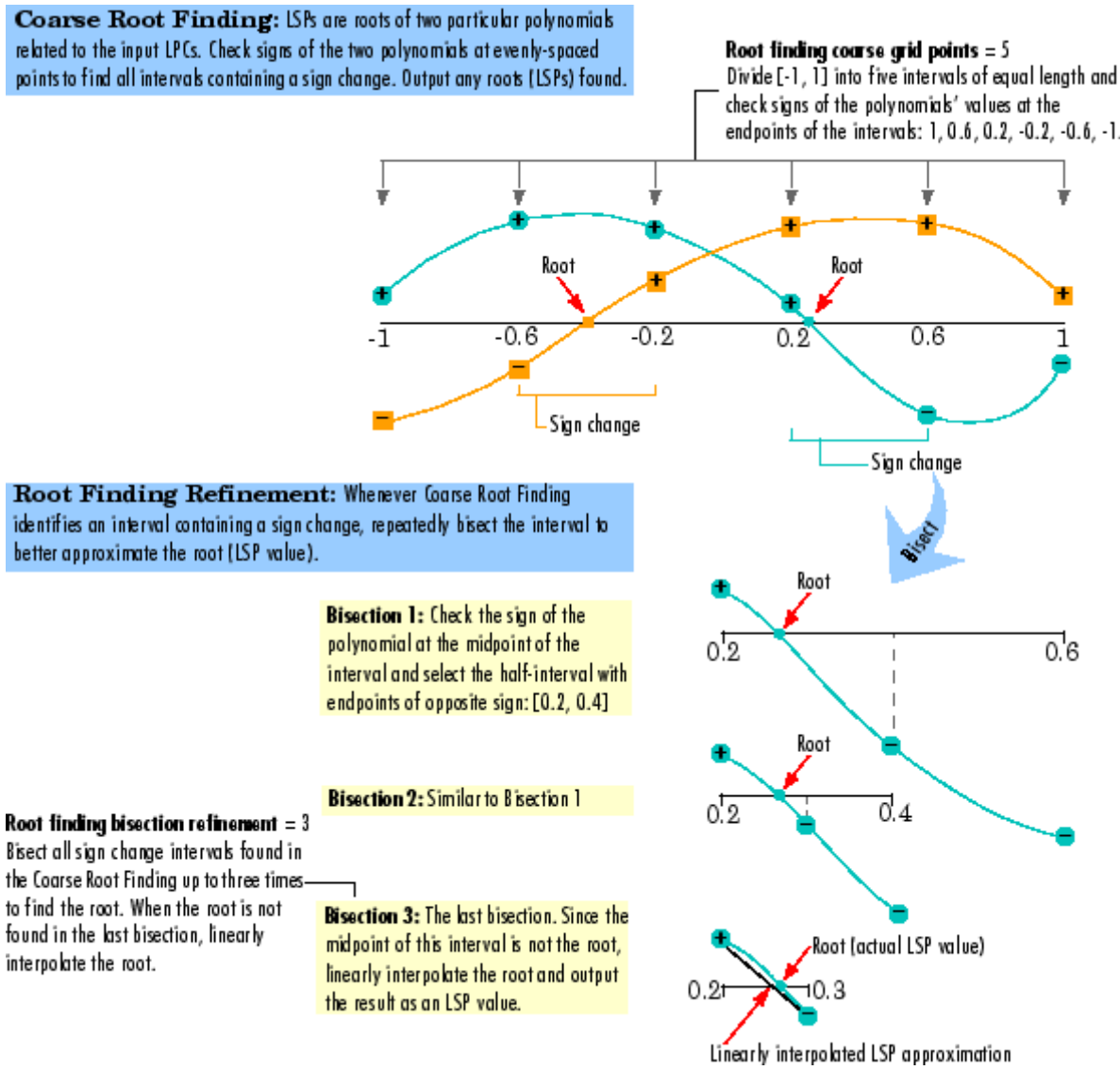
   b The block finds and outputs all M roots (given a length- M+1 LPC input).

   c The block fails to find all M roots and yields invalid outputs as described in "Handling and Recognizing Invalid Inputs and Outputs" on page 2-782.

2 Root Finding Re fi nement — When the block Þnds a sign change in an interval, [a, b], it searches for the root guaranteed to lie in the interval by following these steps:

a Check if Midpoint Is a Root — The block checks the sign of the midpoint of the interval [a, b]. The block outputs the midpoint if it is a root, and continues Step 1, Coarse Root Finding, at the next point, a – 2/n. Otherwise, the block selec ts the half-interval with endpoints of opposite sign (either [a, (a + b)/2] or [(a + b)/2, b]) and executes Step 2b, Stop or Continue Root Finding Re Þnement.

b Stop or Continue Root Finding Re fi nement — When the block has repeated Step 2a k times (k is the value of the Root fi nding bisection re fi nement parameter), the block l inearly interpolates the root by using the half-interval 's endpoints, outputs the result as an LSP value, and returns to Step 1, Coarse Root Finding. Otherwise, the block repeats Step 2a using the half-interval.



Coar se Root Finding and Root Finding Refinement

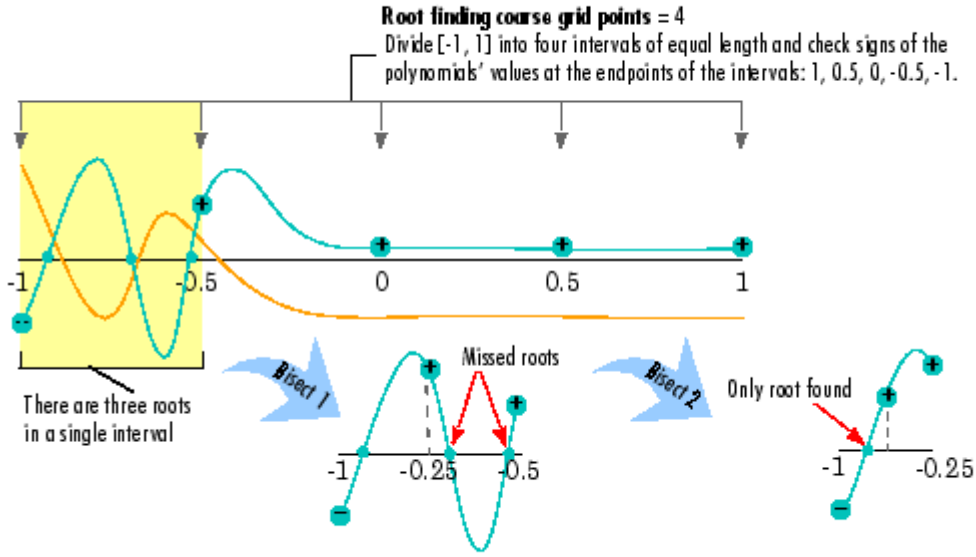
Root Finding Method Limitations: Failure to Find Roots

The block root finding method described above can fail, causing the block to produce invalid outputs (for details on invalid outputs, see “Handling and Recognizing Invalid Inputs and Outputs” on page 2-782).

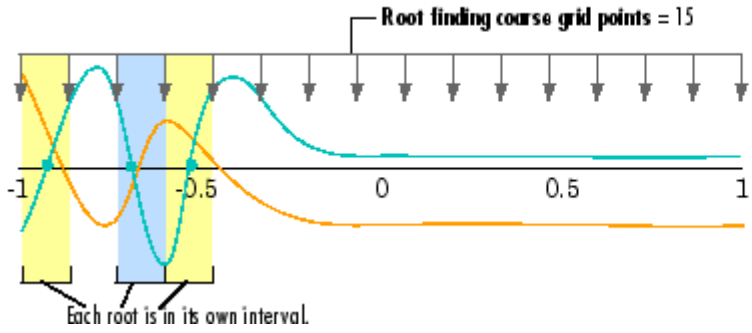
In particular, the block can fail to find some roots if the value of the **Root finding coarse grid points** parameter, `n`, is too small. If the polynomials oscillate quickly and have roots that are very close together, the root finding might be too coarse to identify roots that are very close to each other, as illustrated in Fixing a Failed Root Finding on page 2-792.

For higher-order input LPC polynomials, you should increase the **Root finding coarse grid points** value to ensure the block finds all the roots and produces valid outputs.

Root Finding Fails: The root search divides the interval [-1, 1] into four intervals, but all three roots are in a single interval. The block can only find one root per interval, so two of the roots are never found.



Fix Root Finding so it Succeeds: Increasing the value of the **Root finding course grid points** parameter to 15 ensures that each root is in its own interval, so all roots are found.



Fixing a Failed Root Finding

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Boolean — Supported only by the op tional output port that appears when you set the parameter, Show output validity status (1=valid, 0=invalid)

References

Kabal, P. and Ramachandran, R. “The Computation of Line Spectral Frequencies Using Chebyshev Polynomials.“ IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-34 No. 6, December 1986. pp. 1419-1426.

See Also

LSF/LSP to LPC Conversion	Signal Processing Blockset
LPC to/from RC	Signal Processing Blockset
LPC/RC to Autocorrelation	Signal Processing Blockset
poly2lsf	Signal Processing Toolbox

LSF/LSP to LPC Conversion

Purpose

Convert line spectral frequencies or line spectral pairs to linear prediction coefficients

Library

Estimation / Linear Prediction

dsplp

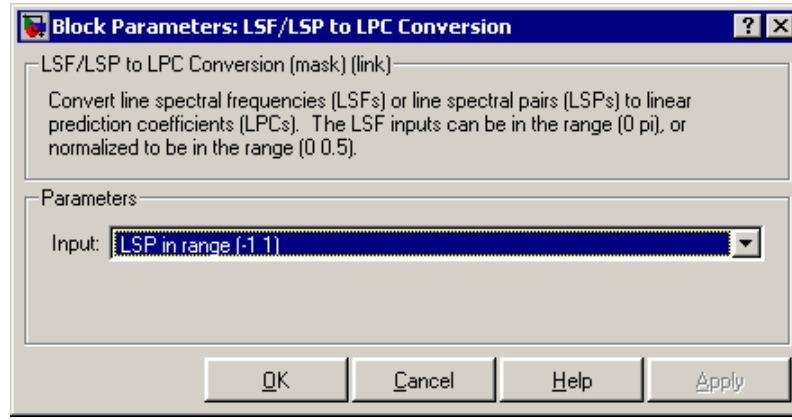
Description

The LSF/LSP to LPC Conversion block takes a vector or matrix of line spectral pairs (LSPs) or line spectral frequencies (LSFs) and converts it to a vector or matrix of linear prediction polynomial coefficients (LPCs). When converting LSFs to LPCs, the block outputs match those of the lsf2poly function.

The block input must be a sample-based row vector, which is treated as a single channel, or a matrix, which is treated as a single channel per column. Each input channel must be in the same format, which you specify in the Input parameter:

- LSF in range (0 pi) — Vector of LSF values between 0 and π radians in increasing order. Do not include the guaranteed LSF values, 0 and π.
- LSF normalized in range (0 0.5) — Vector of normalized LSF values in increasing order, (compute by dividing the LSF values between 0 and π radians by 2π). Do not include the guaranteed normalized LSF values, 0 and 0.5.
- LSP in range (-1 1) — Vector of LSP values in decreasing order, equal to the cosine of the LSF values between 0 and π radians. Do not include the guaranteed LSP values, -1 and 1.

Dialog Box



Input

Specifies whether to convert LSP in range (-1 1), LSF in range (0 pi), or LSF normalized in range (0 0.5) to linear prediction coefficients (LPCs).

Supported Data Types

- Double-precision floating point
- Single-precision floating point

References

Kabal, P. and Ramachandran, R. "The Computation of Line Spectral Frequencies Using Chebyshev Polynomials." IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-34 No. 6, December 1986. pp. 1419-1426.

See Also

LPC to LSF/LSP Conversion	Signal Processing Blockset
LPC to/from RC	Signal Processing Blockset
LPC/RC to Autocorrelation	Signal Processing Blockset
lsf2poly	Signal Processing Toolbox

LPC to/from Cepstral Coefficients

Purpose

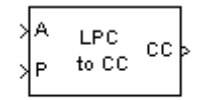
Convert linear prediction coefficients to cepstral coefficients or cepstral coefficients to linear prediction coefficients

Library

Estimation / Linear Prediction

dsplp

Description



LPC to/from Cepstral Coefficients

The LPC to/from Cepstral Coefficients block either converts linear prediction coefficients (LPCs) to cepstral coefficients (CCs) or cepstral coefficients to linear prediction coefficients. Set the Type of conversion parameter to LPCs to cepstral coefficients or Cepstral coefficients to LPCs to select the domain into which you want to convert your coefficients. The LPC port corresponds to LPCs, and the CC port corresponds to the CCs. For more information, see “Algorithm” on page 2-797.

The block input must be a sample-based row vector, which is treated as a single channel, or a matrix, which is treated as a single channel per column.

Consider a signal x(n) as the input to an FIR analysis filter represented by LPCs. The output of this analysis filter, e(n), is known as the prediction error signal. The power of this error signal is denoted by P, the prediction error power.

When you select LPCs to cepstral coefficients from the Type of conversion list, you can specify the prediction error power in two ways. From the Specify P list, choose via input port to input the prediction error power using input port P. The input to the port must be a vector with length equal to the number of input channels. Select assume P equals 1 to set the prediction error power equal to 1 for all channels.

When you select LPCs to cepstral coefficients from the Type of conversion list, the Output size same as input size check box appears. When you select this check box, the length of the input vector of LPCs is equal to the output vector of CCs. When you do not select this check box, enter a positive scalar for the Length of output cepstral coefficients parameter.

When you select LPCs to cepstral coefficients from the Type of conversion list, you can use the If first input value is not 1 parameter to specify the behavior of the block when the first coefficient of the LPC vector is not 1. The following options are available:

- Replace it with 1 —- Changes the first value of the coefficient vector to 1. The other coefficient values are unchanged.
- Normalize — Divides the entire vector of coefficients by the first coefficient so that the first coefficient of the LPC vector is 1.
- Normalize and Warn — Divides the entire vector of coefficients by the first coefficient so that the first coefficient of the LPC vector is 1. The block displays a warning message telling you that your vector of coefficients has been normalized.
- Error — Displays an error telling you that the first coefficient of the LPC vector is not 1.

When you select Cepstral coefficients to LPCs from the Type of conversion list, the Output P check box appears on the block. Select this check box when you want to output the prediction error power from output port P.

Algorithm

The cepstral coefficients are the coefficients of the Fourier transform representation of the logarithm magnitude spectrum. Consider a sequence, x(n), having a Fourier transform $X(\omega)$. The cepstrum, $c_x(n)$, is defined by the inverse Fourier transform of $C_x(\omega)$, where $C_x(\omega) = \log_e X(\omega)$. See the Real Cepstrum block reference page for information on computing cepstrum coefficients from time-domain signals.

LPC to CC

When in this mode, this block uses a recursion technique to convert LPCs to CCs. The LPC vector is defined by $\begin{bmatrix} a_0 & a_1 & a_2 & \dots & a_p \end{bmatrix}$ and the CC vector is defined by $\begin{bmatrix} c_0 & c_1 & c_2 & \dots & c_p & \dots & c_{n-1} \end{bmatrix}$. The recursion is defined by the following equations:

$$c_0 = \log_e E^2$$

$$c_m = -a_m + \frac{1}{m}\sum_{k=1}^{m-1}\left[-(m-k)\cdot a_k \cdot c_{(m-k)}\right], 1 \le m \le p$$

$$c_m = \sum_{k=1}^{p}\left[\frac{-(m-k)}{m}\cdot a_k \cdot c_{(m-k)}\right], p < m < n$$

CC to LPC

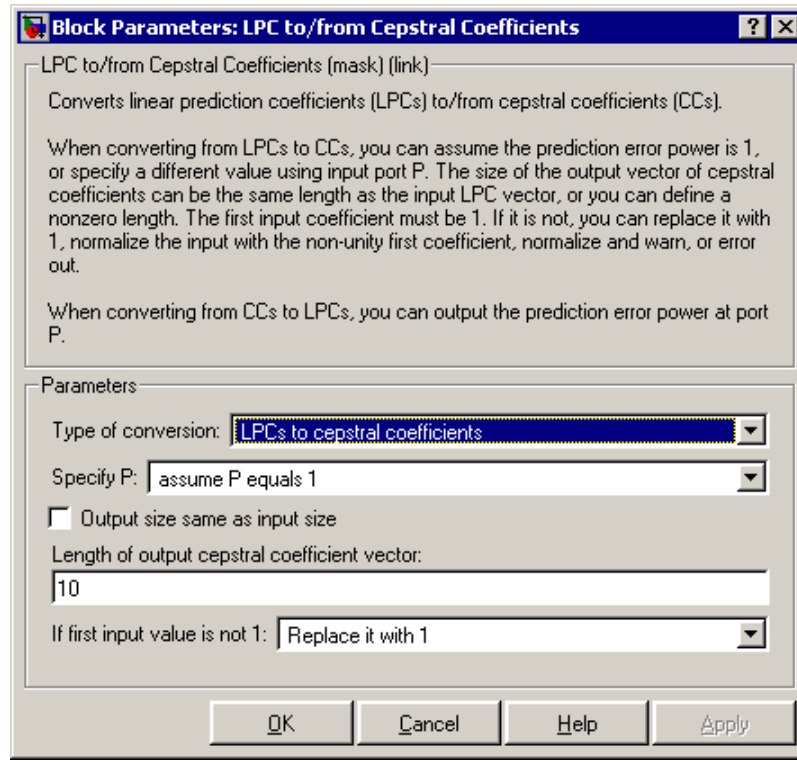
When in this mode, this block uses a recursion technique to convert CCs to LPCs. The CC vector is de Þned by $\begin{bmatrix} c_0 & c_1 & c_2 & ... & c_p & ... & c_n \end{bmatrix}$ and the LPC vector is de Þned by $\begin{bmatrix} a_0 & a_1 & a_2 & ... & a_p \end{bmatrix}$. The recursion is deÞned by the following equations

$$a_m = -c_m - \frac{1}{m}\sum_{k=1}^{m-1}\left[(m-k)\cdot c_{(m-k)} \cdot a_k\right]$$

$$P = \exp(C_0)$$

where $m = 1, 2, ..., p$.

Dialog Box



Type of conversion

Choose LPCs to cepstral coefficients or Cepstral coefficients to LPCs to specify the domain into which you want to convert your coef Þcients.

Specify P

Choose via input port to input the values of prediction error power using input port P. Select assume P equals 1 to set the prediction error power equal to 1.

Output size same as input size

When you select this check box, the length of the input vector of LPCs is equal to the output vector of CCs.

Length of output cepstral coef ficients
: Enter a positive scalar that is the length of each output channel of CCs.

If first input value is not 1
: Select what you would like the block to do when the first coefficient of the LPC vector is not 1. You can choose Replace it with 1 , Normalize , Normalize and Warn , and Error .

Output P
: Select this check box to output the prediction error power for each channel from output port P.

References

Papamichalis, Panos E. Practical Approaches to Speech Coding. Englewood Cliffs, NJ: Prentice Hall, 1987.

Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

Levinson-Durbin	Signal Processing Blockset
LPC to LSF/LSP Conversion	Signal Processing Blockset
LSF/LSP to LPC Conversion	Signal Processing Blockset
LPC to/from RC	Signal Processing Blockset
LPC/RC to Autocorrelation	Signal Processing Blockset
Real Cepstrum	Signal Processing Blockset
Complex Cepstrum	Signal Processing Blockset

Purpose

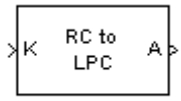
Convert linear prediction coefficients to reflection coefficients or reflection coefficients to linear prediction coefficients

Library

Estimation / Linear Prediction

dsplp

Description



LPC to/from RC

The LPC to/from RC block either converts linear prediction coefficients (LPCs) to reflection coefficients (RCs) or reflection coefficients to linear prediction coefficients. Set the Type of conversion parameter to LPC to RC or RC to LPC to select the domain into which you want to convert your coefficients. The A port corresponds to LPC coefficients, and the K port corresponds to the RC coefficients. For more information, see “Algorithm” on page 2-802.

The block input must be a sample-based row vector, which is treated as a single channel, or a matrix, which is treated as a single channel per column.

Consider a signal $x(n)$ as the input to an FIR analysis filter represented by LPC coefficients. The output of the this analysis filter, $e(n)$, is known as the prediction error signal. The power of this error signal is denoted by P. When the zero lag autocorrelation coefficient of $x(n)$ is one, the autocorrelation sequence and prediction error power are said to be normalized.

Select the Output normalized prediction error power check box to enable port P. The normalized prediction error power output at P is a vector with one element per input channel. Each element varies between zero and one.

Select the Output LPC filter stability check box to output the stability of the filter represented by the LPCs or RCs. The synthesis filter represented by the LPCs is stable when the absolute value of each of the roots of the LPC polynomial is less than one. The lattice filter represented by the RCs is stable when the absolute value of each reflection coefficient is less than 1. When the filter is stable, the block outputs a Boolean value of 1 for each input channel at the S port. When

the filter is unstable, the block outputs a Boolean value of 0 for each input channel at the S port.

If first input value is not 1 parameter specifies the behavior of the block when the first coefficient of the LPC coefficient vector in any channel is not 1. The following options are available:

- Replace it with 1 — Changes the first value of the coefficient channel to 1. The other coefficient values are unchanged.
- Normalize — Divides the entire channel of coefficients by the first coefficient so that the first coefficient of the LPC coefficient vector is 1.
- Normalize and Warn — Divides the entire channel of coefficients by the first coefficient so that the first coefficient of the LPC coefficient vector is 1. The block displays a warning message telling you that your vector of coefficients has been normalized.
- Error — Displays an error telling you that the first coefficient of the LPC coefficient channel is not 1.

Algorithm

LPC to RC

When in this mode, this block uses backward Levinson recursion to convert linear prediction coefficients (LPCs) to reflection coefficients (RCs). For a given Nth order LPC vector $LPC_N = [\ 1 \quad a_{N1} \quad a_{N2} \quad \ldots \quad a_{NN}\ ]$, the block calculates the Nth reflection coefficient value using the formula $\gamma_N = -a_{NN}$. The block then finds the lower order LPC vectors, $LPC_{N-1}, LPC_{N-2}, \ldots, LPC_1$, using the following recursion.

for p = N, N – 1, ..., 2,

$$\gamma_p = a_{pp}$$
$$F = 1 - \gamma_p{}^2$$
$$a_{p-1,m} = \frac{a_{p,m}}{F} - \frac{\gamma_p a_{p,p-m}}{F}, \quad 1 \le m < p$$

end

Finally, $\gamma_1 = -a_{11}$. The reßection coefÞcient vector is $[\gamma_1, \gamma_2, \ldots, \gamma_N]$.

RC to LPC

When in this mode, this block uses Levinson recursion to convert reßection coefÞcients (RCs) to linear prediction coef Þcients (LPCs).

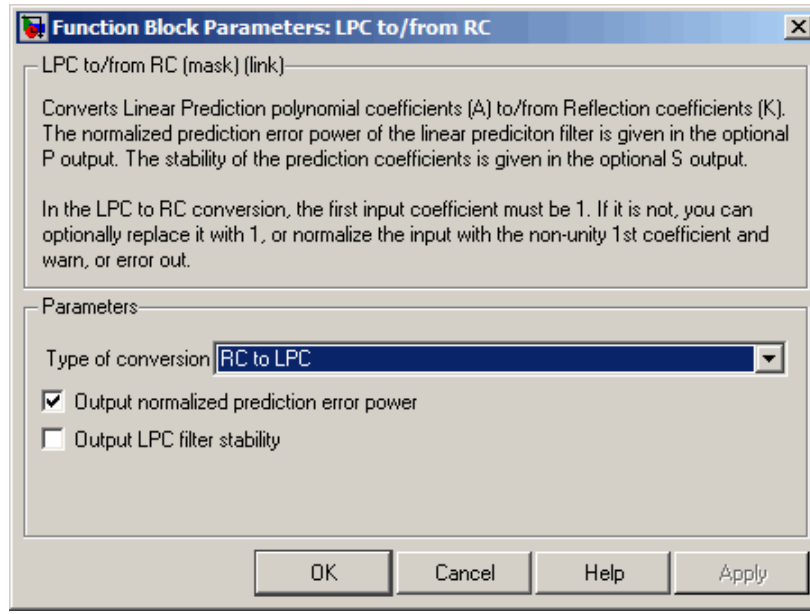
In this case, the input to the block is $RC = [\gamma_1 \quad \gamma_2 \quad \ldots \quad \gamma_N]$. The zeroth order LPC vector term is 1. Starting with this term, the block uses recursion to calculate the higher order LPC vectors, $LPC_2, LPC_3, \ldots LPC_N$, until it has calculated the entire LPC matrix.

$$LPC_{matrix} = \begin{bmatrix} LPC_0 \\ LPC_1 \\ LPC_2 \\ \cdots \\ LPC_N \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & a_{11} & 0 & 0 & \cdots & 0 \\ 1 & a_{21} & a_{22} & 0 & \cdots & 0 \\ 1 & a_{31} & a_{32} & a_{33} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & a_{N1} & a_{N2} & a_{N3} & \cdots & a_{NN} \end{bmatrix}$$

This LPC matrix consists of LPC vectors of order 0 through N found by using the Levinson recursion. The following are the formulas for the recursion steps, for p = 0, 1, ...,N – 1.

$$a_{p+1,m} = a_{p,m} + \gamma_{p+1} a_{p,p+1-m}, \quad 1 \le m \le p$$
$$a_{p+1,p+1} = \gamma_{p+1}$$

Dialog Box



Type of conversion

Select LPC to RC or RC to LPC to select the domain into which you want to convert your coefficients.

Output normalized prediction error power

Select this check box to output the normalized prediction error power at port P.

Output LPC filter stability

Select this check box to output the stability of the filter. When the filter represented by the LPCs or RCs is stable, the block outputs a Boolean value of 1 for each input channel at the S port. When the filter represented by the LPCs or RCs is unstable, the block outputs a Boolean value of 0 for each input channel at the S port.

If first input value is not 1

Select what you would like the block to do when the first coefficient of the LPC coefficient vector is not 1. You can choose Replace it with 1, Normalize, Normalize and Warn, and Error.

References

Makhoul, J Linear Prediction: A tutorial review. Proc. IEEE. 63, 63, 56 (1975).

Markel, J.D. and A. H. Gray, Jr., Linear Prediction of Speech. New York, Springer-Verlag, 1976.

Supported Data Types

- Double-precision floating-point
- Single-precision floating-point

See Also

Levinson-Durbin	Signal Processing Blockset
LPC to LSF/LSP Conversion	Signal Processing Blockset
LSF/LSP to LPC Conversion	Signal Processing Blockset
LPC/RC to Autocorrelation	Signal Processing Blockset

LPC/RC to Autocorrelation

Purpose

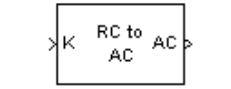
Convert linear prediction coefficients or reflection coefficients to autocorrelation coefficients

Library

Estimation / Linear Prediction

dsplp

Description



LPC/RC to Autocorrelation

The LPC/RC to Autocorrelation block either converts linear prediction coefficients (LPCs) to autocorrelation coefficients (ACs) or reflection coefficients (RCs) to autocorrelation coefficients (ACs). Set the Type of conversion parameter to LPC to autocorrelation or RC to autocorrelation to select the domain from which you want to convert your coefficients. The A port corresponds to LPC coefficients, and the K port corresponds to the RC coefficients.

The block input must be a sample-based row vector, which is treated as a single channel, or a matrix, which is treated as a single channel per column.

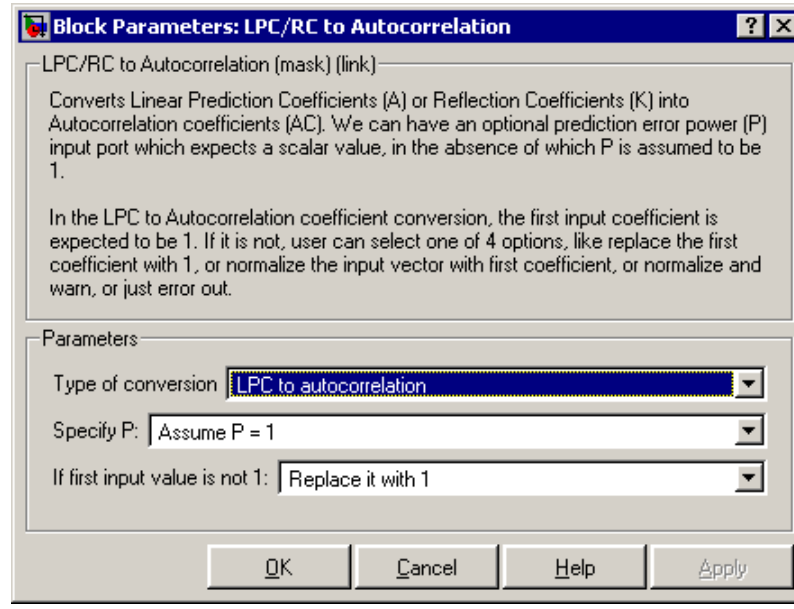
Use the Specify P parameter to set the value of the prediction error power. You can set this parameter to 1 by selecting Assume P=1. When you select Via input port, a P port appears on the block. You can use this port to input the value of the actual, non-unity prediction error power for each channel. The length of this vector must equal the number of channels in the input.

The If first input value is not 1 parameter specifies the behavior of the block when the first coefficient of the LPC coefficient vector is not 1. The following options are available:

- Replace it with 1 — The block changes the first value of the coefficient vector to 1. The rest of the coefficient values are unchanged.
- Normalize — The block divides the entire vector of coefficients by the first coefficient so that the first coefficient of the LPC coefficient vector is 1.

- Normalize and Warn — The block divides the entire vector of coefficients by the first coefficient so that the first coefficient of the LPC coefficient vector is 1. The block displays a warning message telling you that your vector of coefficients has been normalized.
- Error — The block displays an error telling you that the first coefficient of the LPC coefficient vector is not 1.

Dialog Box



Type of conversion

From the list select LPC to autocorrelation or RC to autocorrelation to specify the domain from which you want to convert your coefficients.

Specify P

From the list select Assume P=1 or Via input port to specify the value of prediction error power.

If first input value is not 1

Select what you would like the block to do when the first coefficient of the LPC coefficient vector is not 1. You can choose Replace it with 1, Normalize, Normalize and Warn, and Error.

References

Orfanidis, S.J. Optimum Signal Processing. New York, McGraw-Hill, 1988.

Makhoul, J. Linear Prediction: A tutorial review. Proc. IEEE. 63, 63, 56 (1975).

Markel, J.D. and A. H. Gray, Jr., Linear Prediction of Speech. New York, Springer-Verlag, 1976.

Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

Levinson-Durbin	Signal Processing Blockset
LPC to LSF/LSP Conversion	Signal Processing Blockset
LSF/LSP to LPC Conversion	Signal Processing Blockset
LPC to/from RC	Signal Processing Blockset

Purpose

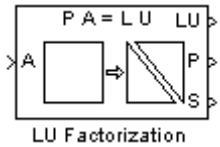
Factor square matrix into lower and upper triangular components

Library

Math Functions / Matrices and Linear Algebra / Matrix Factorizations

dspfactors

Description



LU Factorization

The LU Factorization block factors a row permutation of the square input matrix A as $A_p = L*U$, where L is the unit-lower triangular matrix, and U is the upper triangular matrix. For more information, see the lu function reference page in the MATLAB documentation. The row-pivoted matrix A_p contains the rows of A permuted as indicated by the permutation index vector P.

```
Ap = A(P,:)        % Equivalent
MATLAB code
```

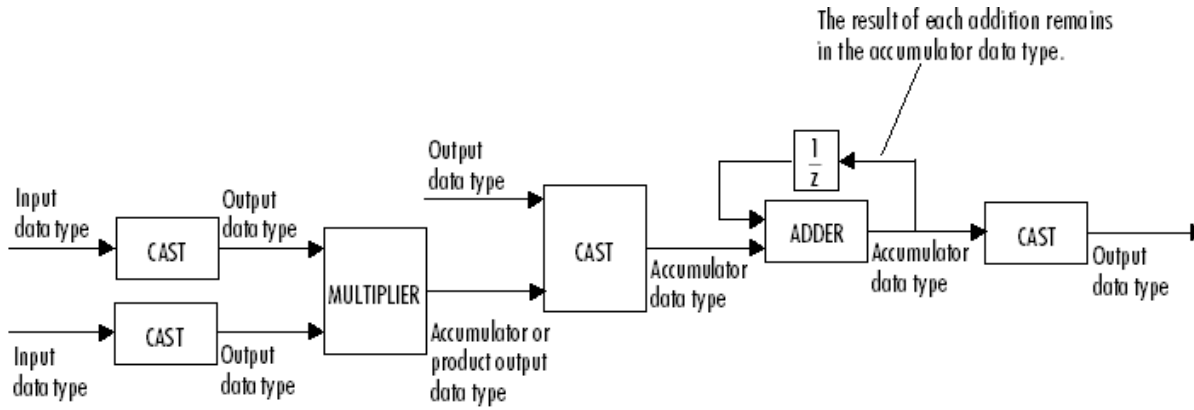
The output of the LU Factorization block at port LU is a composite matrix with lower subtriangle elements from L and upper triangle elements from U. It is always sample based. The output is not in the same form as the output of the MATLAB lu function. In order to convert the output of the LU Factorization block to the MATLAB form, use the following equations:

```
L = tril(LU,-1)+diag(ones(size(LU,1),1));
U = triu(LU);
```

Here, LU is the output of the LU Factorization block. Due to roundoff error, these equations do not produce a result that is exactly the same as the MATLAB result.

Fixed-Point Data Types

The following diagram shows the data types used within the LU Factorization block for fixed-point signals.



You can set the product output, accumulator, and output data types in the block dialog as discussed below.

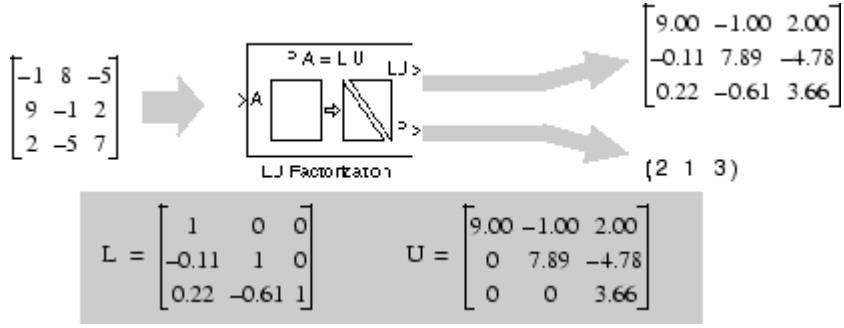
The output of the multiplier is in the product output data type when the input is real. When the input is complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see “Multiplication Data Types”.

Examples

The row-pivoted matrix A_p and permutation index vector P computed by the block are shown below for 3-by-3 input matrix A.

$$A = \begin{bmatrix} -1 & 8 & -5 \\ 9 & -1 & 2 \\ 2 & -5 & 7 \end{bmatrix} \qquad P = \begin{pmatrix} 2 & 1 & 3 \end{pmatrix} \qquad A_P = \begin{bmatrix} 9 & -1 & 2 \\ -1 & 8 & -5 \\ 2 & -5 & 7 \end{bmatrix}$$

The LU output is a composite matrix whose lower subtriangle forms L and whose upper triangle forms U.

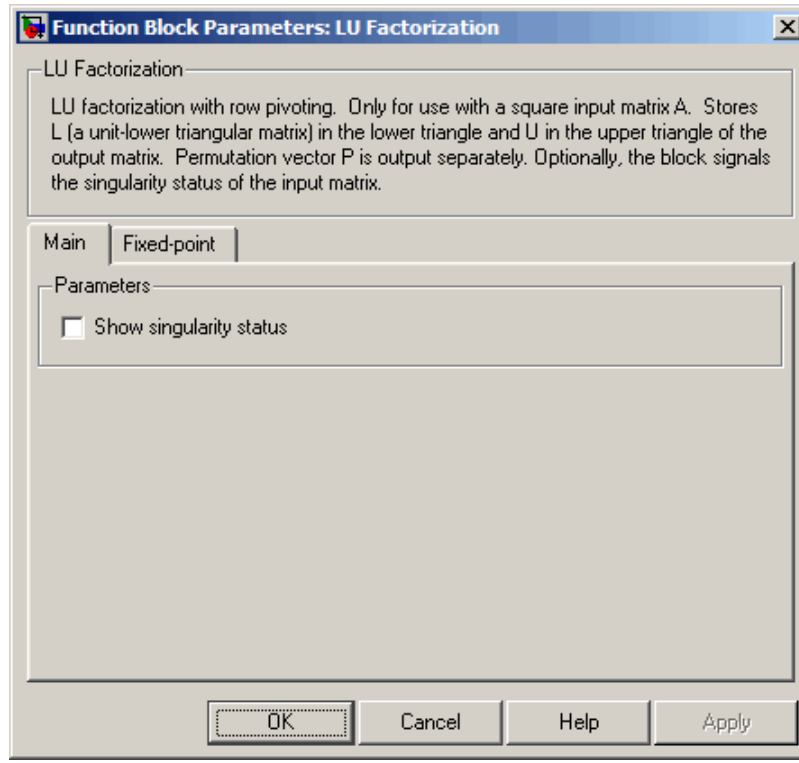


$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.11 & 1 & 0 \\ 0.22 & -0.61 & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 9.00 & -1.00 & 2.00 \\ 0 & 7.89 & -4.78 \\ 0 & 0 & 3.66 \end{bmatrix}$$

See “Example: LU Factorization” in the Signal Processing Blockset User’s Guide for another example using the LU Factorization block.

Dialog Box

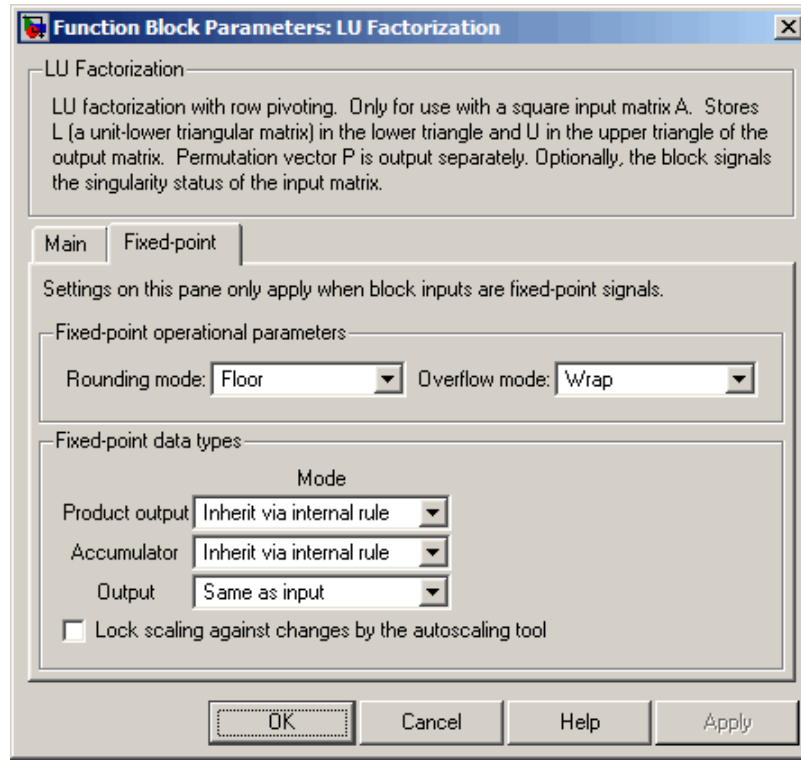
The Main pane of the LU Factorization block dialog appears as follows.



Show singularity status

Select to output the singularity of the input at port S, which outputs Boolean data type values of 1 or 0. An output of 1 indicates that the current input is singular, and an output of 0 indicates the current input is nonsingular.

The Fixed-point pane of the LU Factorization block dialog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-809 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Inherit via internal rule, the product output word length and fraction length are calculated automatically. For information about how the product output word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths. See “Fixed-Point Data Types” on page 2-809 and “Multiplication Data Types” for illustrations depicting the use of the accumulator data type in this block.

- When you select Inherit via internal rule, the accumulator word length and fraction length are calculated automatically. For information about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as product output, these characteristics match those of the product output.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.

- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Use this parameter to specify how you would like to designate the output word and fraction lengths. See “Fixed-Point Data Types” on page 2-809 for an illustration depicting the use of the output data type in this block:

- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

References

Golub, G. H., and C. F. Van Loan. Matrix Computations . 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.

Supported Data Types

Port	Supported Data Types
A	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

Port	Supported Data Types
LU	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
P	• Double-precision ßoating point • Single-precision ßoating point • 32-bit unsigned integers
S	• Boolean

See Also

Autocorrelation LPC	Sig nal Processing Blockset
Cholesky Factorization	Signal Processing Blockset
LDL Factorization	Signal Processing Blockset
LU Inverse	Signal Processing Blockset
LU Solver	Signal Processing Blockset
Permute Matrix	Signal Processing Blockset
QR Factorization	Signal Processing Blockset
lu	MATLAB

See “Matrix Factorizations” for related information.

Purpose

Compute inverse of square matrix using LU factorization

Library

Math Functions / Matrices and Linear Algebra / Matrix Inverses

dspinverses

Description

The LU Inverse block computes the inverse of the square input matrix A by factoring and inverting row-pivoted variant A_p.

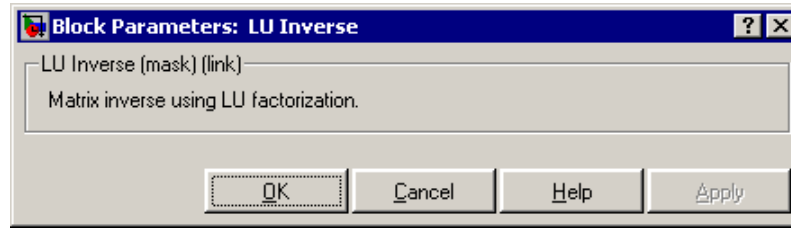
$$A_p^{-1} = (LU)^{-1}$$

L is a lower triangular square matrix with unity diagonal elements, and U is an upper triangular square matrix. The block's output is A^{-1}, and is always sample based.

Examples

See "Example: LU Inverse" in the Signal Processing Blockset User's Guide.

Dialog Box



References

Golub, G. H., and C. F. Van Loan. Matrix Computations. 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.

Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

Cholesky Inverse	Signa l Processing Blockset
LDL Inverse	Signal Processing Blockset
LU Factorization	Signal Processing Blockset
LU Solver	Signal Processing Blockset
inv	MATLAB

See “Matrix Inverses” for related information.

Purpose

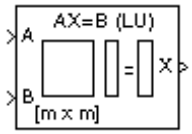
Solve AX=B for X when A is square matrix

Library

Math Functions / Matrices and Linea r Algebra / Linear System Solvers

dspsolvers

Description



The LU Solver block solves the linear system AX=B by applying LU factorization to the M-by-M matrix at the A port. The input to the B port is the right side M-by-N matrix, B. The output is the unique solution of the equations, M-by-N matrix X, and is always sample based.

A length- M 1-D vector input for right side B is treated as an M-by-1 matrix.

Algorithm

The LU algorithm factors a row-permuted variant (A_p) of the square input matrix A as

$$A_p = LU$$

where L is a lower triangular square matrix with unity diagonal elements, and U is an upper triangular square matrix.

The matrix factors are substituted for A_p in

$$A_p X = B_p$$

where B_p is the row-permuted variant of B, and the resulting equation

$$LUX = B_p$$

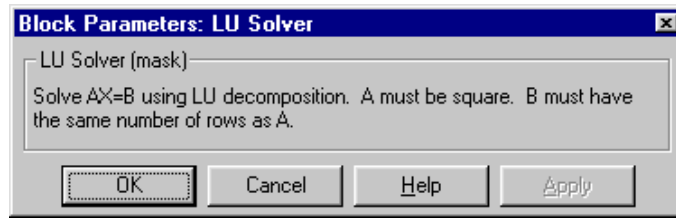
is solved for X by making the substitution Y = UX, and solving two triangular systems.

$$LY = B_p$$
$$UX = Y$$

Examples

See “Example: LU Solver” in the Signal Processing Blockset User’s Guide.

Dialog Box



Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

Autocorrelation LPC	Signal Processing Blockset
Cholesky Solver	Signal Processing Blockset
LDL Solver	Signal Processing Blockset
Levinson-Durbin	Signal Processing Blockset
LU Factorization	Signal Processing Blockset
LU Inverse	Signal Processing Blockset
QR Solver	Signal Processing Blockset

See “Linear System Solvers” for related information.

Purpose

Compute nonparametric estimate of spectrum using periodogram method

Library

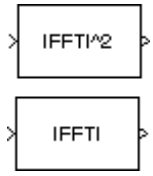
- Estimation / Power Spectrum Estimation

  dspspect3

- Transforms

  dspxfrm3

Description



The Magnitude FFT block computes a nonparametric estimate of the spectrum using the periodogram method.

When the Output parameter is set to Magnitude squared, the block output for an M-by-N input u is equivalent to

```
y = abs(fft(u,nfft)).^2          % M ≤ nfft
```

When the Output parameter is set to Magnitude, the block output for an input u is equivalent to

```
y = abs(fft(u,nfft))             % M ≤ nfft
```

When $M > N_{fft}$, the block wraps the input to N_{fft} before computing the FFT using one of the above equations:

```
y(:,k)=datawrap(u(:,k),nfft)     % 1 ≤ k ≤ N
```

Both an M-by-N frame-based matrix input and an M-by-N sample-based matrix input are treated as M sequential time samples from N independent channels. The block computes a separate estimate for each of the N independent channels and generates an N_{fft}-by-N matrix output. When you select Inherit FFT length from input dimensions, N_{fft} is specified by the frame size of the input, which must be a power of 2. When you do not select Inherit FFT length from input dimensions, N_{fft} is specified as a power of 2 by the FFT length parameter, and the block zero pads or wraps the input to N_{fft} before computing the FFT.

Each column of the output matrix contains the estimate of the corresponding input column's power spectral density at N_{fft} equally spaced frequency points in the range [0, F_s), where F_s is the signal's sample frequency. The output is always sample based.

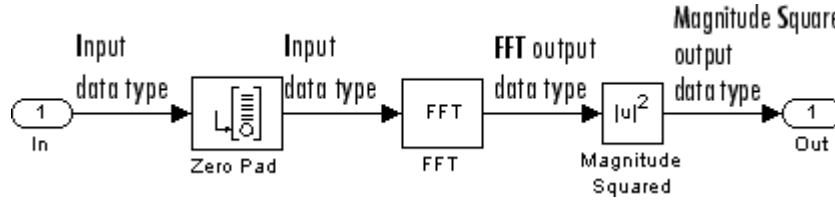
The block does not accept sample-based 1-by-N row vector inputs.

The Magnitude FFT block supports real and complex ßoating-point inputs. The block also supports real Þxed-point inputs in both Magnitude and Magnitude squared modes, and complex Þxed-point inputs in the Magnitude squared mode.

The Magnitude FFT block supports real and complex ßoating-point inputs. The block also supports real Þxed-point inputs in both Magnitude and Magnitude squared modes, and complex Þxed-point inputs in the Magnitude squared mode.

Fixed-Point Data Types

The following diagram shows the data types used within the Magnitude FFT subsystem block for fixed-point signals.



The settings for the fixed-point parameters of the FFT block in the diagram above are as follows:

- Sine table — Same word length as input
- Integer rounding mode — Floor
- Saturate on integer overflow — unchecked
- Product output — Inherit via internal rule
- Accumulator — Inherit via internal rule
- Output — Inherit via internal rule

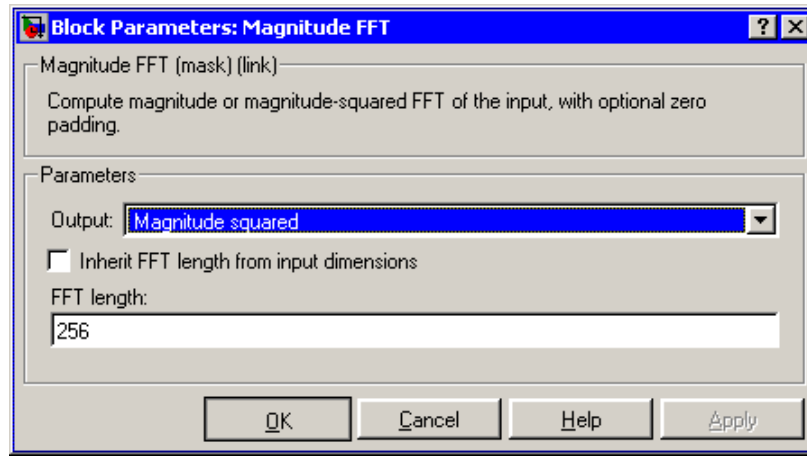
The settings for the fixed-point parameters of the Magnitude Squared block in the diagram above are as follows:

- Integer rounding mode — Floor
- Saturate on integer overflow — checked
- Output — Inherit via internal rule

Examples

The dspsacomp demo compares the periodogram method with several other spectral estimation methods.

Dialog Box



Output

Specify whether the block computes the magnitude FFT or magnitude-squared FFT of the input.

Inherit FFT length from input dimensions

Select to use the input frame size as the number of data points, N_{fft}, on which to perform the FFT.

FFT length

Enter the number of data points on which to perform the FFT, N_{fft}. When N_{fft} is larger than the input frame size, each frame is zero-padded as needed. When N_{fft} is smaller than the input frame size, each frame is wrapped as needed. This parameter is enabled when you clear the Inherit FFT length from input dimensions check box.

References

Oppenheim, A. V. and R. W. Schafer. Discrete-Time Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1989.

Orfanidis, S. J. Introduction to Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1995.

Proakis, J. and D. Manolakis. Digital Signal Processing. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers

See Also

Burg Method	Signal Processing Blockset
Short-Time FFT	Signal Processing Blockset
Spectrum Scope	Signal Processing Blockset
Yule-Walker Method	Signal Processing Blockset
pwelch	Signal Processing Toolbox

See “Power Spectrum Estimatio n” for related information.

Matrix 1-Norm

Purpose

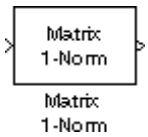
Compute 1-norm of matrix

Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

dspmtrx3

Description

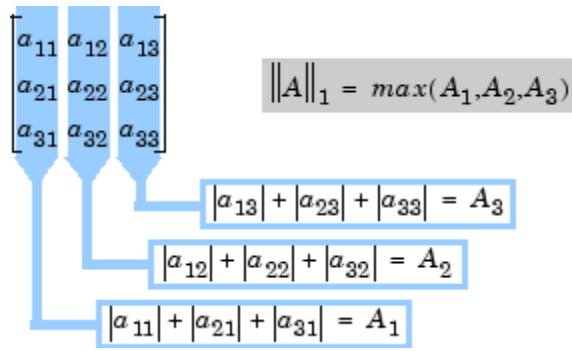


The Matrix 1-Norm block computes the 1-norm, or maximum column-sum, of an M-by-N input matrix, A.

$$y = \|A\|_1 = 1 \le j \le N \max \sum_{i=1}^{M} |a_{ij}|$$

This is equivalent to

```
y = max(sum(abs(A)))          % Equivalent
MATLAB code
```

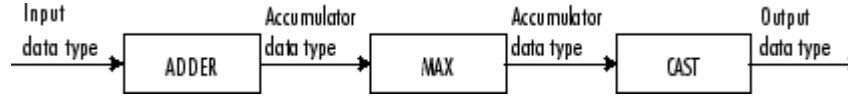


A length- M 1-D vector input is treated as an M-by-1 matrix. The output, y, is always a scalar.

The Matrix 1-Norm block supports real and complex ßoating-point inputs, and real Þxed-point inputs.

Fixed-Point Data Types

The following diagram shows the data types used within the Matrix 1-Norm block for Þxed-point signals.

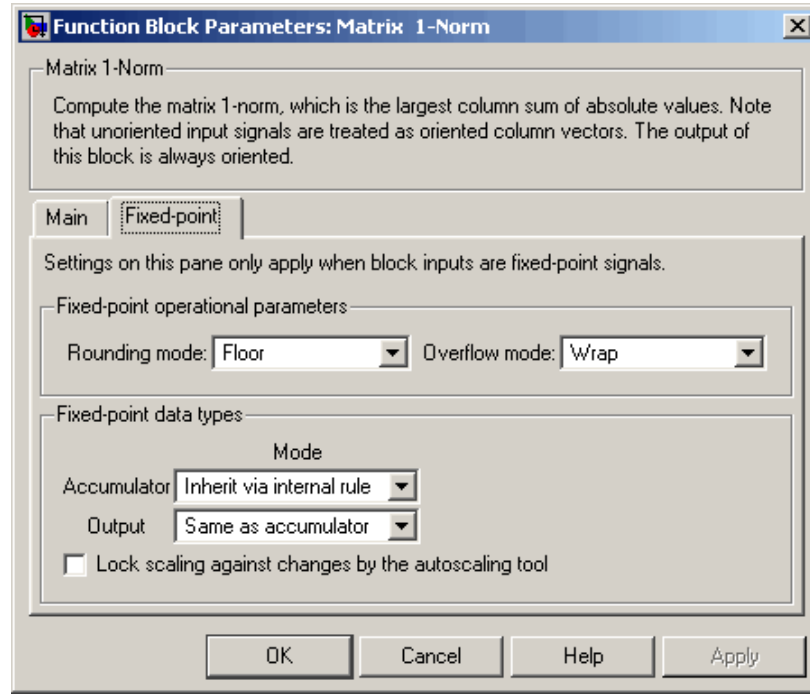


The block calculations are all done in the accumulator data type until the max is performed. The result is then cast to the output data type. You can set the accumulator and outp ut data types in the block dialog as discussed in “Dialog Box” on page 2-827 below.

Dialog Box

There are no parameters on the Main pane of this dialog.

The Fixed-point pane of the Matrix 1-Norm block dialog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Accumulator

Choose how you specify the word length and fraction length of the accumulator:

- When you select `Inherit via internal rule`, the accumulator word length and fraction length are calculated automatically. For information about how the accumulator word and fraction lengths are calculated when an internal rule is used, see "Inherit via Internal Rule".
- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the word length and fraction length of the output of the block:

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Same as accumulator`, these characteristics match those of the accumulator.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

References

Golub, G. H., and C. F. Van Loan. Matrix Computations. 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Normalization	Signal Processing Blockset
Reciprocal Condition	Signal Processing Blockset
norm	MATLAB

Matrix Concatenate

Purpose

Concatenate input signals of same data type to create contiguous output signal

Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

dspmtrx3

Description

The Matrix Concatenate block is an implementation of the Simulink Matrix Concatenate block. See Matrix Concatenate for more information.

Purpose

Compute matrix exponential

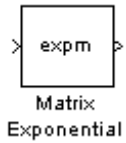
Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

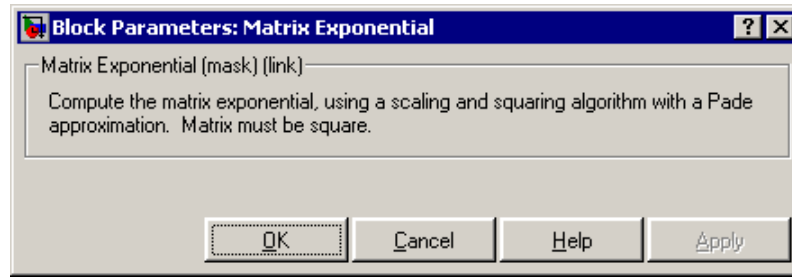
dspmtrx3

Description

The Matrix Exponential block compu tes the matrix exponential using a scaling and squaring algorithm with a Pade approximation. The input matrix must be square.



Dialog Box



Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Array-Vector Multiply	Signal Processing Blockset
expm	MATLAB
Dot Product	Simulink

Matrix Product	Signal Processing Blockset
Product	Simulink

Purpose

Multiply or divide inputs

Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

dspmtrx3

Description

The Matrix Multiply block is an implementation of the Simulink Product block. See Product for more information.

Matrix Product

Purpose

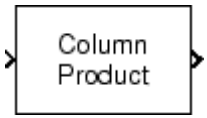
Multiply matrix elements along rows, columns, or entire input

Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

dspmtrx3

Description



The Matrix Product block mul tiplies the elements of an M-by-N input matrix u along its rows, its columns, or over all its elements.

When the Multiply over parameter is set to Rows, the block multiplies across the elements of each r ow and outputs the resulting M-by-1 matrix. A length- N 1-D vector input is treated as a 1-by- N matrix.

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix} \Rightarrow \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \left(\prod_{j=1}^{3} u_{1j} \right) \\ \left(\prod_{j=1}^{3} u_{2j} \right) \\ \left(\prod_{j=1}^{3} u_{3j} \right) \end{bmatrix}$$

When the Multiply over parameter is set to Columns , the block multiplies down the elements of each column and outputs the resulting 1-by-N matrix. A length- M 1-D vector input is treated as a M-by-1 matrix.

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} \left(\prod_{i=1}^{3} u_{i1} \right) & \left(\prod_{i=1}^{3} u_{i2} \right) & \left(\prod_{i=1}^{3} u_{i3} \right) \end{bmatrix}$$

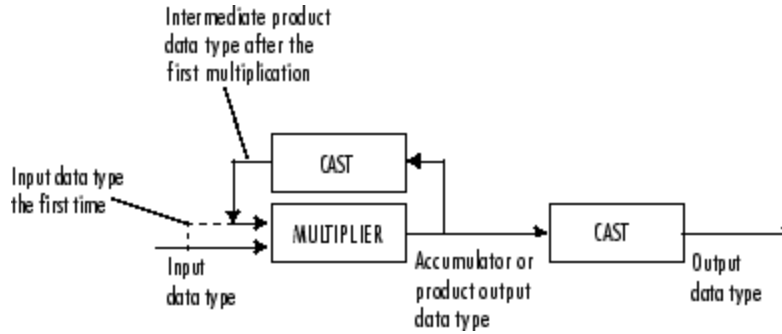
When the Multiply over parameter is set to Entire input, the block multiplies all the elements of the input together and outputs the resulting scalar.

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix} \Rightarrow y = (\prod_{i=1}^{3}\prod_{j=1}^{3} u_{ij})$$

The output of the Matrix Product block has the same frame status as the input. This block accepts real and complex fixed-point and floating-point inputs except for complex unsigned fixed-point inputs.

Fixed-Point Data Types

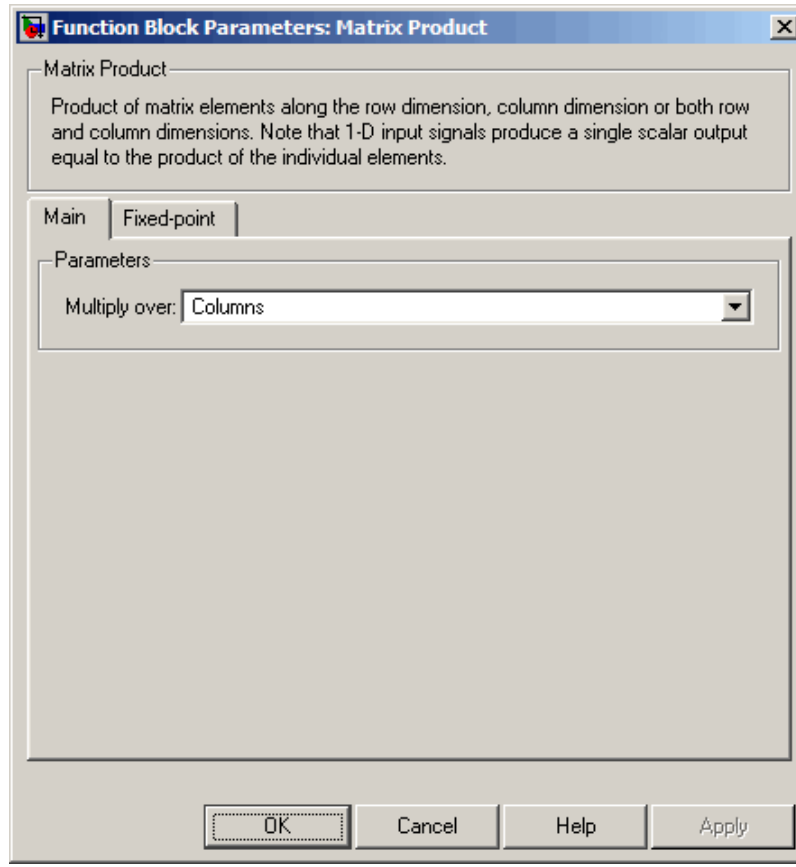
The following diagram shows the data types used within the Matrix Product block for fixed-point signals.



The output of the multiplier is in the product output data type when at least one of the inputs to the multiplier is real. When both of the inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see “Multiplication Data Types”. You can set the accumulator, product output, intermediate product, and output data types in the block dialog as discussed in “Dialog Box” on page 2-836 below.

Dialog Box

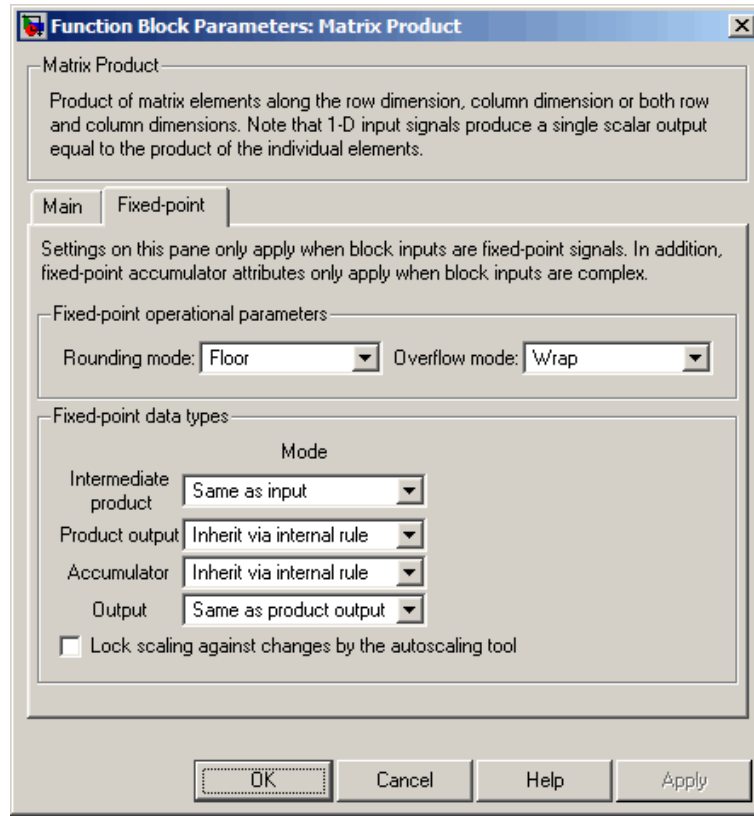
The Main pane of the Matrix Product block dialog appears as follows.



Multiply over

Indicate whether to multiply together the elements of each row, each column, or the entire input.

The Fixed-point pane of the Matrix Product block dialog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Intermediate product

As shown in “Fixed-Point Data Types” on page 2-835, the output of the multiplier is cast to the intermediate product data type before the next element of the input is multiplied into it. Use

this parameter to specify how you would like to designate the intermediate product word and fraction lengths:

- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the intermediate product, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the intermediate product. This block requires power-of-two slope and a bias of zero.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-835 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block:

- When you select Inherit via internal rule , the product output word length and fraction length are calculated automatically. For information about how the product output word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths. See “Fixed-Point

Data Types” on page 2-835 and “Multiplication Data Types” for illustrations depicting the use of the accumulator data type in this block. Note that the accumula tor data type is only used when both inputs to the multiplier are complex:

- When you select Inherit via internal rule , the accumulator word length and fraction length are calculated automatically. For informati on about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the word length and fraction length of the output of the block:

- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Array-Vector Multiply	Signal Processing Blockset
Matrix Square	Signal Processing Blockset
Matrix Sum	Signal Processing Blockset
prod	MATLAB

Purpose

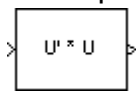
Compute square of input matrix

Library

Math Functions / Matrices and Linear Algebra / Matrix Operations

dspmtrx3

Description



The Matrix Square block computes the square of an M-by-N input matrix, u, by premultiplying with the Hermitian transpose.

```
y = u' * u        % Equivalent MATLAB code
```

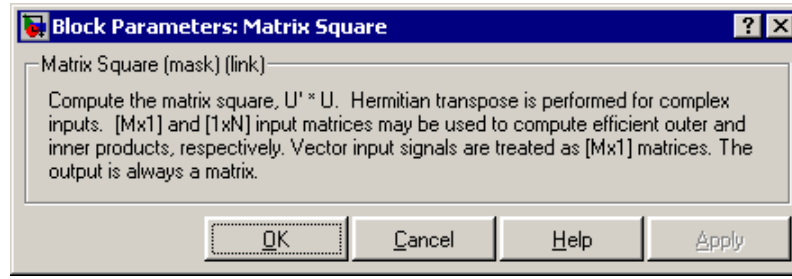
A length-M 1-D vector input is treated as an M-by-1 matrix. For both sample-based and frame-based inputs, output y is sample based with dimension N-by-N.

Applications

The Matrix Square block is useful in a variety of applications:

- General matrix squares — The Matrix Square block computes the output matrix, y, without explicitly forming u'. It is therefore more efficient than other methods for computing the matrix square.
- Sum of squares — When the input is a column vector (N=1), the block's operation is equivalent to a multiply-accumulate (MAC) process, or inner product. The output is the sum of the squares of the input, and is always a real scalar.
- Correlation matrix — When the input is a row vector (M=1), the output, y, is the symmetric autocorrelation matrix, or outer product.

Dialog Box



Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Matrix Multiply	Signal Processing Blockset
Matrix Product	Signal Processing Blockset
Matrix Sum	Signal Processing Blockset
Transpose	Signal Processing Blockset

Purpose

Sum matrix elements along rows, columns, or entire input

Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

dspmtrx3

Description

The Matrix Sum block is an implementation of the Simulink Sum block. See Sum for more information.

Matrix Sum (Obsolete)

Purpose

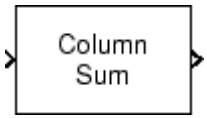
Sum matrix elements along rows, columns, or entire input

Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

dspobslib

Description



The Matrix Sum block sums the elements of an M-by-N input matrix u along its rows, its columns, or over all its elements.

When the Sum over parameter is set to Rows, the block sums across the elements of each row and outputs the resulting M-by-1 matrix. A length- N 1-D vector input is treated as a 1-by- N matrix.

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix} \Rightarrow \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \left(\sum_{j=1}^{3} u_{1j} \right) \\ \left(\sum_{j=1}^{3} u_{2j} \right) \\ \left(\sum_{j=1}^{3} u_{3j} \right) \end{bmatrix}$$

When the Sum over parameter is set to Columns , the block sums down the elements of each column and outputs the resulting 1-by- N matrix. A length- M 1-D vector input is treated as a M-by-1 matrix.

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} \left(\sum_{i=1}^{3} u_{i1} \right) & \left(\sum_{i=1}^{3} u_{i2} \right) & \left(\sum_{i=1}^{3} u_{i3} \right) \end{bmatrix}$$

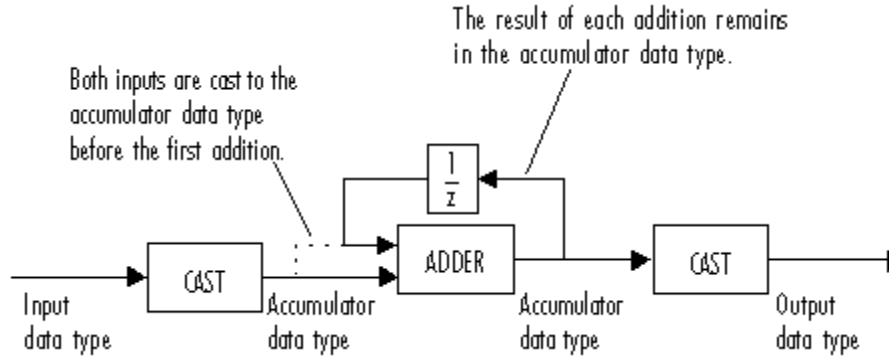
When the Sum over parameter is set to Entire input, the block sums all the elements of the input together and outputs the resulting scalar.

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix} \Rightarrow y = (\sum_{i=1}^{3}\sum_{j=1}^{3} u_{ij})$$

The output of the Matrix Sum block has the same frame status as the input. This block accepts real and complex fixed-point and floating-point inputs except for complex unsigned fixed-point inputs.

Fixed-Point Data Types

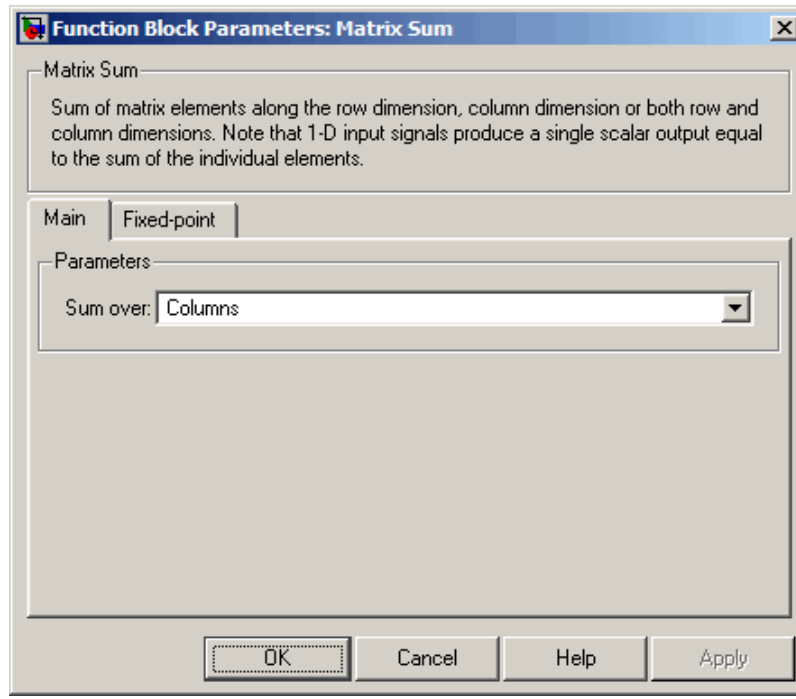
The following diagram shows the data types used within the Matrix Sum block for fixed-point signals.



You can set the accumulator and output data types in the block dialog as discussed in “Dialog Box” on page 2-846 below.

Dialog Box

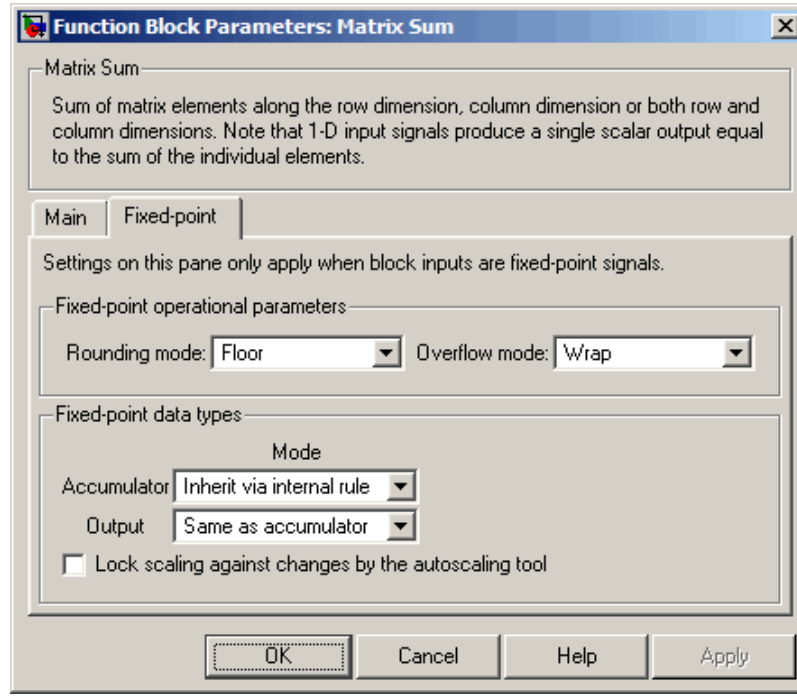
The Main pane of the Matrix Sum block dialog appears as follows.



Sum over

Indicate whether to sum the elements of each row, each column, or of the entire input.

The Fixed-point pane of the Matrix Sum block dialog appears as follows.



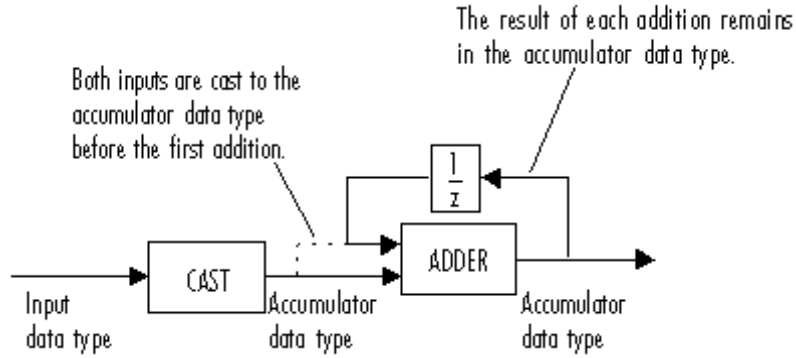
Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Accumulator



As depicted above, the elements of the block input are cast to the accumulator data type befor e they are added together. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how you would like to designate this accumulator word and fraction lengths:

- When you select Inherit via internal rule , the accumulator word length and fraction length are calculated automatically. For informati on about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as accumulator , these characteristics match those of the accumulator.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any Þxed-point scaling you specify in this block mask f rom being overridden by the autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Matrix Product	Signal Processing Blockset
Matrix Multiply	Signal Processing Blockset
sum	MATLAB

Purpose

Display matrices as color images

Library

Signal Processing Sinks

dspsnks4

Description

The Matrix Viewer block displays an M-by-N matrix input by mapping the matrix element values to a specified range of colors. The display is updated as each new input is received. This block treats a length M 1-D vector input as an M-by-1 matrix.

Image Properties

Select the Image Properties tab to show the image property parameters, which control the colormap and display.

You specify the mapping of matrix element values to colors in the Colormap matrix, Minimum input value, and Maximum input value parameters. For a colormap with L colors, the colormap matrix has dimension L-by-3, with one row for each color and one column for each element of the RGB triple that defines the color. Examples of RGB triples are

```
[ 1    0    0 ]   (red)
[ 0    0    1 ]   (blue)
[0.8  0.8  0.8]   (light gray)
```

See the ColorSpec property in the MATLAB documentation for complete information about defining RGB triples.

MATLAB provides a number of functions for generating predefined colormaps, such as hot, cool, bone, and autumn. Each of these functions accepts the colormap size as an argument, and can be used in the Colormap matrix parameter. For example, when you specify gray(128) for the Colormap matrix parameter, the matrix is displayed in 128 shades of gray. The color in the first row of the colormap matrix represents the value specified by the Minimum input value parameter, and the color in the last row represents the value specified by the Maximum input value parameter. Values

between the minimum and maximum a re quantized and mapped to the intermediate rows of the colormap matrix.

The documentation for the MATLAB colormap function provides complete information about specifying colormap matrices, and includes a complete list of the available colormap functions.

Axis Properties

Select the Axis Properties tab to show the axis property parameters, which control labeling and positioning.

The Axis origin parameter determines where the Þrst element of the input matrix, U(1,1), is displayed. When you specify Upper left corner, the matrix is displayed in matrix orientation, with U(1,1) in the upper-left corner.

$$\begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} \\ U_{21} & U_{22} & U_{23} & U_{24} \\ U_{31} & U_{32} & U_{33} & U_{34} \\ U_{41} & U_{42} & U_{43} & U_{44} \end{bmatrix}$$

When you specify Lower left corner, the matrix is ßipped vertically to image orientation, with U(1,1) in the lower-left corner.

$$\begin{bmatrix} U_{41} & U_{42} & U_{43} & U_{44} \\ U_{31} & U_{32} & U_{33} & U_{34} \\ U_{21} & U_{22} & U_{23} & U_{24} \\ U_{11} & U_{12} & U_{13} & U_{14} \end{bmatrix}$$

Axis zoom, when selected, causes the image display to completely Þll the Þgure window. Axis titles are not displayed. This option can also be selected from the pop-up menu that is displayed when you right-click in the Þgure window. When Axis zoom is cleared, the axis labels and titles are displayed in a gray border surrounding the image axes.

Figure Window

The image title in the figure title bar is the same as the block title. The axis tick marks reflect the size of the input matrix; the x-axis is numbered from 1 to N (number of columns), and the y-axis is numbered from 1 to M (number of rows).

Right-click the image in the figure window to access the following menu items:

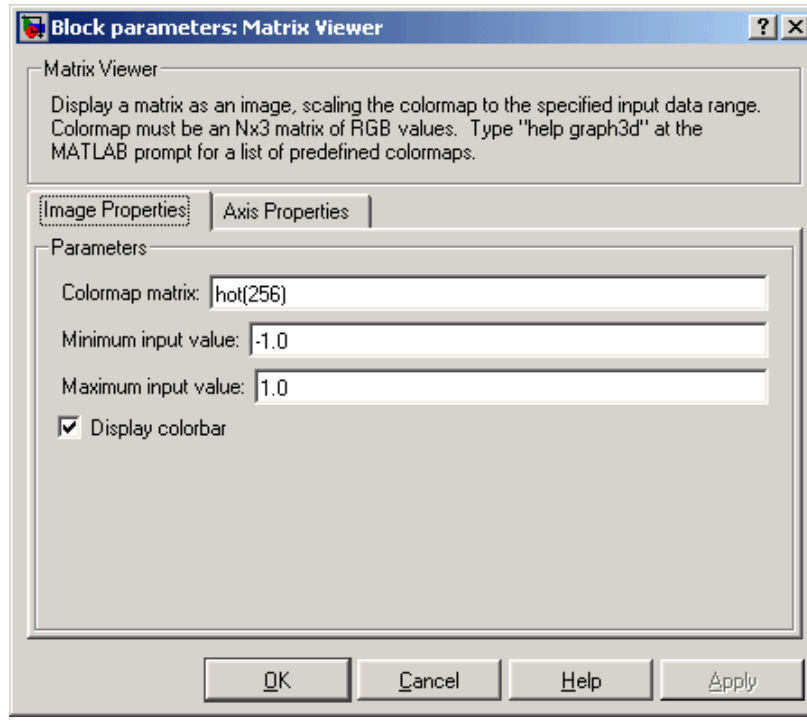
- Refresh erases all data on the scope display except for the most recent image.
- Autoscale recomputes the minimum and maximum input values to fit the range of values observed in a series of 10 consecutive inputs. The numerical limits selected by the autoscale feature are shown in the Minimum input value and Maximum input value parameters, where you can make further adjustments to them manually.
- Axis zoom, when selected, causes the image to completely fill the figure window. Axis titles are not displayed. When Axis zoom is cleared, the axis labels and titles are displayed in a gray border surrounding the scope axes. This option can also be set in the Axis Properties pane of the parameter dialog.
- Colorbar, when selected, displays a bar with the specified colormap to the right of the image axes.
- Save Position automatically updates the Figure position parameter in the Axis Properties pane to reflect the figure window's current position and size on the screen. To make the scope window open at a particular location on the screen when the simulation runs, drag the window to the desired location, resize it, and select Save Position. The parameter dialog must be closed when you select Save Position for the Figure position parameter to be updated.

Examples

See the demo dspstfft.mdl for an example of using the Matrix Viewer block to create a moving spectrogram, or time-frequency plot, of a

speech signal by updating just one column of the input matrix at each sample time.

Dialog Box



Colormap matrix

A 3-column matrix de Þning the colormap as a set of RGB triples, or a call to a colormap-generating function such as hot or spring . See the ColorSpec property for complete information about deÞning RGB triples, and the MATLAB colormap function for a list of colormap-generat ing functions. Tunable.

Minimum input value

The input value to be mapped to the color de Þned in the Þrst row of the colormap matrix. Right-click in the Þgure window and select Autoscale from pop-up menu to set this parameter to the

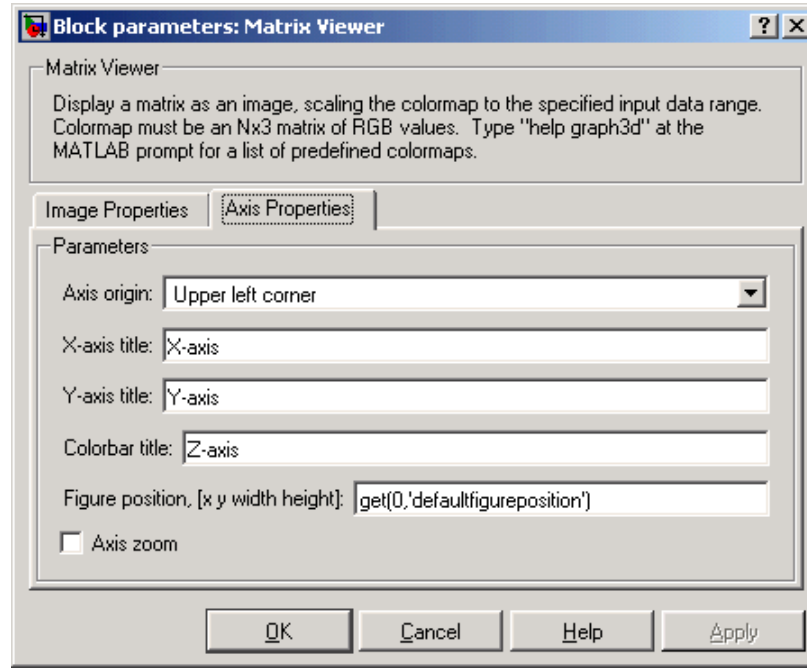
minimum value observed in a series of 10 consecutive matrix inputs. Tunable.

Maximum input value

The input value to be mapped to the color de Þned in the last row of the colormap matrix. Right-click in the Þgure window and select Autoscale from the pop-up menu to set this parameter to the maximum value observed in a series of 10 consecutive matrix inputs. Tunable.

Display colorbar

Select to display a bar with the se lected colormap to the right of the image axes. Tunable.



Axis origin
: The position within the axes where the first element of the input matrix, U(1,1), is plotted; bottom left or top left. Tunable.

X-axis title
: The text to be displayed below the x-axis. Tunable.

Y-axis title
: The text to be displayed to the left of the y-axis. Tunable.

Colorbar title
: The text to be displayed to the right of the color bar, when Display colorbar is currently selected. Tunable.

Figure position, [x y width height]
: A 4-element vector of the form [x y width height] specifying the position of the figure window, where (0,0) is the lower-left corner of the display. Tunable.

Axis zoom
: Resizes the image to fill the figure window. Tunable.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Spectrum Scope	Signal Processing Blockset
Vector Scope	Signal Processing Blockset
colormap	MATLAB

ColorSpec	MATLAB
image	MATLAB

Purpose

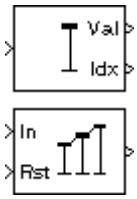
Find maximum values in input or sequence of inputs

Library

Statistics

dspstat3

Description



The Maximum block identifies the value and/or position of the largest element in each row or column of the input, along vectors of a specified dimension of the input, or of the entire input. The Maximum block can also track the maximum values in a sequence of inputs over a period of time. The **Mode** parameter specifies the block's mode of operation and can be set to `Value`, `Index`, `Value and Index`, or `Running`.

The Maximum block supports real and complex floating-point, fixed-point, and Boolean inputs. Real fixed-point inputs can be either signed or unsigned, while complex fixed-point inputs must be signed. The data type of the maximum values output by the block match the data type of the input. The index values output by the block are `double` when the input is `double`, and `uint32` otherwise.

The frame status of the block output is the same as that of the input, except when the **Find the maximum value of** parameter is set to `Entire input`. The output is always sample based when `Entire input` is selected.

For the `Value`, `Index`, and `Value and Index` modes, the Maximum block produces identical results as the MATLAB `max` function when it is called as `[y I] = max(u,[],D)`, where `u` and `y` are the input and output, respectively, `D` is the dimension, and `I` is the index.

Value Mode

When the **Mode** parameter is set to `Value`, the block computes the maximum value in each row or column of the input, along vectors of a specified dimension of the input, or of the entire input at each sample time, and outputs the array `y`. Each element in `y` is the maximum value in the corresponding column, row, vector, or entire input. The output `y` depends on the setting of the **Find the maximum value**

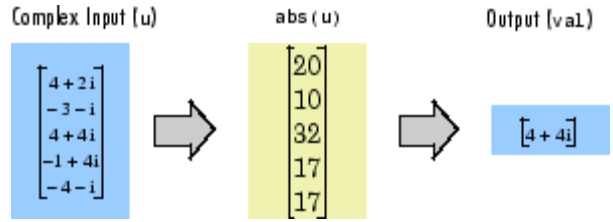
over parameter. For example, consider a 3-dimensional input signal of size M-by-N-by-P:

- Each row — The output at each sample time consists of an M-by-1-by-P array, where each element contains the maximum value of each vector over the second dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is an M-by-1 column vector. In this mode, the frame status of the output is the same as that of the input.
- Each column — The output at each sample time consists of a 1-by-N-by-P array, where each element contains the maximum value of each vector over the first dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is a 1-by-N row vector. In this mode, the frame status of the output is the same as that of the input.

  For convenience, length-M 1-D vector inputs are treated as M-by-1 column vectors when the block is in this mode. Sample-based length-M row vector inputs are also treated as M-by-1 column vectors when the Treat sample-based row input as a column check box is selected.
- Entire input — The output at each sample time is a scalar that contains the maximum value in the M-by-N-by-P input matrix. In this mode, the block output is always sample based.
- Specified dimension — The output at each sample time depends on Dimension. If Dimension is set to 1, the output is the same as that when you select Each column. If Dimension is set to 2, the output is the same as when you select Each row. If Dimension is set to 3, the output at each sample time is an M-by-N matrix containing the maximum value of each vector over the third dimension of the input. In this mode, the frame status of the output is the same as that of the input.

For complex inputs, the block selects the value in each row or column of the input, along vectors of a specified dimension of the input, or of

the entire input that has the maximum magnitude squared as shown below. For complex value $u = a + bi$, the magnitude squared is $a^2 + b^2$.



Index Mode

When Mode is set to Index, the block computes the maximum value in each row or column of the input, along vectors of a specified dimension of the input, or of the entire input, and outputs the index array I. Each element in I is an integer indexing the maximum value in the corresponding column, row, vector, or entire input. The output I depends on the setting of the Find the maximum value over parameter. For example, consider a 3-dimensional input signal of size M-by-N-by-P:

- Each row — The output at each sample time consists of an M-by-1-by-P array, where each element contains the index of the maximum value of each vector over the second dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is an M-by-1 column vector. In this mode, the frame status of the output is the same as that of the input.
- Each column — The output at each sample time consists of a 1-by-N-by-P array, where each element contains the index of the maximum value of each vector over the first dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is a 1-by-N row vector. In this mode, the frame status of the output is the same as that of the input.

  For convenience, length-M 1-D vector inputs are treated as M-by-1 column vectors when the block is in this mode. Sample-based length-M row vector inputs are also treated as M-by-1 column vectors

when the Treat sample-based row input as a column check box is selected.

- Entire input — The output at each sample time is a 1-by-3 vector that contains the location of the maximum value in the M-by-N-by-P input matrix. In this mode, the block output is always sample based. For an input that is an M-by-N matrix, the output will be a 1-by-2 vector.
- Specified dimension — The output at each sample time depends on Dimension. If Dimension is set to 1, the output is the same as when you select Each column. If Dimension is set to 2, the output is the same as when you select Each row. If Dimension is set to 3, the output at each sample time is an M-by-N matrix containing the indices of the maximum values of each vector over the third dimension of the input. In this mode, the frame status of the output is the same as that of the input.

When a maximum value occurs more than once, the computed index corresponds to the first occurrence. For example, when the input is the column vector [3 2 1 2 3]', the computed one-based index of the maximum value is 1 rather than 5 when Each column is selected.

When inputs to the block are double-precision values, the index values are double-precision values. Otherwise, the index values are 32-bit unsigned integer values.

Value and Index Mode

When Mode is set to Value and Index, the block outputs both the maxima and the indices.

Running Mode

When Mode is set to Running, the block tracks the maximum value of each channel in a time sequence of M-by-N inputs. For sample-based inputs, the output is a sample-based M-by-N array with each element y_{ij} containing the maximum value observed in element u_{ij} for all inputs since the last reset. For frame-based inputs, the output is a frame-based M-by-N matrix with each element y_{ij} containing the maximum value

observed in the jth column of all inputs since the last reset, up to and including element u_{ij} of the current input.

N-D signals cannot be frame based. When the block is set to Running mode, each element of the N-D signal is treated as a separate channel.

There are $\prod d_i$ channels, where d_i is the size of the ith dimension.

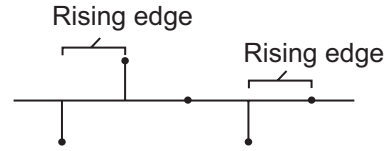
Resetting the Running Maximum

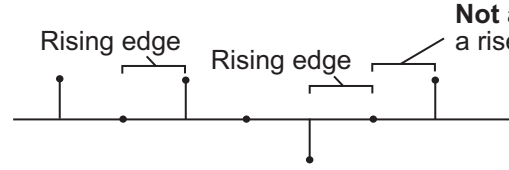
The block resets the running maximum whenever a reset event is detected at the optional Rst port. The reset sample time must be a positive integer multiple of the input sample time.

For sample-based inputs, a reset event causes the running maximum for each channel to be initialized to the value in the corresponding channel of the current input. For frame-based inputs, a reset event causes the running maximum for each channel to be initialized to the earliest value in each channel of the current input.

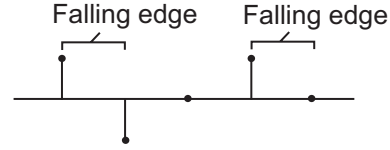
You specify the reset event in the Reset port menu:

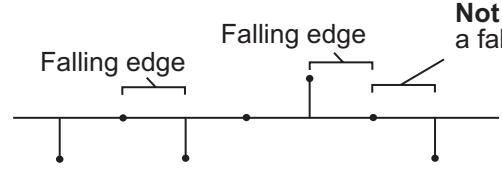
- None — Disables the Rst port.
- Rising edge — Triggers a reset operation when the Rst input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following figure)





- Falling edge — Triggers a reset operation when the Rst input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)





- Either edge — Triggers a reset o peration when the Rst input is a Rising edge or Falling edge (as described above)
- Non-zero sample — Triggers a reset opera tion at each sample time that the Rst input is not zero

Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see "Excess Algorithmic Delay (Tasking Latency)" and "Models with Multiple Sample Rates" in the Real-Time Workshop User's Guide.

ROI Processing

To calculate the statistical value within a particular region of interest (ROI) of the input, select the Enable ROI processing check box. This option is only available when the Find the maximum value over parameter is set to Entire input and the Enable ROI processing check box is selected. ROI processing is only supported for 2-D inputs.

Note Full ROI processing is only available to users who have a Video and Image Processing Blockset license. If you only have a Signal Processing Blockset license, you can still use ROI processing, but are limited to the ROI type Rectangles.

Use the ROI type parameter to specify whether the ROI is a rectangle, line, label matrix, or binary mask. A binary mask is a binary image that enables you to specify which pixels to highlight, or select. In a label matrix, pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on. When the ROI type parameter is set to Label matrix, the Label and Label Numbers ports appear on the block. Use the Label Numbers port to specify the objects in the label matrix for which the block calculates statistics. The input to this port must be a vector of scalar values that correspond to the labeled regions in the label matrix. For more information about the format of the input to the ROI port when the ROI is a rectangle or a line, see the Draw Shapes block reference page.

For rectangular ROIs, use the ROI portion to process parameter to specify whether to calculate the statistical value for the entire ROI or just the ROI perimeter.

Use the Output parameter to specify the block output. The block can output separate statistical values for each ROI or the statistical value for all specified ROIs. This parameter is not available if, for the ROI type parameter, you select Binary mask.

If, for the ROI type parameter, you select Rectangles or Lines, the Output flag indicating if ROI is within image bounds check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

Output = Individual statistics for each ROI

Flag Port Output	Description
0	ROI is completely outside the input image.
1	ROI is completely or partially inside the input image.

Output = Single statistic for all ROIs

Flag Port Output	Description
0	All ROIs are completely outside the input image.
1	At least one ROI is completely or partially inside the input image.

If the ROI is partially outside the image, the block only computes the statistical values for the portion of the ROI that is within the image.

If, for the ROI type parameter, you select Label matrix, the Output flag indicating if input label numbers are valid check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

Output = Individual statistics for each ROI

Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

Output = Single statistic for all ROIs

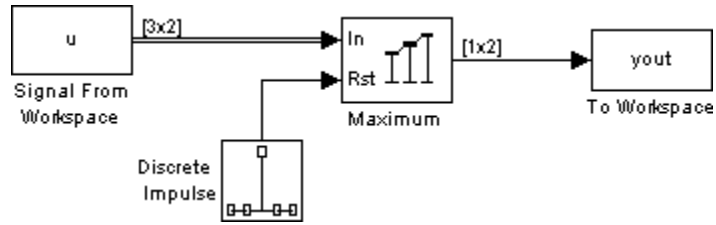
Flag Port Output	Description
0	None of the label numbers are in the label matrix.
1	At least one of the label numbers is in the label matrix.

Fixed-Point Data Types

The parameters on the Fixed-point pane of the block dialog are only used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before a comparison is made, as described in “Value Mode” on page 2-858. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into the accumulator data type. These parameters are ignored for other types of inputs.

Examples

The Maximum block in the following model calculates the running maximum of a frame-based 3-by- 2 (two-channel) matrix input, u. The running maximum is reset at t=2 by an impulse to the block's Rst port.



The Maximum block has the following settings:

- Mode = Running
- Reset port = Non-zero signal

The Signal From Workspace block has the following settings:

- Signal = u
- Sample time = 1/3
- Samples per frame = 3

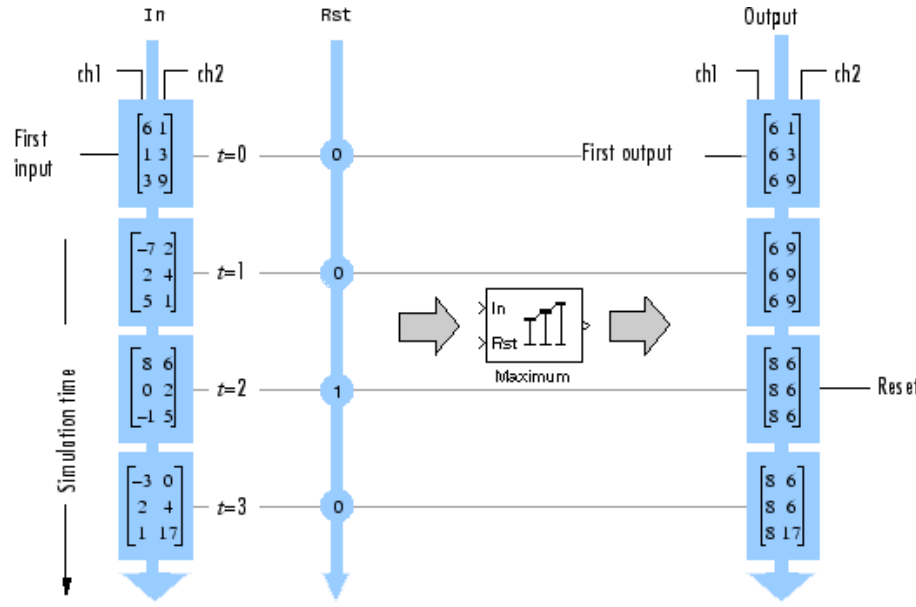
where

```
u
= [6 1 3 -7 2 5 8 0 -1 -3 2 1;1 3 9 2 4 1 6 2 5 0 4 17]'
```

The Discrete Impulse block has the following settings:

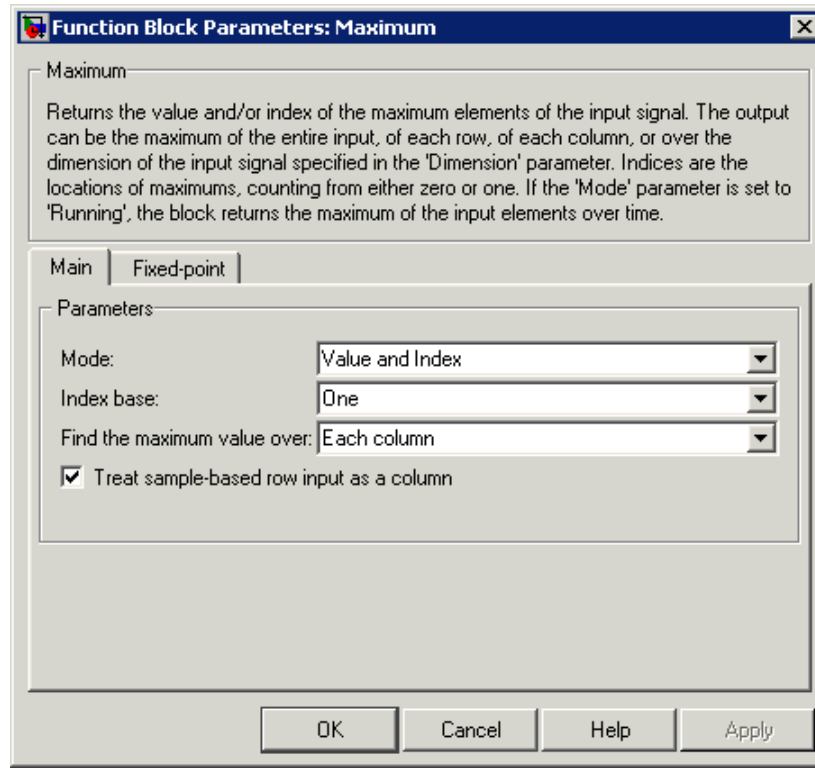
- Delay (samples) = 2
- Sample time = 1
- Samples per frame = 1

The block's operation is shown in the figure below.


In
Rst
Output
ch1
ch2
ch1
ch2
First input
t=0
t=1
t=2
t=3
0
0
1
0
First output
In
Rst
Maximum
Reset
Simulation time

Dialog Box

The Main pane of the Maximum block dialog appears as follows.



Mode

Specify the block's mode of operation:

- Value — Output the maximum value of each input
- Index — Output the index of the maximum value
- Value and index — Output both the value and the index
- Running — Track the maximum value of the input sequence over time

For more information, see Description.

Index base

Specify whether the index of the m aximum value is reported using one-based or zero-based numbering. This parameter is only visible when the Mode parameter is set to Index or Value and index .

Find the maximum value over

Specify whether to Þnd the maximum value along rows, columns, entire input, or the dimension speci Þed in the Dimension parameter. For more information, see Description.

Treat sample-based row input as a column

Select to treat sample-based length- M row vector inputs as M-by-1 column vectors. This parameter is only visible when the Find the maximum value of parameter is set to Each column .

Reset port

Specify the reset event detected at the Rst input port when you select Running for the Mode parameter. The rate of the reset signal must be a positive integer multiple of the rate of the data signal input. For information ab out the possible values of this parameter, see “Resetting the R unning Maximum” on page 2-862.

Dimension

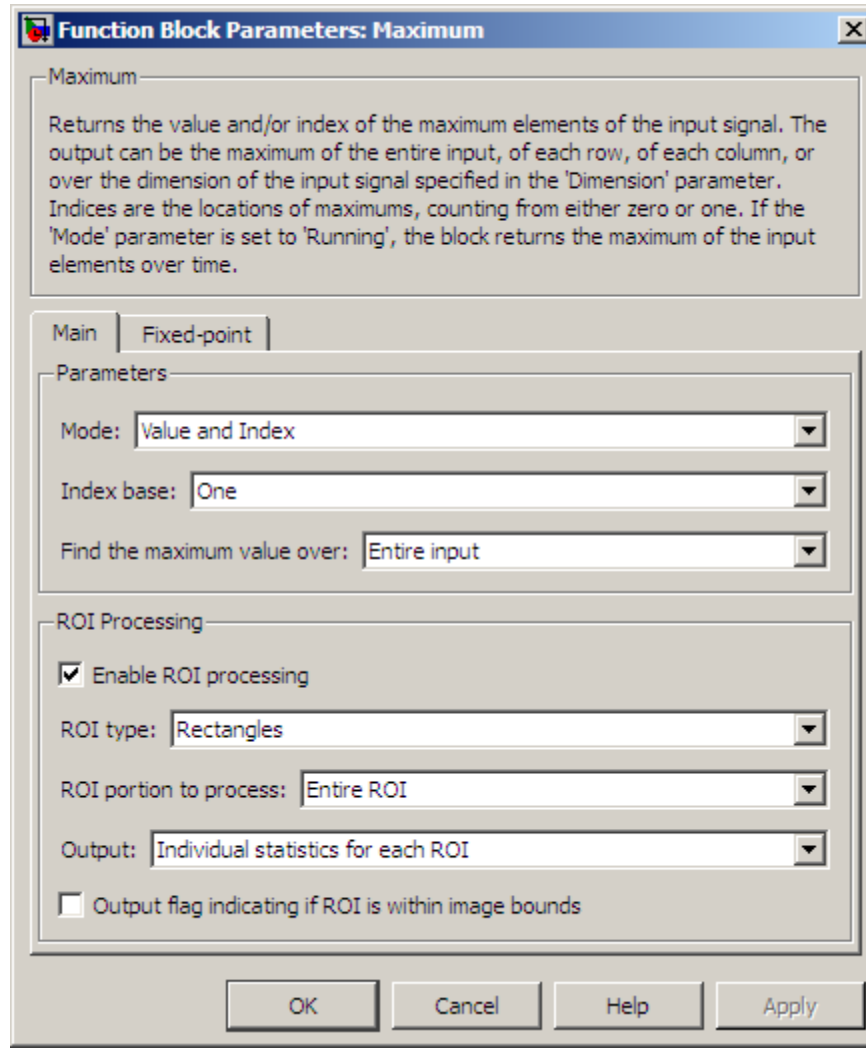
Specify the dimension (one-based value) of the input signal, over which the maximum is computed. The value of this parameter cannot exceed the number of dimensions in the input signal. This parameter is only visible when the Find the maximum value over parameter is set to Specified dimension .

Enable ROI processing

Select this check box to calculat e the statistical value within a particular region of each imag e. This parameter appears only when you set the Find the maximum value over parameter to Entire input , and the block is not in running mode.

Note Full ROI processing is only available to users who have a Video and Image Processing Blockset license. If you only have a Signal Processing Blockset license, you can still use ROI processing, but are limited to the ROI type Rectangles .

The ROI processing parameters a ppear on the dialog box as follows.



ROI type

Specify the type of ROI you want to use. Your choices are Rectangles , Lines , Label matrix , or Binary mask .

ROI portion to process

Specify whether you want to calculate the statistical value for the entire ROI or just the ROI perimeter. This parameter appears only if you specify an ROI type of Rectangles.

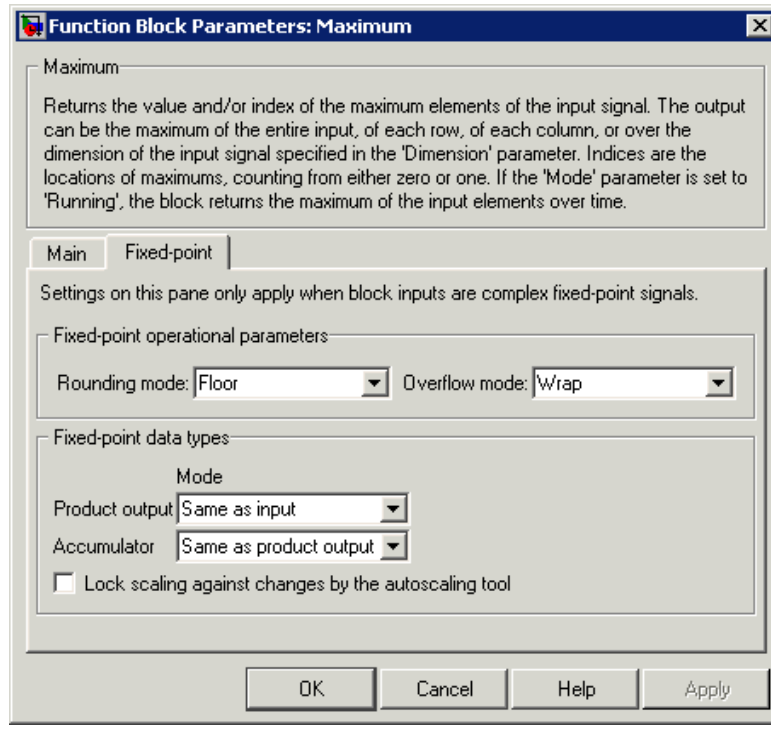
Output

Specify the block output. The block can output a vector of separate statistical values for each ROI or a scalar value that represents the statistical value for all the specified ROIs. This parameter is not available if, for the ROI type parameter, you select Binary mask.

Output flag indicating if ROI is within image bounds

If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in "ROI Processing" on page 2-907. This parameter appears if you select an ROI type of Rectangles or Lines.

The Fixed-point pane of the Maximum block dialog appears as follows.



Note The parameters on the Fixed-point pane are only used for complex Þxed-point inputs. The sum of the squares of the real and imagina ry parts of such an input are f ormed before a comparison is made, as described in “Value Mode” on page 2-858. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into the accumulator data type. These parameters are ignored for other types of inputs.

Round ing mode

Select the rounding mode for Þxed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths resulting from a complex-complex multiplication in the block. See “Multiplication Data Types” for more information:

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block. See “Multiplication Data Types” for more information:

- When you select Same as product output, these characteristics match those of the product output.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the

autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Reset	• Double-precision ßoating point • Single-precision ßoating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Idx	• Double-precision ßoating point • 32-bit unsigned integers
Val	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Mean	Signal Processing Blockset
Minimum	Signal Processing Blockset
MinMax	Simulink
max	MATLAB

Purpose

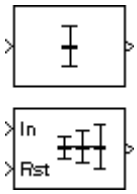
Find mean value of input or sequence of inputs

Library

Statistics

dspstat3

Description



The Mean block computes the mean of each row or column of the input, along vectors of a specified dimension of the input, or of the entire input. The Mean block can also track the mean value in a sequence of inputs over a period of time. The Running mean parameter selects between basic operation and running operation.

The Mean block accepts real and complex fixed-point and floating-point inputs.

Basic Operation

When you do not select the Running mean check box, the block computes the mean value in each row or column of the input, along vectors of a specified dimension of the input, or of the entire input at each individual sample time. Each element in the output array y is the mean value of the corresponding column, row, vector, or entire input. The output array y depends on the setting of the Find the mean value over parameter. For example, consider a 3-dimensional input signal of size M-by-N-by-P:

- Entire input — The output at each sample time is a scalar that contains the mean value of the M-by-N-by-P input matrix. In this mode, the block output is always sample based.

  ```
  y = mean(u(:))        % Equivalent MATLAB code
  ```

- Each row — The output at each sample time consists of an M-by-1-by-P array, where each element contains the mean value of each vector over the second dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is an M-by-1 column vector. In this mode, the frame status of the output is the same as that of the input.

```
y = mean(u,2)        % Equivalent MATLAB code
```

- Each column — The output at each sample time consists of a 1-by-N-by-P array, where each element contains the mean value of each vector over the first dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is a 1-by-N row vector. In this mode, the frame status of the output is the same as that of the input.

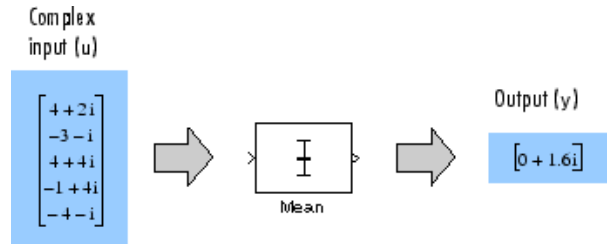
  ```
  y = mean(u)        % Equivalent MATLAB code
  ```

  For convenience, length-M 1-D vector inputs are treated as M-by-1 column vectors when the block is in this mode. Sample-based length-M row vector inputs are also treated as M-by-1 column vectors when the Treat sample-based row input as a column check box is selected.

- Specified dimension — The output at each sample time depends on Dimension. If Dimension is set to 1, the output is the same as when you select Each column. If Dimension is set to 2, the output is the same as when you select Each row. If Dimension is set to 3, the output at each sample time is an M-by-N matrix containing the mean value of each vector over the third dimension of the input. In this mode, the frame status of the output is the same as that of the input.

  ```
  y = mean(u,Dimension)        % Equivalent MATLAB code
  ```

The mean of a complex input is computed independently for the real and imaginary components, as shown in the next figure.



Running Operation

When the Running mean check box is selected, the block tracks the mean value of each channel in a time sequence of inputs. For sample-based M-by-N inputs, the output is a sample-based M-by-N array with each element y_{ij} containing the mean value of the elements u_{ij} for all inputs since the last reset. For frame-based M-by-N inputs, the output is a frame-based M-by-N matrix with each element y_{ij} containing the mean of the values in the jth column of all inputs since the last reset, up to and including element u_{ij} of the current input.

N-D signals cannot be frame based. When the block is set to Running mode, each element of the N-D signal is treated as a separate channel.

There are $\prod d_i$ channels, where d_i is the size of the ith dimension.

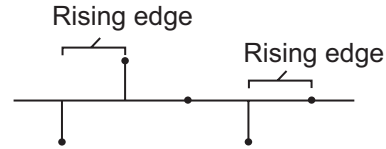
Resetting the Running Mean

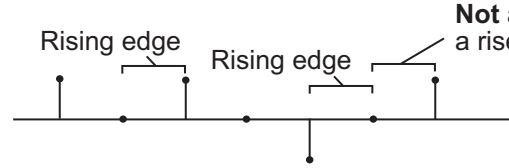
The block resets the running mean whenever a reset event is detected at the optional Rst port. The reset sample time must be a positive integer multiple of the input sample time.

When the block is reset for sample-based inputs, the running mean for each channel is initialized to the value in the corresponding channel of the current input. For frame-based inputs, the running mean for each channel is initialized to the earliest value in each channel of the current input.

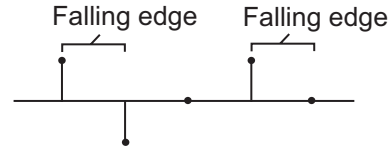
You specify the reset event by the Reset port parameter:

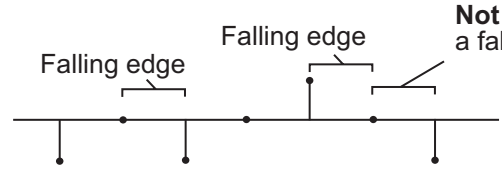
- None disables the Rst port.
- Rising edge — Triggers a reset operation when the Rst input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following figure)





- Falling edge — Triggers a reset operation when the Rst input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)





- Either edge — Triggers a reset o peration when the Rst input is a Rising edge or Falling edge (as described earlier)
- Non-zero sample — Triggers a reset opera tion at each sample time that the Rst input is not zero

---

Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see "Excess Algorithmic Delay (Tasking Latency)" and "Models with Multiple Sample Rates" in the Real-Time Workshop User's Guide.

ROI Processing

To calculate the statistical value within a particular region of interest (ROI) of the input, select the Enable ROI processing check box. This option is only available when the Find the mean value over parameter is set to Entire input and the Running mean check box is not selected. ROI processing is only supported for 2-D inputs.

Note Full ROI processing is only available to users who have a Video and Image Processing Blockset license. If you only have a Signal Processing Blockset license, you can still use ROI processing, but are limited to the ROI type Rectangles.

Use the ROI type parameter to specify whether the ROI is a rectangle, line, label matrix, or binary mask. A binary mask is a binary image that enables you to specify which pixels to highlight, or select. In a label matrix, pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on. When the ROI type parameter is set to Label matrix, the Label and Label Numbers ports appear on the block. Use the Label Numbers port to specify the objects in the label matrix for which the block calculates statistics. The input to this port must be a vector of scalar values that correspond to the labeled regions in the label matrix. For more information about the format of the input to the ROI port when the ROI is a rectangle or a line, see the Draw Shapes block reference page.

For rectangular ROIs, use the ROI portion to process parameter to specify whether to calculate the statistical value for the entire ROI or just the ROI perimeter.

Use the Output parameter to specify the block output. The block can output separate statistical values for each ROI or the statistical value for all specified ROIs. This parameter is not available if, for the ROI type parameter, you select Binary mask .

If, for the ROI type parameter, you select Rectangles or Lines , the Output flag indicating if ROI is within image bounds check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

Output = Individual statistics for each ROI

Flag Port Output	Description
0	ROI is completely outside the input image.
1	ROI is completely or partially inside the input image.

Output = Single statistic for all ROIs

Flag Port Output	Description
0	All ROIs are completely outside the input image.
1	At least one ROI is completely or partially inside the input image.

If the ROI is partially outside the image, the block only computes the statistical values for the portion of the ROI that is within the image.

If, for the ROI type parameter, you select Label matrix, the Output flag indicating if input label numbers are valid check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

Output = Individual statistics for each ROI

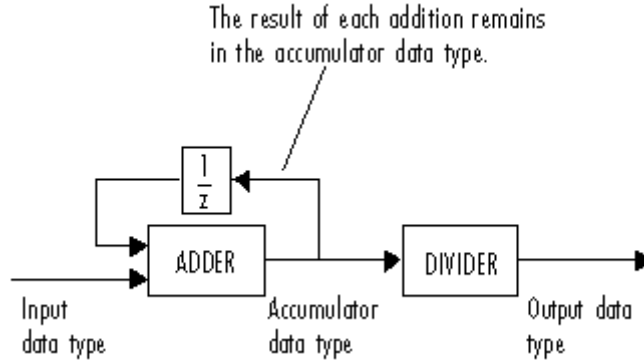
Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

Output = Single statistic for all ROIs

Flag Port Output	Description
0	None of the label numbers are in the label matrix.
1	At least one of the label numbers is in the label matrix.

Fixed-Point Data Types

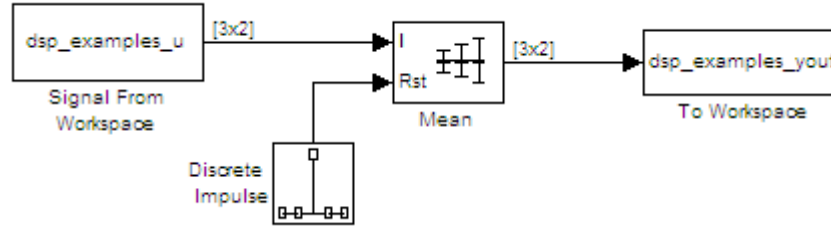
The following diagram shows the data types used within the Mean block for fixed-point signals.



You can set the accumulator and outp ut data types in the block dialog, as discussed in “Dialog Box” on page 2-887.

Examples

The Mean block in the following model calculates the running mean of a frame-based 3-by-2 (two-channel) matrix input, u. The running mean is reset at t=2 by an impulse to the block’s Rst port.



The Mean block has the following settings:

- Running mean = Select this check box
- Reset port = Non-zero sample

The Signal From Workspace bloc k has the follo wing settings:

- Signal = dsp_examples_u

- Sample time = 1/3
- Samples per frame = 3

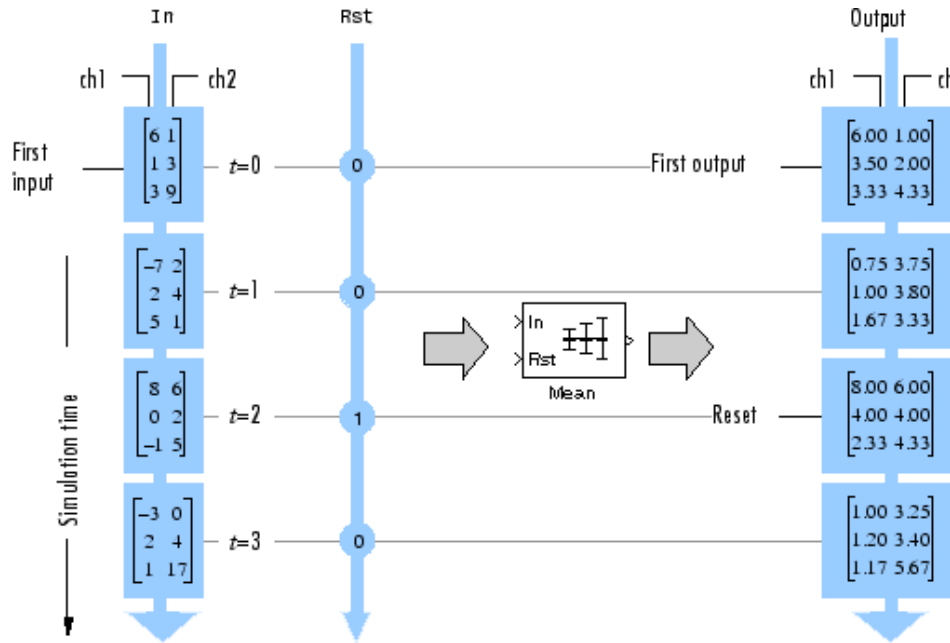
where

```
dsp_examples_u = [6 1 3 -7 2 5 8 0 -1 -3 2 1;1 3 9 2 4 1 6 2 5 0 4 17]'
```

The Discrete Impulse block has the following settings:

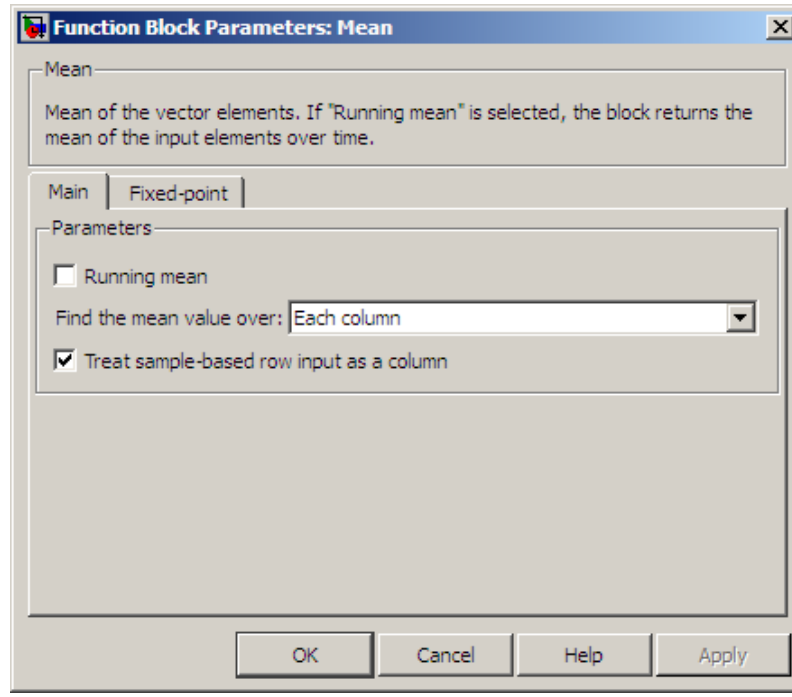
- Delay (samples) = 2
- Sample time = 1
- Samples per frame = 1

The block's operation is shown in the next Þgure.



Dialog Box

The Main pane of the Mean block dialog appears as follows.



Running mean

Enables running operation when selected.

Reset port

Determines the reset event that causes the block to reset the running mean. The rate of the reset signal must be a positive integer multiple of the rate of the data signal input. This parameter is enabled only when you set the Running mean parameter. For more information, see “Resetting the Running Mean” on page 2-880.

Find the mean value over

Specify whether to find the mean value along rows, columns, entire input, or the dimension specified in the Dimension parameter. For more information, see “Basic Operation” on page 2-878.

Treat sample-based row input as a column

Select to treat sample-based length-M row vector inputs as M-by-1 column vectors. This parameter is only visible when the Find the mean value over parameter is set to Each column.

Dimension

Specify the dimension (one-based value) of the input signal, over which the mean is computed. The value of this parameter cannot exceed the number of dimensions in the input signal. This parameter is only visible when the Find the mean value over parameter is set to Specified dimension.

Enable ROI Processing

Select this check box to calculate the statistical value within a particular region of each image. This parameter is only available when the Find the mean value over parameter is set to Entire input, and the block is not in running mode.

Note Full ROI processing is only available to users who have a Video and Image Processing Blockset license. If you only have a Signal Processing Blockset license, you can still use ROI processing, but are limited to the ROI type Rectangles.

ROI type

Specify the type of ROI you want to use. Your choices are Rectangles, Lines, Label matrix, or Binary mask.

ROI portion to process

Specify whether you want to calculate the statistical value for the entire ROI or just the ROI perimeter. This parameter is only visible if, for the ROI type parameter, you specify Rectangles.

Output

Specify the block output. The block can output a vector of separate statistical values for each ROI or a scalar value that represents the statistical value for all the specified ROIs. This parameter is not available if, for the ROI type parameter, you select Binary mask.

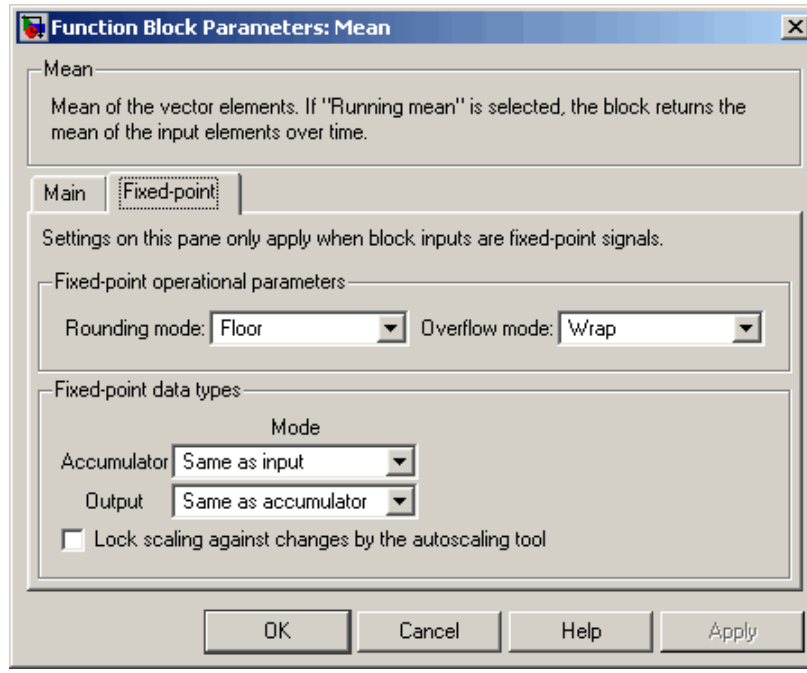
Output flag indicating if ROI is within image bounds

If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in "ROI Processing" on page 2-882. This parameter is visible if, for the ROI type parameter, you select Rectangles or Lines.

Output flag indicating if label numbers are valid

If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in "ROI Processing" on page 2-882. This parameter is visible if, for the ROI type parameter, you select Label matrix.

The Fixed-point pane of the Mean block dialog appears as follows.



Roundin g mode

Select th e rounding mode for Þxed-point operations.

Over flow m ode

Select th e overßow mode for Þxed-point operations.

Accumula tor

Use this p arameter to specify the accumulator word and fraction lengths:

- When you select Same as input , these characteristics match those of t he input to the block.
- When you select Binary point scaling , you can enter the word leng th and the fraction length of the accumulator, in bits.

- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as accumulator , these characteristics match those of the accumulator.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any Þxed-point scaling you specify in this block mask f rom being overridden by the autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Reset	• Boolean

Port	Supported Data Types
ROI	Rectangles and lines: • Double-precision ßoating point • Single-precision ßoating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers Binary Mask: • Boolean
Label	• 8-, 16-, and 32-bit unsigned integers
Label Numbers	• 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Flag	• Boolean

See Also

Maximum	Signal Processing Blockset
Median	Signal Processing Blockset
Minimum	Signal Processing Blockset
Standard Deviation	Sig nal Processing Blockset
mean	MATLAB

Purpose

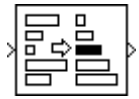
Find median value of input

Library

Statistics

dspstat3

Description



The Median block computes the median value of each row or column of the input, along vectors of a specified dimension of the input, or of the entire input. The median of a set of input values is calculated as follows:

1. The values are sorted.
2. If the number of values is odd, the median is the middle value.
3. If the number of values is even, the median is the average of the two middle values.

For a given input `u`, the size of the output array `y` depends on the setting of the Find the median value over parameter. For example, consider a 3-dimensional input signal of size M-by-N-by-P:

- Entire input — The output at each sample time is a scalar that contains the median value of the M-by-N-by-P input matrix. In this mode, the block output is always sample based.

  ```
  y = median(u(:))        % Equivalent MATLAB code
  ```

- Each row — The output at each sample time consists of an M-by-1-by-P array, where each element contains the median value of each vector over the second dimension of the input. For an input that is an M-by-N matrix, the output is an M-by-1 column vector. In this mode, the frame status of the output is the same as that of the input.

  ```
  y = median(u,2)         % Equivalent MATLAB code
  ```

- Each column — The output at each sample time consists of a 1-by-N-by-P array, where each element contains the median value of each vector over the first dimension of the input. For an input that

is an M-by-N matrix, the output at each sample time is a 1-by-N row vector. In this mode, the frame status of the output is the same as that of the input.

```
y = median(u)        % Equivalent MATLAB code
```

For convenience, length-M 1-D vector inputs are treated as M-by-1 column vectors when the block is in this mode. Sample-based length-M row vector inputs are also treated as M-by-1 column vectors when the Treat sample-based row input as a column check box is selected.

- Specified dimension — The output at each sample time depends on Dimension. If Dimension is set to 1, the output is the same as when you select Each column. If Dimension is set to 2, the output is the same as when you select Each row. If Dimension is set to 3, the output at each sample time is an M-by-N matrix containing the median value of each vector over the third dimension of the input. In this mode, the frame status of the output is the same as that of the input.

  ```
  y = median(u,Dimension)        % Equivalent MATLAB code
  ```

Complex inputs are sorted by magnitude squared. For complex value $u = a + bi$, the magnitude squared is a^2+b^2.

The Median block accepts real and complex fixed-point and floating-point inputs.

Fixed-Point Data Types

For fixed-point inputs, you can specify accumulator, product output, and output data types as discussed in “Dialog Box” on page 2-896. Not all these fixed-point parameters are applicable for all types of fixed-point inputs. The following table shows when each kind of data type and scaling is used.

	Output data type	Accumulator data type	Product output data type
Even M	X	X	
Odd M	X		
Odd M and complex	X	X	X
Even M and complex	X	X	X

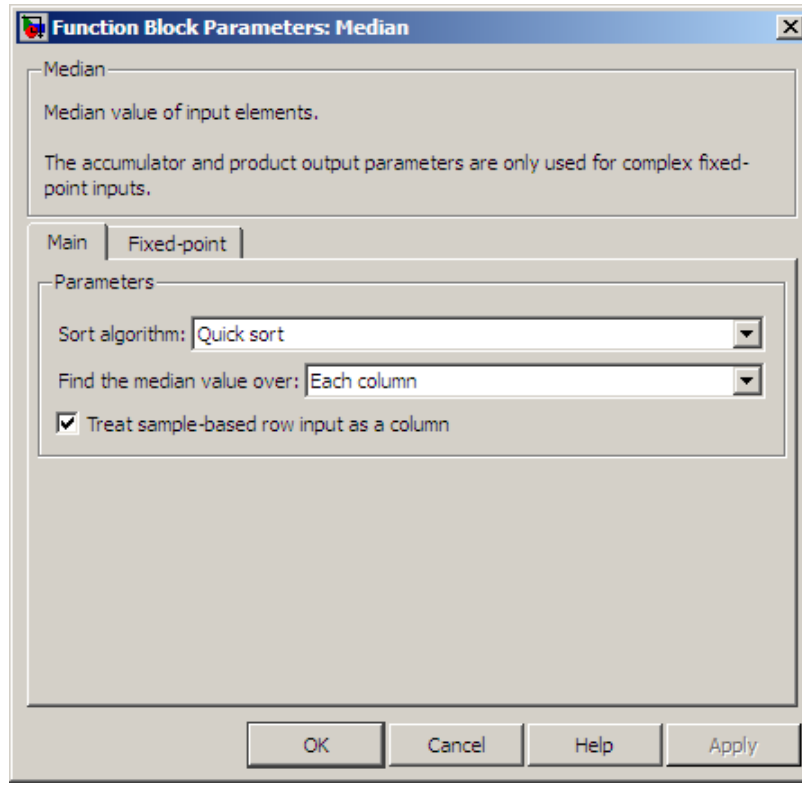
The accumulator and output data types and scalings are used for fixed-point signals when M is even. The result of the sum performed while calculating the average of the two central rows of the input matrix is stored in the accumulator data type and scaling. The total result of the average is then put into the output data type and scaling.

The accumulator and product output parameters are used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before the input elements are sorted, as described in Description. The results of the squares of the real and imaginary parts are placed into the product output data type and scaling. The result of the sum of the squares is placed into the accumulator data type and scaling.

For fixed-point inputs that are both complex and have even M, the data types are used in all of the ways described. Therefore, in such cases, the accumulator type is used in two different ways.

Dialog Box

The Main pane of the Median block dialog appears as follows.



Sort algorithm

Specify whether to sort the elements of the input using a Quick sort or an Insertion sort algorithm.

Find the median value over

Specify whether to find the median value along rows, columns, entire input, or the dimension specified in the Dimension parameter. For more information, see Description.

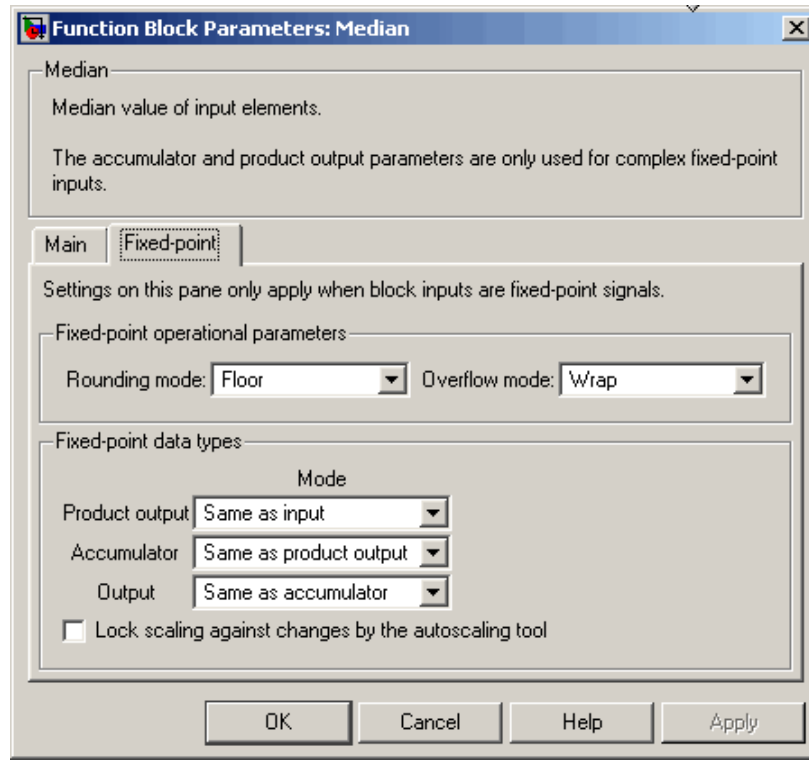
Treat sample-based row input as a column

Select to treat sample-based length- M row vector inputs as M-by-1 column vectors. This parameter is only visible when the Find the median value over parameter is set to Each column .

Dimension

Specify the dimension (one-based value) of the input signal, over which the median is computed. The value of this parameter cannot exceed the number of dimensions in the input signal. This parameter is only visible when the Find the median value over parameter is set to Specified dimension .

The Fixed-point pane of the Median block di alog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Note The product output, accumulator, and output parameters listed are only used in certain cases. See “Fixed-Point Data Types” on page 2-894 for more information.

Product output

Use this parameter to specify how you want to designate the product output word and fraction lengths:

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block:

- When you select Same as product output, these characteristics match those of the product output
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.

- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as accumulator , these characteristics match those of the accumulator.
- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any Þxed-point scaling you specify in this block mask f rom being overridden by the autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, 32-, and 128-bit signed integers • 8-, 16-, 32-, and 128-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, 32-, and 128-bit signed integers • 8-, 16-, 32-, and 128-bit unsigned integers

See Also

Maximum	Signal Processing Blockset
Mean	Signal Processing Blockset
Minimum	Signal Processing Blockset
Sort	Signal Processing Blockset
Standard Deviation	Sig nal Processing Blockset
Variance	Signal Processing Blockset
median	MATLAB

Purpose

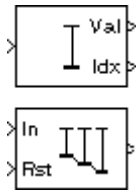
Find minimum values in input or sequence of inputs

Library

Statistics

dspstat3

Description



The Minimum block identifies the value and/or position of the smallest element in each row or column of the input, along vectors of a specified dimension of the input, or of the entire input. The Minimum block can also track the minimum values in a sequence of inputs over a period of time. The **Mode** parameter specifies the block's mode of operation, and can be set to `Value`, `Index`, `Value and Index`, or `Running`.

The Minimum block supports real and complex floating-point, fixed-point, and Boolean inputs. Real fixed-point inputs can be either signed or unsigned, while complex fixed-point inputs must be signed. The data type of the minimum values output by the block match the data type of the input. The index values output by the block are `double` when the input is `double`, and `uint32` otherwise.

The frame status of the block output is the same as that of the input, except when the **Find the minimum value of** parameter is set to `Entire input`. The output is always sample-based when `Entire input` is selected.

For the `Value`, `Index`, and `Value and Index` modes, the Minimum block produces identical results as the MATLAB `min` function when it is called as `[y I] = min(u,[],D)`, where `u` and `y` are the input and output, respectively, `D` is the dimension, and `I` is the index.

Value Mode

When the **Mode** parameter is set to `Value`, the block computes the minimum value in each row or column of the input, along vectors of a specified dimension of the input, or of the entire input at each sample time, and outputs the array `y`. Each element in `y` is the minimum value in the corresponding column, row, vector, or entire input. The output `y` depends on the setting of the **Find the minimum value**

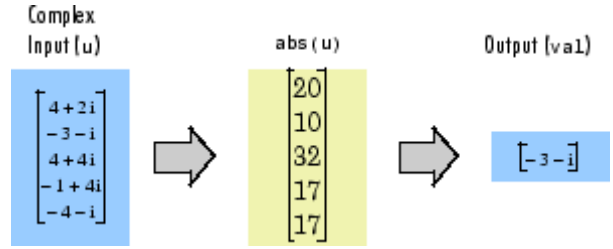
over parameter. For example, consider a 3-dimensional input signal of size M-by-N-by-P:

- Each row — The output at each sample time consists of an M-by-1-by-P array, where each element contains the minimum value of each vector over the second dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is an M-by-1 column vector. In this mode, the frame status of the output is the same as that of the input.
- Each column — The output at each sample time consists of a 1-by-N-by-P array, where each element contains the minimum value of each vector over the first dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is a 1-by-N row vector. In this mode, the frame status of the output is the same as that of the input.

  For convenience, length-M 1-D vector inputs are treated as M-by-1 column vectors when the block is in this mode. Sample-based length-M row vector inputs are also treated as M-by-1 column vectors when the Treat sample-based row input as a column check box is selected.
- Entire input — The output at each sample time is a scalar that contains the minimum value in the M-by-N-by-P input matrix. In this mode, the block output is always sample based.
- Specified dimension — The output at each sample time depends on Dimension. If Dimension is set to 1, the output is the same as when you select Each column. If Dimension is set to 2, the output is the same as when you select Each row. If Dimension is set to 3, the output at each sample time is an M-by-N matrix containing the minimum value of each vector over the third dimension of the input. In this mode, the frame status of the output is the same as that of the input.

For complex inputs, the block selects the value in each row or column of the input, along vectors of a specified dimension of the input, or of the

entire input that has the minimum ma gnitude squared as shown below. For complex value $u = a + bi$, the magnitude squared is $a^2 + b^2$.



Index Mode

When Mode is set to Index, the block computes the minimum value in each row or column of the input, along vectors of a speci Þed dimension of the input, or of the entire input, and outputs the index array I. Each element in I is an integer indexing the minimum value in the corresponding column, row, vector, or entire input. The output I depends on the setting of the Find the minimum value over parameter. For example, consider a 3-dimensional input signal of size M-by-N-by-P:

- Each row — The output at each sample time consists of an M-by-1-by-P array, where each element contains the index of the minimum value of each vector over the second dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is an M-by-1 column vector. In this mode, the frame status of the output is the same as that of the input.
- Each column — The output at each sample time consists of a 1-by-N-by-P array, where each element contains the index of the minimum value of each vector over the Þrst dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is a 1-by- N row vector. In this mode, the frame status of the output is the same as that of the input.

  For convenience, length- M 1-D vector inputs are treated as M-by-1 column vectors when the block is in this mode. Sample-based length- M row vector inputs are also treated as M-by-1 column vectors

when the Treat sample-based row input as a column check box is selected.

- Entire input — The output at each sample time is a 1-by-3 vector that contains the location of the minimum value in the M-by-N-by-P input matrix. In this mode, the block output is always sample based. For an input that is an M-by-N matrix, the output will be a 1-by-2 vector.
- Specified dimension — The output at each sample time depends on Dimension. If Dimension is set to 1, the output is the same as when you select Each column. If Dimension is set to 2, the output is the same as when you select Each row. If Dimension is set to 3, the output at each sample time is an M-by-N matrix containing the indices of the minimum values of each vector over the third dimension of the input. In this mode, the frame status of the output is the same as that of the input.

When a minimum value occurs more than once, the computed index corresponds to the first occurrence. For example, when the input is the column vector [-1 2 3 2 -1]', the computed one-based index of the minimum value is 1 rather than 5 when Each column is selected.

Value and Index Mode

When Mode is set to Value and Index, the block outputs both the minima and the indices.

Running Mode

When Mode is set to Running, the block tracks the minimum value of each channel in a time sequence of M-by-N inputs. For sample-based inputs, the output is a sample-based M-by-N array with each element y_{ij} containing the minimum value observed in element u_{ij} for all inputs since the last reset. For frame-based inputs, the output is a frame-based M-by-N array with each element y_{ij} containing the minimum value observed in the jth column of all inputs since the last reset, up to and including element u_{ij} of the current input.

N-D signals cannot be frame based. When the block is set to Running mode, each element of the N-D signal is considered as a separate channel. There are $\prod d_i$ channels, where d_i is the size of the ith dimension.

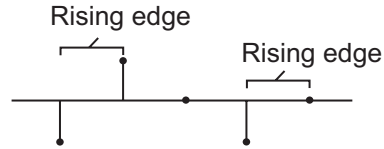
Resetting the Running Minimum

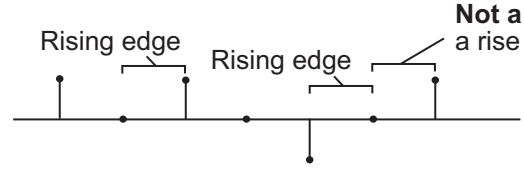
The block resets the running minimum whenever a reset event is detected at the optional Rst port. The reset sample time must be a positive integer multiple of the input sample time.

When the block is reset for sample-based inputs, the running minimum for each channel is initialized to the value in the corresponding channel of the current input. For frame-based inputs, the running minimum for each channel is initialized to the earliest value in each channel of the current input.

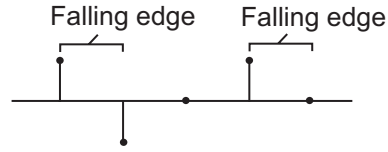
You specify the reset event by the Reset port parameter:

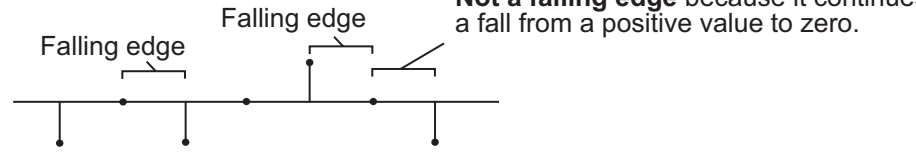
- None — Disables the Rst port
- Rising edge — Triggers a reset operation when the Rst input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following figure)





- Falling edge — Triggers a reset operation when the Rst input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)





- Either edge — Triggers a reset o peration when the Rst input is a Rising edge or Falling edge (as described above)
- Non-zero sample — Triggers a reset opera tion at each sample time that the Rst input is not zero

Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see "Excess Algorithmic Delay (Tasking Latency)" and "Models with Multiple Sample Rates" in the Real-Time Workshop User's Guide.

ROI Processing

To calculate the statistical value within a particular region of interest (ROI) of the input, select the Enable ROI processing check box. This option is only available when the Find the minimum value over parameter is set to Entire input and the Enable ROI processing check box is selected. ROI processing is only supported for 2-D inputs.

Note Full ROI processing is only available to users who have a Video and Image Processing Blockset license. If you only have a Signal Processing Blockset license, you can still use ROI processing, but are limited to the ROI type Rectangles.

Use the ROI type parameter to specify whether the ROI is a rectangle, line, label matrix, or binary mask. A binary mask is a binary image that enables you to specify which pixels to highlight, or select. In a label matrix, pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on. When the ROI type parameter is set to Label matrix, the Label and Label Numbers ports appear on the block. Use the Label Numbers port to specify the objects in the label matrix for which the block calculates statistics. The input to this port must be a vector of scalar values that correspond to the labeled regions in the label matrix. For more information about the format of the input to the ROI port when the ROI is a rectangle or a line, see the Draw Shapes block reference page.

For rectangular ROIs, use the ROI portion to process parameter to specify whether to calculate the statistical value for the entire ROI or just the ROI perimeter.

Use the Output parameter to specify the block output. The block can output separate statistical values for each ROI or the statistical value for all specified ROIs. This parameter is not available if, for the ROI type parameter, you select Binary mask.

If, for the ROI type parameter, you select Rectangles or Lines, the Output flag indicating if ROI is within image bounds check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

Output = Individual statistics for each ROI

Flag Port Output	Description
0	ROI is completely outside the input image.
1	ROI is completely or partially inside the input image.

Output = Single statistic for all ROIs

Flag Port Output	Description
0	All ROIs are completely outside the input image.
1	At least one ROI is completely or partially inside the input image.

If the ROI is partially outside the image, the block only computes the statistical values for the portion of the ROI that is within the image.

If, for the ROI type parameter, you select Label matrix, the Output flag indicating if input label numbers are valid check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

Output = Individual statistics for each ROI

Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

Output = Single statistic for all ROIs

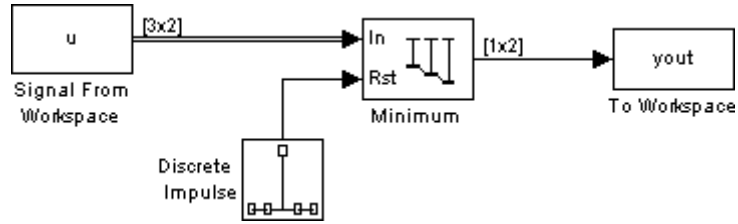
Flag Port Output	Description
0	None of the label numbers are in the label matrix.
1	At least one of the label numbers is in the label matrix.

Fixed-Point Data Types

The parameters on the Fixed-point pane of the block dialog are only used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before a comparison is made, as described in “Value Mode” on page 2-901. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into the accumulator data type. These parameters are ignored for other types of inputs.

Examples

The Minimum block in the followin g model calculates the running minimum of a frame-based 3-by-2 (two-channel) matrix input. The running minimum is reset at t=2 by an impulse to the block's Rst port.



The Minimum block has the following settings:

- Mode = Running
- Reset port = Non-zero sample

The Signal From Workspace bloc k has the follo wing settings:

- Signal = u
- Sample time = 1/3
- Samples per frame = 3

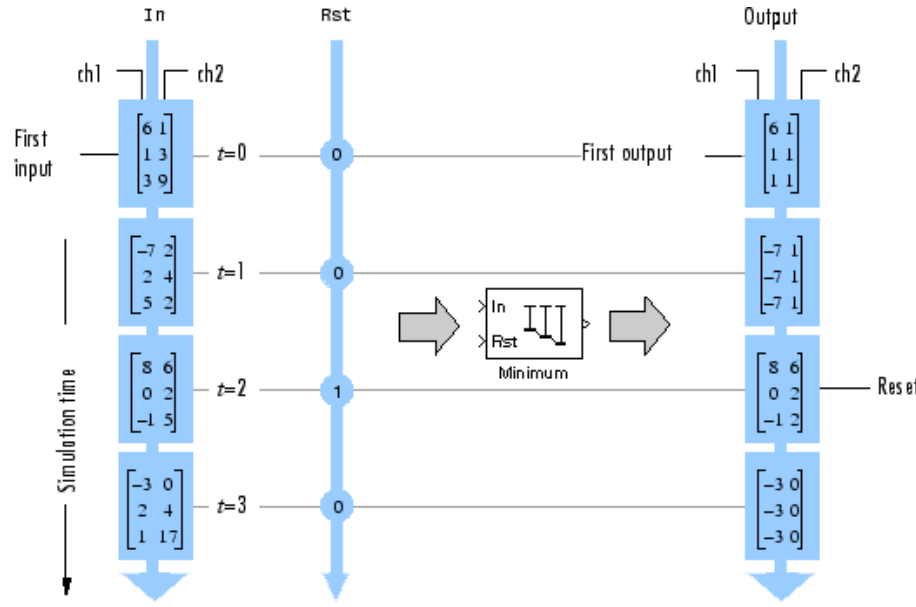
where

```
u = [6 1 3 -7 2 5 8 0 -1 -3 2 1;1 3 9 2 4 2 6 2 5 0 4 17]'
```

The Discrete Impulse block has the following settings:

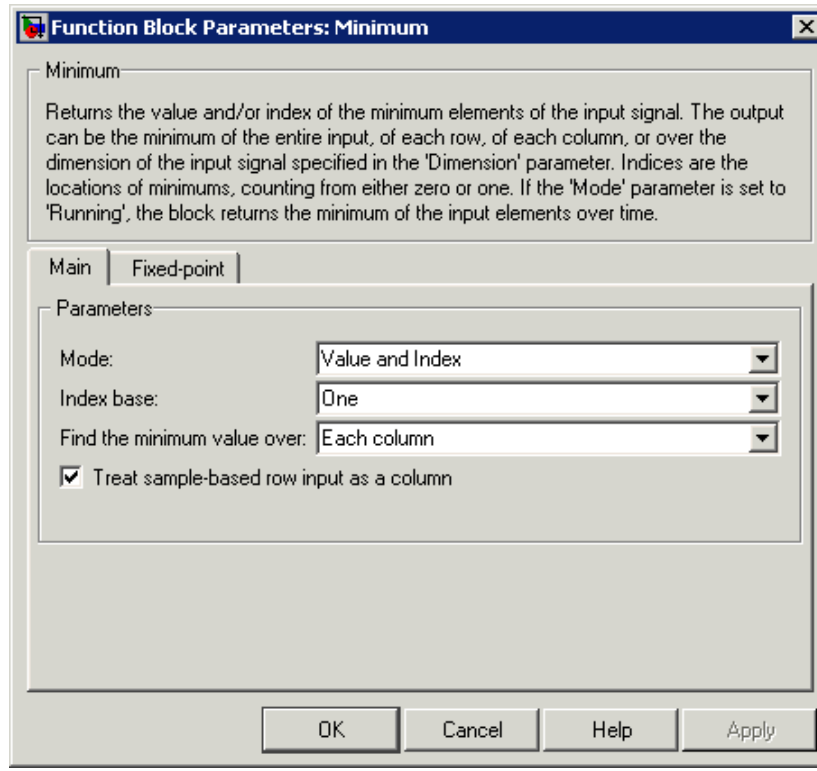
- Delay (samples) = 2
- Sample time = 1
- Samples per frame = 1

The block's operation is shown in the Þgure below.


In
Rst
Output
ch1
ch2
ch1
ch2
First input
t=0
t=1
t=2
t=3
0
0
1
0
First output
In
Rst
Minimum
Reset
Simulation time

Dialog Box

The Main pane of the Minimum block dialog appears as follows.



Mode

Specify the block's mode of operation:

- Value — Output the minimum value of each input
- Index — Output the index of the minimum value
- Value and index — Output both the value and the index
- Running — Track the minimum value of the input sequence over time

For more information, see Description.

Index base

Specify whether the index of the minimum value is reported using one-based or zero-based numbering. This parameter is only visible when the Mode parameter is set to Index or Value and index .

Find the minimum value over

Specify whether to find the minimum value along rows, columns, entire input, or the dimension specified in the Dimension parameter. For more information, see Description.

Reset port

Specify the reset event detected at the RST input port when you select Running for the Mode parameter. The rate of the reset signal must be a positive integer multiple of the rate of the data signal input. This parameter is enabled only when you set the Mode parameter to Running . For information about the possible values of this parameter, see " Resetting the Running Minimum" on page 2-905.

Treat sample-based row input as a column

Select to treat sample-based length- M row vector inputs as M-by-1 column vectors. This parameter is only visible when the Find the minimum value of parameter is set to Each column .

Dimension

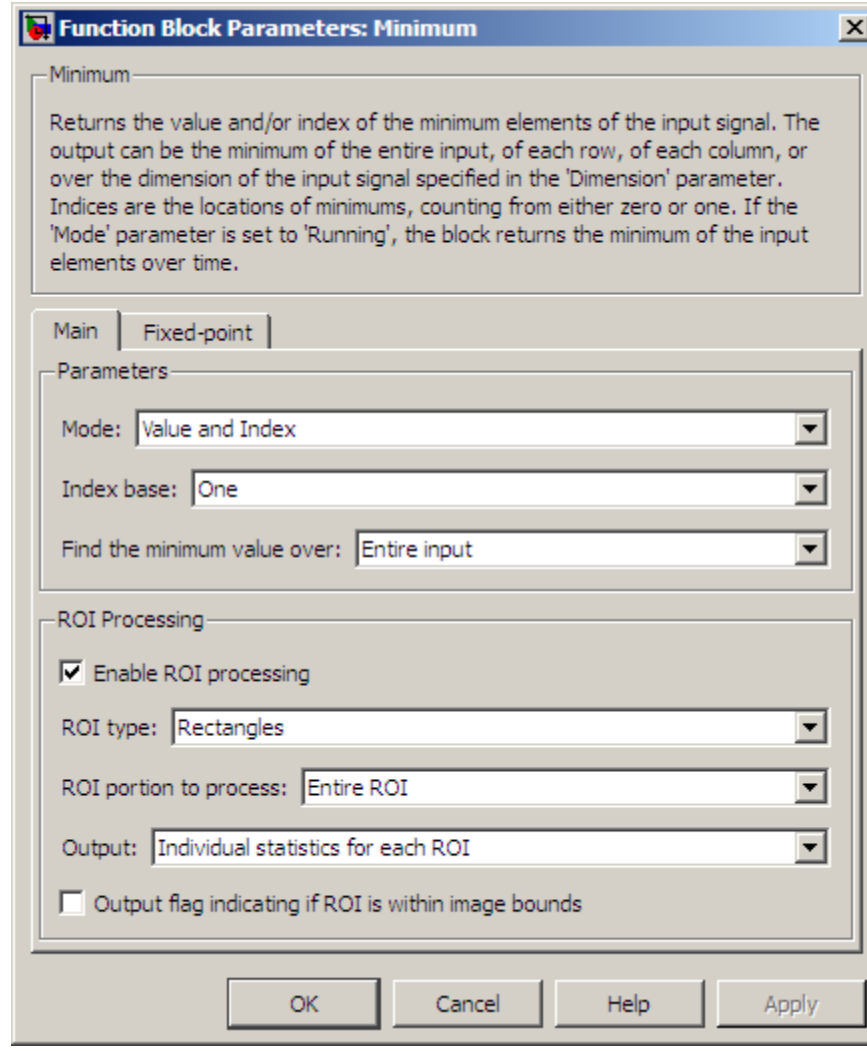
Specify the dimension (one-based value) of the input signal, over which the minimum is computed. The value of this parameter cannot exceed the number of dimensions in the input signal. This parameter is only visible when the Find the minimum value over parameter is set to Specified dimension .

Enable ROI processing

Select this check box to calculate the statistical value within a particular region of each image. This parameter is only available when the Find the minimum value over parameter is set to Entire input , and the block is not in running mode.

Note Full ROI processing is only available to users who have a Video and Image Processing Blockset license. If you only have a Signal Processing Blockset license, you can still use ROI processing, but are limited to the ROI type Rectangles .

The ROI processing parameters appear as follows.



ROI type

Specify the type of ROI you want to use. Your choices are Rectangles , Lines , Label matrix , or Binary mask .

ROI portion to process

Specify whether you want to calculate the statistical value for the entire ROI or just the ROI perimeter. This parameter is only visible if, for the ROI type parameter, you specify Rectangles.

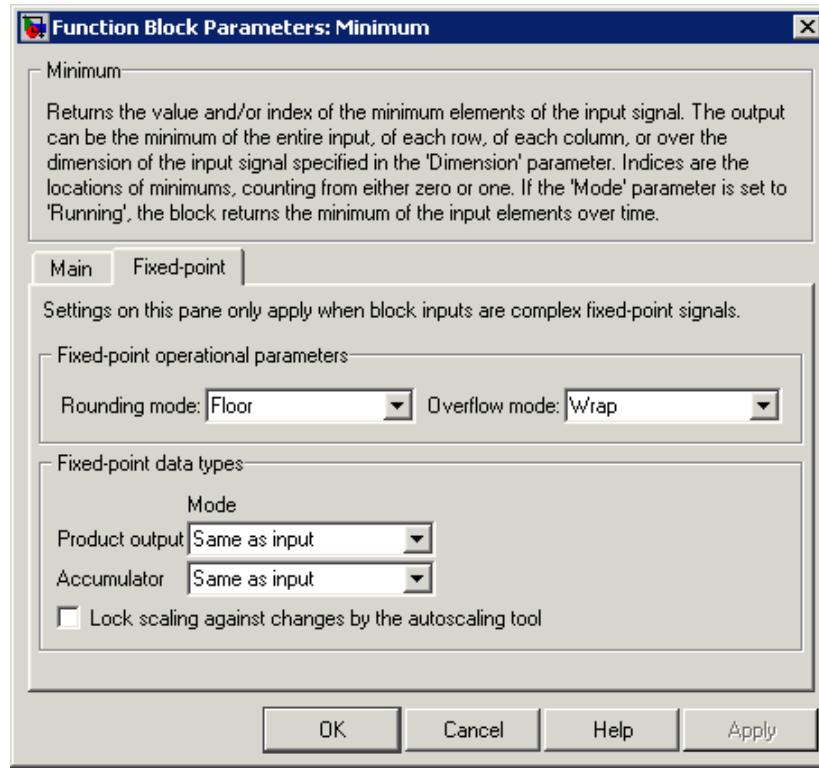
Output

Specify the block output. The block can output a vector of separate statistical values for each ROI or a scalar value that represents the statistical value for all the specified ROIs. This parameter is not available if, for the ROI type parameter, you select Binary mask.

Output flag indicating if ROI is within image bounds

If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in ROI Processing"ROI Processing" on page 2-907. This parameter is visible if, for the ROI type parameter, you select Rectangles or Lines.

The Fixed-point pane of the Minimum block dialog appears as follows.



Note The parameters on the Fixed-point pane are only used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before a comparison is made, as described in "Value Mode" on page 2-901. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into the accumulator data type. These parameters are ignored for other types of inputs.

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths resulting from a complex-complex multiplication in the block. See “Multiplication Data Types” for more information:

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block. See “Multiplication Data Types” for more information:

- When you select Same as product output, these characteristics match those of the product output.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling feature of the Fixed-Point Tool. See the `fxptdlg` reference page for more information.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Reset	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Idx	• Double-precision floating point • 32-bit unsigned integers
Val	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Maximum	Signal Processing Blockset
Mean	Signal Processing Blockset
MinMax	Simulink
Histogram	Signal Processing Blockset
min	MATLAB

Purpose

Compute estimate of autoregressive (AR) model parameters using modified covariance method

Library

Estimation / Parametric Estimation

dspparest3

Description

The Modified Covariance AR Estimator block uses the modified covariance method to fit an autoregressive (AR) model to the input data. This method minimizes the forward and backward prediction errors in the least squares sense. The input is a frame of consecutive time samples, which is assumed to be the output of an AR system driven by white noise. The block computes the normalized estimate of the AR system parameters, A(z), independently for each successive input.

$$H(z) = \frac{G}{A(z)} = \frac{G}{1 + a(2)z^{-1} + \ldots + a(p+1)z^{-p}}$$

You specify the order, p, of the all-pole model in the Estimation order parameter. To guarantee a valid output, you must set the Estimation order parameter to be less than or equal to two thirds the input vector length.

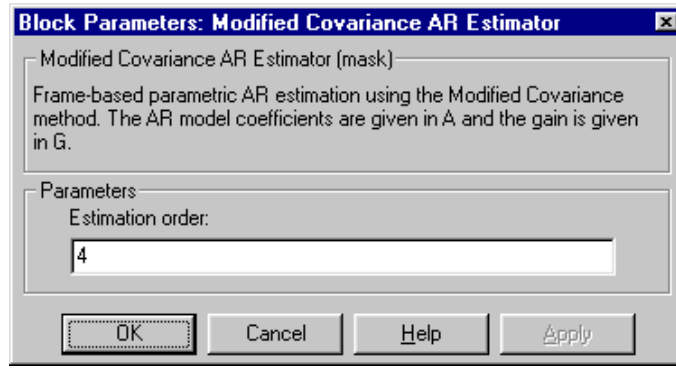
The output port labeled A outputs the normalized estimate of the AR model coefficients in descending powers of z.

```
[1  a(2)  ...  a(p+1)]
```

The scalar gain, G, is output from the output port labeled G.

See the Burg AR Estimator block reference page for a comparison of the Burg AR Estimator, Covariance AR Estimator, Modified Covariance AR Estimator, and Yule-Walker AR Estimator blocks.

Dialog Box



Estimation order

The order of the AR model, p.

References

Kay, S. M. Modern Spectral Estimation: Theory and Application. Englewood Cliffs, NJ: Prentice-Hall, 1988.

Marple, S. L., Jr., Digital Spectral Analy sis with Applications. Englewood Cliffs, NJ: Prentice-Hall, 1987.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point
A	• Double-precision ßoating point • Single-precision ßoating point
G	• Double-precision ßoating point • Single-precision ßoating point

The output data type is the same as the input data type.

See Also

Burg AR Estimator	Signal Processing Blockset
Covariance AR Estimator	Signal Processing Blockset
Modified Covariance Method	Signal Processing Blockset
Yule-Walker AR Estimator	Signal Processing Blockset
armcov	Signal Processing Toolbox

Modified Covariance Method

Purpose

Compute parametric spectral estimate using modified covariance method

Library

Estimation / Power Spectrum Estimation

dspspect3

Description

The Modified Covariance Method block estimates the power spectral density (PSD) of the input using the modified covariance method. This method fits an autoregressive (AR) model to the signal by minimizing the forward and backward prediction errors in the least squares sense. The order of the all-pole model is the value specified by the Estimation order parameter. To guarantee a valid output, you must set the Estimation order parameter to be less than or equal to two thirds the input vector length. The spectrum is computed from the FFT of the estimated AR model parameters.

The input is a sample-based vector (row, column, or 1-D) or frame-based vector (column only) representing a frame of consecutive time samples from a single-channel signal. The block's output (a column vector) is the estimate of the signal's power spectral density at N_{fft} equally spaced frequency points in the range [0, F_s), where F_s is the signal's sample frequency.

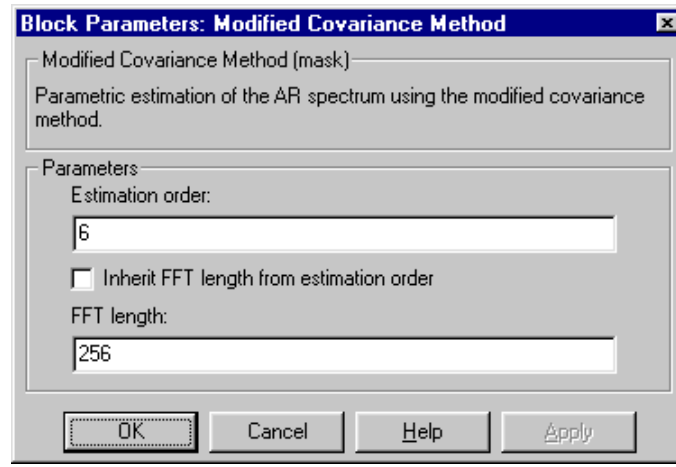
When you select Inherit FFT length from estimation order, N_{fft} is one greater than the estimation order. When you do not select Inherit FFT length from estimation order, N_{fft} is specified as a power of 2 by the FFT length parameter, and the block zero pads or wraps the input to N_{fft} before computing the FFT. The output is always sample based.

See the Burg Method block reference for a comparison of the Burg Method, Covariance Method, Modified Covariance Method, and Yule-Walker Method blocks.

Examples

The dspsacomp demo compares the modified covariance method with several other spectral estimation methods.

Dialog Box



Estimation order

The order of the AR model.

Inherit FFT length from estimation order

When selected, the FFT length is one greater than the estimation order.

FFT length

Enter the number of data points on which to perform the FFT, N_{fft}. When N_{fft} is larger than the input frame size, each frame is zero-padded as needed. When N_{fft} is smaller than the input frame size, each frame is wrapped as needed. This parameter is enabled when you clear the Inherit FFT length from estimation order check box.

References

Kay, S. M. Modern Spectral Estimation: Theory and Application. Englewood Cliffs, NJ: Prentice-Hall, 1988.

Marple, S. L., Jr., Digital Spectral Analy sis with Applications. Englewood Cliffs, NJ: Prentice-Hall, 1987.

Orfanidis, S. J. Introduction to Signal Processing . Englewood Cliffs, NJ: Prentice-Hall, 1995.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point
Output	• Double-precision floating point • Single-precision floating point

The output data type is the same as the input data type.

See Also

Burg Method	Signal Processing Blockset
Covariance Method	Signal Processing Blockset
Modified Covariance AR Estimator	Signal Processing Blockset
Short-Time FFT	Signal Processing Blockset
Yule-Walker Method	Signal Processing Blockset
pmcov	Signal Processing Toolbox

See “Power Spectrum Estimation” for related information.

Purpose

Generate multiple binary clock signals

Library

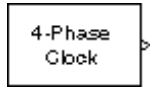
- Signal Processing Sources

  dspsrcs4

- Signal Management / Switches and Counters

  dspswit3

Description



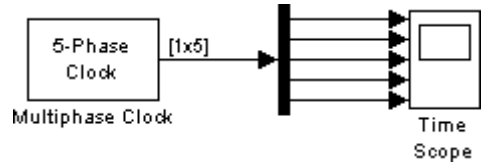
The Multiphase Clock block generates a sample-based 1-by- N vector of clock signals, where you specify the integer N in the Number of phases parameter. Each of the N phases has the same frequency, f, specified in hertz by the Clock frequency parameter.

The clock signal indexed by the Starting phase parameter is the first to become active, at t=0. The other signals in the output vector become active in turn, each one lagging the preceding signal's activation by 1/(N*f) seconds, the phase interval. The period of the sample-based output is therefore 1/(N*f) seconds.

The active level can be either high (1) or low (0), as specified by the Active level (polarity) parameter. The duration of the active level, D, is set by the Number of phase intervals over which the clock is active. This value, which can be an integer value between 1 and N-1, specifies the number of phase intervals that each signal should remain in the active state after becoming active. The active duty cycle of the signal is D/N.

Examples

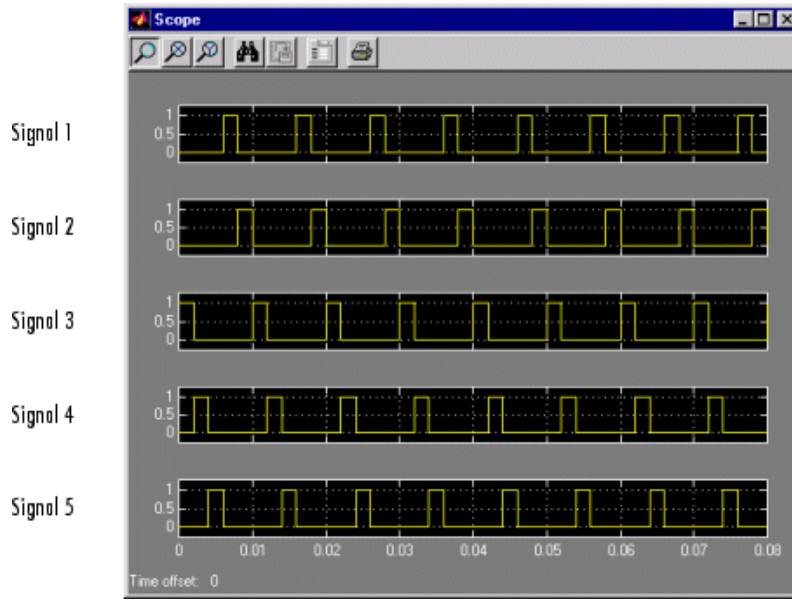
Configure the Multiphase Clock block in the model below to generate a 100 Hz five-phase output in which the third signal is first to become active. Use a high active level with a duration of one interval.



The corresponding settings are as follows:

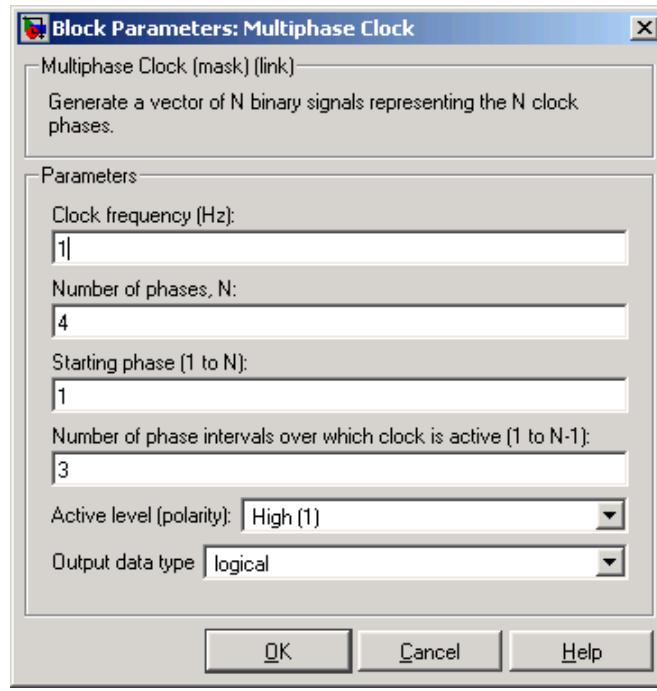
- Clock frequency = 100
- Number of phases = 5
- Starting phase = 3
- Number of phase intervals over which the clock is active = 1
- Active level (polarity) = High (1)

The Scope window below shows the Multiphase Clock block's output for these settings. Note that the first active level appears at t=0 on y(3), the second active level appears at t=0.002 on y(4), the third active level appears at t=0.004 on y(5), the fourth active level appears at t=0.006 on y(1), and the fifth active level appears at t=0.008 on y(2). Each signal becomes active 1/(5*100) seconds after the previous signal.



To experiment further, try changing the Number of phase intervals over which clock is active setting to 3 so that the active-level duration is three phase intervals (60% duty cycle).

Dialog Box



Opening this dialog causes a running simulation to pause. See "Changing Source Block Parameters During Simulation" in the online Simulink documentation for details.

Clock frequency
: The frequency of all output clock signals.

Number of phases
: The number of different phases, N, in the output vector.

Starting phase
: The vector index of the output signal to first become active.

Number of phase intervals over which clock is active
: The duration of the active level for every output signal.

Active level
: The active level, High (1) or Low (0) .

Output data type
: The output data type.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Boolean

See Also

Clock	Simulink
Counter	Signal Processing Blockset
Pulse Generator	Simulink
Event-Count Comparator	S ignal Processing Blockset

Purpose

Distribute arbitrary subsets of input rows or columns to multiple output ports

Library

Signal Management / Indexing

dspindex

Description

The Multiport Selector block extracts multiple subsets of rows or columns from M-by-N input matrix u, and propagates each new submatrix to a distinct output port. A length-M 1-D vector input is treated as an M-by-1 matrix.

The Indices to output parameter is a cell array whose kth cell contains a one-dimensional indexing expression specifying the subset of input rows or columns to be propagated to the kth output port. The total number of cells in the array determines the number of output ports on the block.

When the Select parameter is set to Rows, the specified one-dimensional indices are used to select matrix rows, and all elements on the chosen rows are included. When the Select parameter is set to Columns, the specified one-dimensional indices are used to select matrix columns, and all elements on the chosen columns are included. A given input row or column can appear any number of times in any of the outputs, or not at all.

When an index references a nonexistent row or column of the input, the block reacts with the behavior specified by the Invalid index parameter. The following options are available:

- Clip index — Clip the index to the nearest valid value, and do not issue an alert.

  Example: For a 64-by-4 input with Select = Rows, an index of 72 is clipped to 64; with Select = Columns, an index of 72 is clipped to 4. In both cases, an index of -2 is clipped to 1.

- Clip and warn — Display a warning message in the MATLAB Command Window, and clip as above.

- Generate error — Display an error dialog and terminate the simulation.

Examples

Example 1

Consider the following Indices to output cell array:

```
{4,[1:2 5],[7;        8],10:-1:6}
```

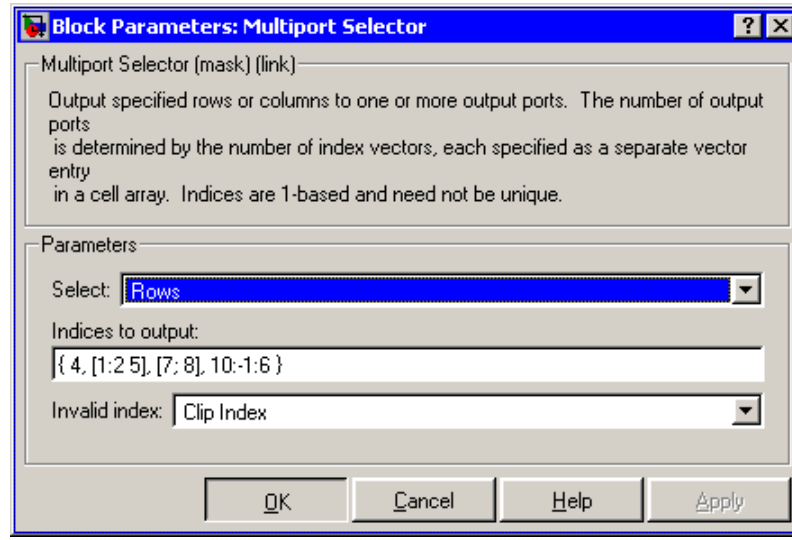
This is a four-cell array, which requires the block to generate four independent outputs (each at a distinct port). The table below shows the dimensions of these outputs when Select = Rows and the input dimension is M-by-N.

Cell	Expression	Description	Output Size
1	4	Row 4 of input	1-by-N
2	[1:2 5]	Rows 1, 2, and 5 of input	3-by-N
3	[7;8]	Rows 7 and 8 of input	2-by-N
4	10:-1:6	Rows 10, 9, 8, 7, and 6 of input	5-by-N

Example 2

To see the Multiport Selector block used in a model, see “Splitting Multichannel Sample-Based Signals into Individual Signals”, and “Splitting Multichannel Frame-Based Signals into Individual Signals” in the Signal Processing Blockset User’s Guide.

Dialog Box



Select

The dimension of the input to select, Rows or Columns .

Indices to output

A cell array specifying the row- or column-subsets to propagate to each of the output ports. The number of cells in the array determines the number of output ports on the block.

Invalid index

Response to an invalid index value.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Outputs	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Permute Matrix	Signal Processing Blockset
Selector	Simulink
Submatrix	Signal Processing Blockset
Variable Selector	Signal Processing Blockset

Purpose

Output ones or zeros for specified number of sample times

Library

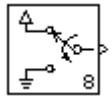
- Signal Processing Sources

  dspsrcs4

- Signal Management / Switches and Counters

  dspswit3

Description



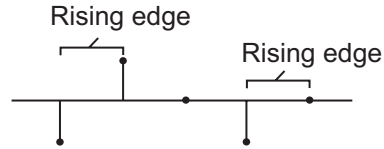
The N-Sample Enable block outputs the inactive value (0 or 1, whichever is not selected in the Active level parameter) during the first N sample times, where N is the Trigger count value. Beginning with output sample N+1, the block outputs the active value (1 or 0, whichever you select in the Active level parameter) until a reset event occurs or the simulation terminates.

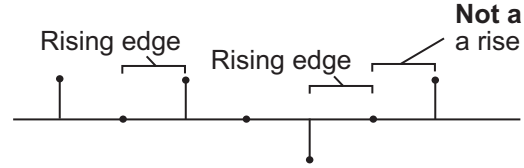
The output is always sample based.

The Reset input check box enables the Rst input port. At any time during the count, a trigger event at the input port resets the counter to its initial state. The reset sample time must be a positive integer multiple of the input sample time . This block supports triggered subsystems when you select the Reset input check box.

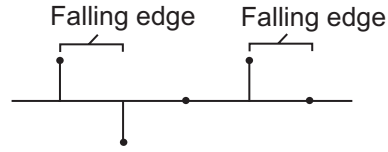
You specify the triggering event in the Trigger type pop-up menu:

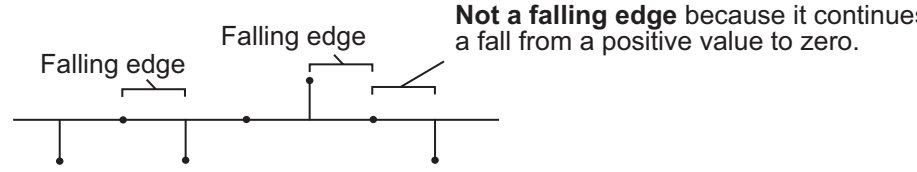
- Rising edge — Triggers a reset operation when the Rst input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following figure)





- Falling edge — Triggers a reset operation when the Rst input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)

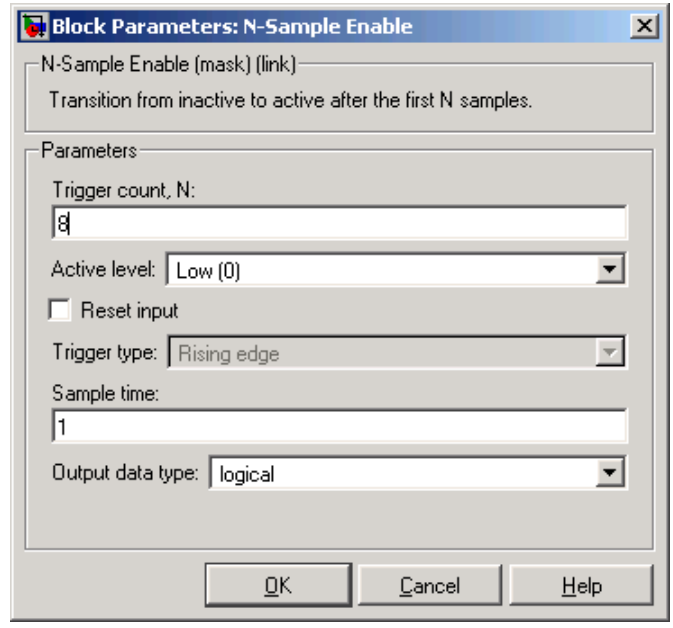




- Either edge — Triggers a reset o peration when the Rst input is a Rising edge or Falling edge (as described above).
- Non-zero sample — Triggers a reset opera tion at each sample time that the Rst input is not zero.

Note When running simulations in the Simulink MultiTasking mode, sample-based reset signals have a one-sample latency, and frame-based reset signals have one frame of latency. Thus, there is a one-sample or one-frame delay between the time the block detects a reset event, and when it applies the reset. For more information on latency and the Simulink tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Mo dels with Multiple Sample Rates” in the Real-Time Workshop User’s Guide .

Dialog Box



Opening this dialog causes a running simulation to pause. See “Changing Source Block Parameters During Simulation” in the online Simulink documentation for details.

Trigger count
: Specify the number of samples for which the block outputs the active value. Tunable.

Active level
: Specify the value to output after the first N sample times, 0 or 1. Tunable.

Reset input
: Select to enable the Rst input port. The rate of the reset signal must be a positive integer multiple of the rate of the data signal input.

Trigger type
: Select type of event that triggers a reset when the Rst port is enabled.

Sample time
: Specify the sample period, T_s, for the block's counter. The block switches from the active value to the inactive value at $t=T_s*(N+1)$.

Output data type
: Select the output data type.

Supported Data Types

- Double-precision floating point
- Boolean — The block accepts Boolean inputs to the Rst port, which is enabled when you set the Reset input parameter.

See Also

Counter	Signal Processing Blockset
N-Sample Switch	Signal Processing Blockset

Purpose

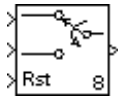
Switch between two inputs after specified number of sample periods

Library

Signal Management / Switches and Counters

dspswit3

Description



The N-Sample Switch block outputs the signal connected to the top input port during the first N sample times after the simulation begins or the block is reset, where you specify N in the Switch count parameter. Beginning with output sample N+1, the block outputs the signal connected to the bottom input until the next reset event or the end of the simulation.

You specify the sample period of the output in the Sample time parameter (that is, the output sample period is not inherited from the sample period of either input). The block applies a zero-order hold at the input ports, so the value the block reads from a given port between input sample times is the value of the most recent input to that port.

Both inputs must have the same dimension, except in the following two cases:

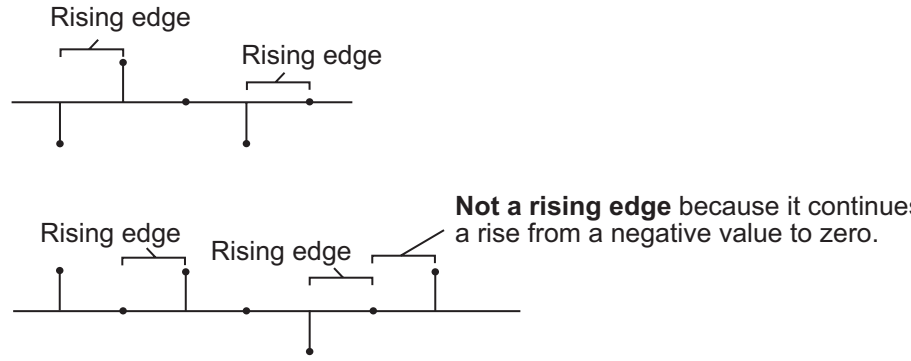
- When one input is a scalar, the block expands the scalar input to match the size of the other input.
- When one input is a 1-D vector and the other input is a row or column vector with the same number of elements, the block reshapes the 1-D vector to match the dimension of the other input.

The inputs must either both be frame based or both be sample based.

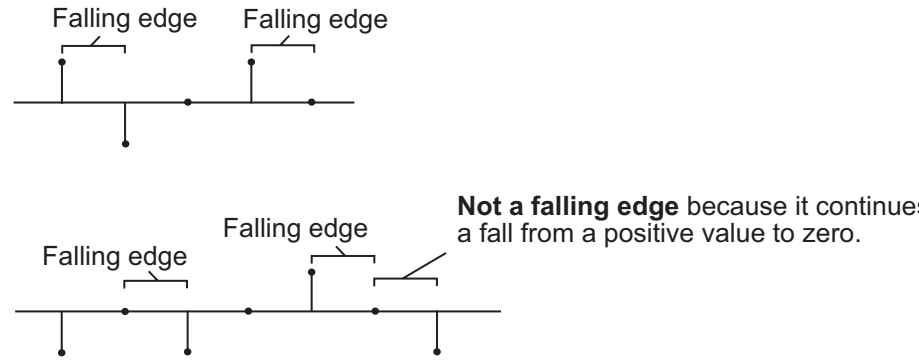
The Reset input check box enables the Rst input port. At any time during the count, a trigger event at the Rst port resets the counter to zero. The reset sample time must be a positive integer multiple of the input sample time. This block supports triggered subsystems when you select the Reset input check box.

You specify the triggering event in the Trigger type pop-up menu, and can be one of the following:

- Rising edge — Triggers a reset operation when the Rst input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following Þgure)



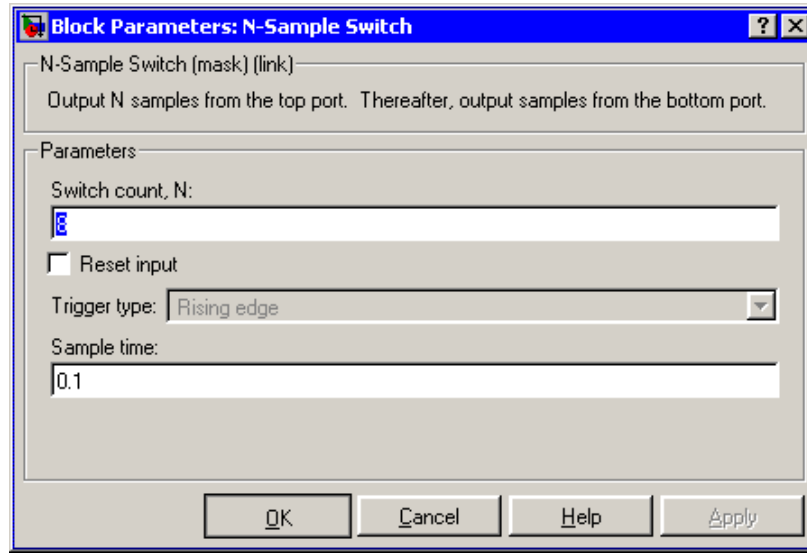
- Falling edge — Triggers a reset operation when the Rst input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)



- Either edge — Triggers a reset o peration when the Rst input is a Rising edge or Falling edge (as described above).
- Non-zero sample — Triggers a reset opera tion at each sample time that the Rst input is not zero.

Note When running simulations in the Simulink MultiTasking mode, sample-based reset signals have a one-sample latency, and frame-based reset signals have one frame of latency. Thus, there is a one-sample or one-frame delay between the time the block detects a reset event, and when it applies the reset. For more information on latency and the Simulink tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Mo dels with Multiple Sample Rates” in the Real-Time Workshop User’s Guide .

Dialog Box



Switch count

The number of sample periods, N, for which the output is connected to the top input before switching to the bottom input. Tunable.

Reset input

Enables the Rst input port when selected. The rate of the reset signal must be a positive integer multiple of the rate of the data signal input.

Trigger type

The type of event at the Rst port that resets the block's counter. This parameter is enabled when you select Reset input. Tunable.

Sample time

The sample period, T_s, for the block's counter. The block switches inputs at $t=T_s*(N+1)$.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Fixed point (signed and unsigned)
- Boolean — The block accepts Boolean inputs to the Rst port, which is enabled when you set the Reset input parameter.
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

See Also

Counter	Signal Processing Blockset
N-Sample Enable	Signal Processing Blockset

Purpose

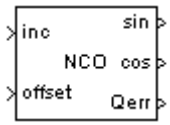
Generate real or complex sinusoidal signals

Library

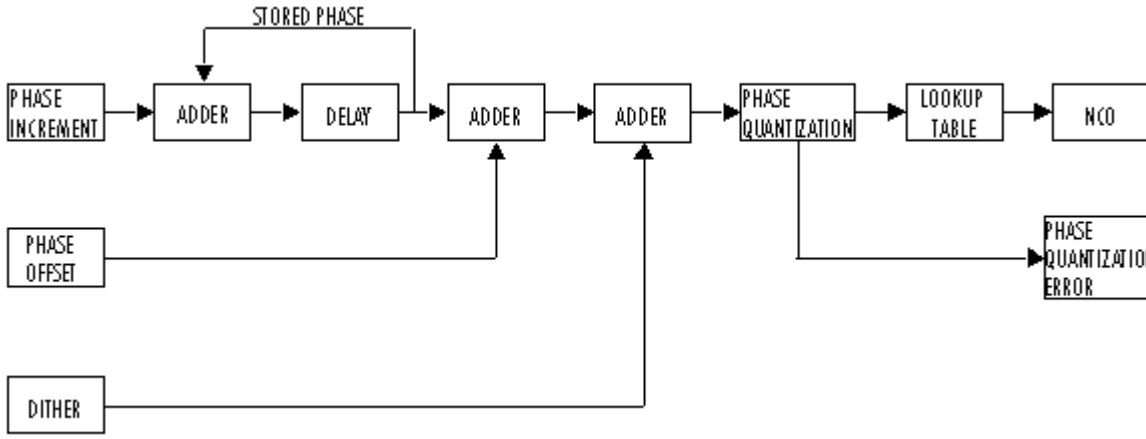
Signal Operations

dspsigops

Description



The NCO block generates a multich annel real or complex sinusoidal signal, with independent frequency and phase in each output channel. The amplitude of the created signal is always 1. The block implements the algorithm as shown in the following diagram:



The implementation of a numericall y controlled oscillator (NCO) has two distinct parts. First, a phase accumulator accumulates the phase increment and adds in the phase offset. In this stage, an optional internal dither signal can also be added. The NCO output is then calculated by quantizing the resul ts of the phase accumulator section and using them to select values from a lookup table.

Given a desired output frequency F_0, calculate the value of the Phase increment block parameter with

$$phase\ increment = (\frac{F_0 \cdot 2^N}{F_s})$$

where N is the accumulator word length and

$$F_s = \frac{1}{T_s} = \frac{1}{sample\ time}$$

The frequency resolution of an NCO is de Þned by

$$\Delta f = \frac{1}{T_s \cdot 2^N} \text{Hz}$$

Given a desired phase offset (in radians), calculate the Phase offset block parameter with

$$phase\ offset = \frac{2^N \cdot desired\ phase\ offset}{2\pi}$$

The spurious free dynamic range (SFDR) is estimated as follows for a lookup table with 2^P entries, where P is the number of quantized accumulator bits:

$SFDR = (6P)$ dB without dither

$SFDR = (6P + 12)$ dB with dither

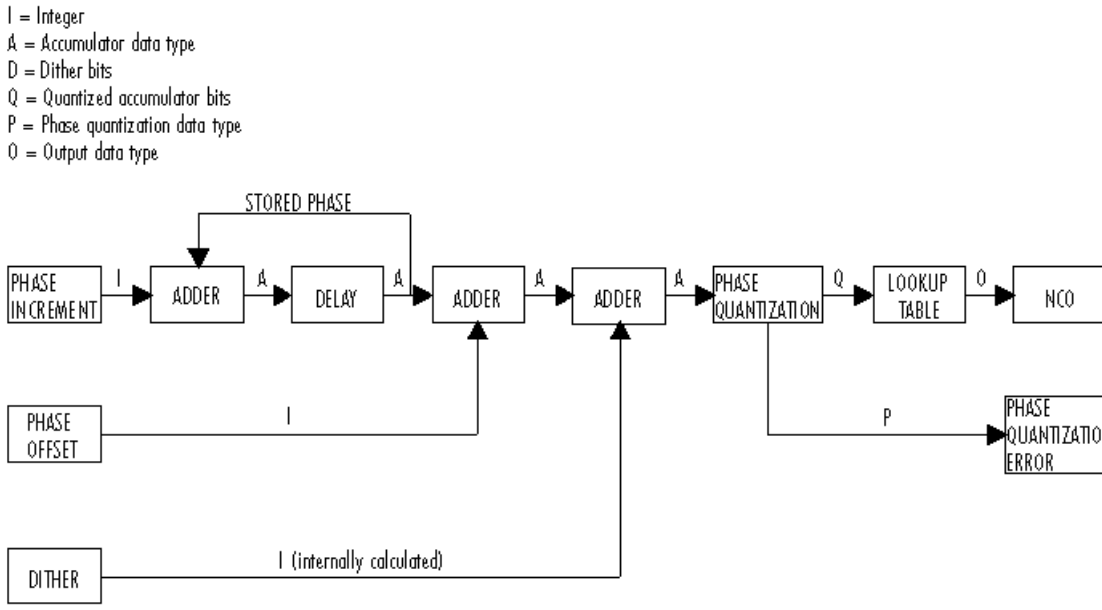
This block uses a quarter-wave lookup table technique that stores table values from 0 to π/2. The block calculates other values on demand using the accumulator data type, then cast s them into the output data type. This can lead to quantization effects at the range limits of a given data type. For example, consider a case where you would expect the value of the sine wave to be –1 at π. Because the lookup table value at that point must be calculated, the block might not yield exactly –1, depending on the precision of the accumulator and output data types.

The NCO block supports real inputs only. All outputs are real except for the output signal in Complex exponential mode.

To produce a multichannel output, specify a vector quantity for the Phase increment and Phase offset parameters. Both parameters must have the same length, which de Þnes the number of output channels. Each element of each vector is applied to a different output channel.

Fixed-Point Data Types

The following diagram shows the data types used within the NCO block.



- You can set the accumulator and output data types in the block dialo g as discussed in “Dialog Box” on page 2-954 below.

- The phase increment and phase offset inputs must be integers or Þxed-point data types with zero fraction length.
- You specify the number of quantized accumulator bits in the Number of quantized accumulator bits parameter.
- The phase quantization error word length is equal to the accumulator word length minus the number of quantized accumulator bits, and the fraction length is zero.

Examples

The NCO block is used in the GSM Digital Down Converter product demo. Open this demo by typing `dspddc` at the MATLAB command line.

You can also try the following example. Design an NCO source with the following speci Þcations:

- Desired output frequency $F_0 = 510$ Hz
- Frequency resolution $\Delta f = 0.05$ Hz
- Spurious free dynamic range $SFDR \geq 90$ dB
- Sample period $T_s = 1/8000$ s
- Desired phase offset $\pi/2$

1

Calculate the number of required accumulator bits from the equation for frequency resolution:

$$\Delta f = \frac{1}{T_s \cdot 2^N} \text{Hz}$$

$$0.05 = \frac{1}{\frac{1}{8000} \cdot 2^N} \text{Hz}$$

$$N = 18$$

Note that N must be an integer value. The value of N is rounded up to the nearest integer; 18 accumulato r bits are needed to accommodate the value of the frequency resolution.

2

Using this best value of N, calculate the frequency resolution that will be achieved by the NCO block:

$$\Delta f = \frac{1}{T_s \cdot 2^N} \text{Hz}$$

$$\Delta f = \frac{1}{\frac{1}{8000} \cdot 2^{18}} \text{ Hz}$$

$$\Delta f = 0.0305$$

3

Calculate the number of quantized accumulator bits from the equation for spurious free dynamic range and the fact that for a lookup table with 2^P entries, P is the number of quantized accumulator bits:

$$SFDR = (6P + 12)\text{dB}$$

$$96 \text{ dB} = (6P + 12) \text{ dB}$$

$$P = 14$$

4

Select the number of dither bits. In general, a good choice for the number of dither bits is the accumulator word length minus the output word length; in this case 4.

5

Calculate the phase increment:

$$phase\ increment = \text{round}(\frac{F_0 \cdot 2^N}{F_s})$$

$$phase\ increment = \text{round}(\frac{501 \cdot 2^{18}}{8000})$$

$$phase\ increment = 16417$$

6

Calculate the phase offset:

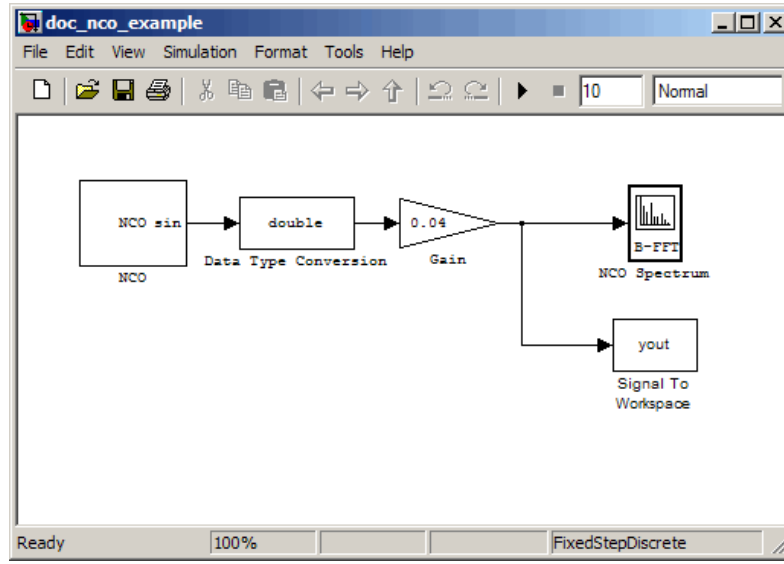
$$phase\ offset = \frac{2^{accumulator\ word\ length} \cdot desired\ \ phase\ offset}{2\pi}$$

$$phase\ offset = \frac{2^{18} \cdot \frac{\pi}{2}}{2\pi}$$

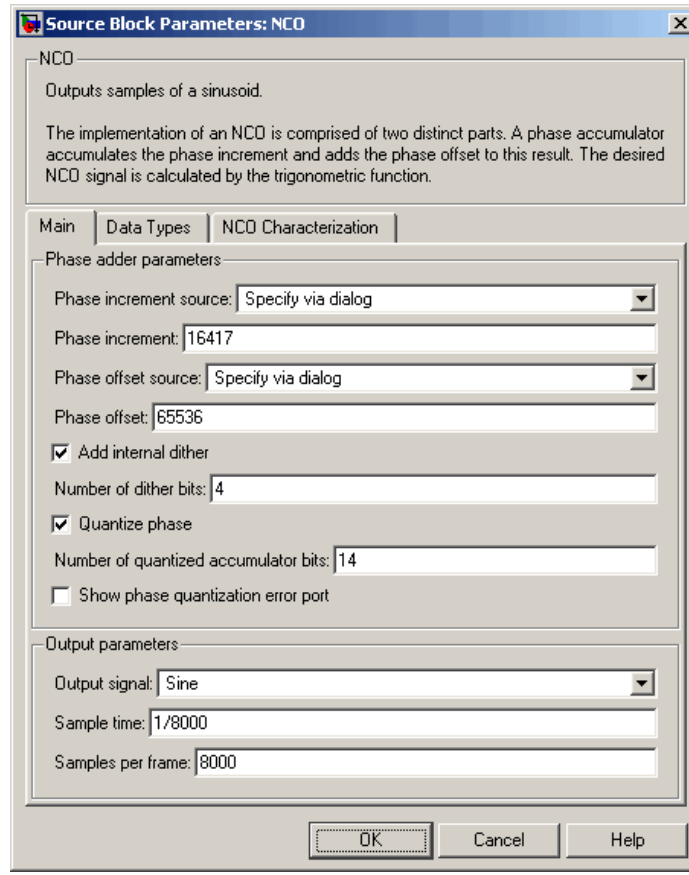
$$phase\ offset = 65536$$

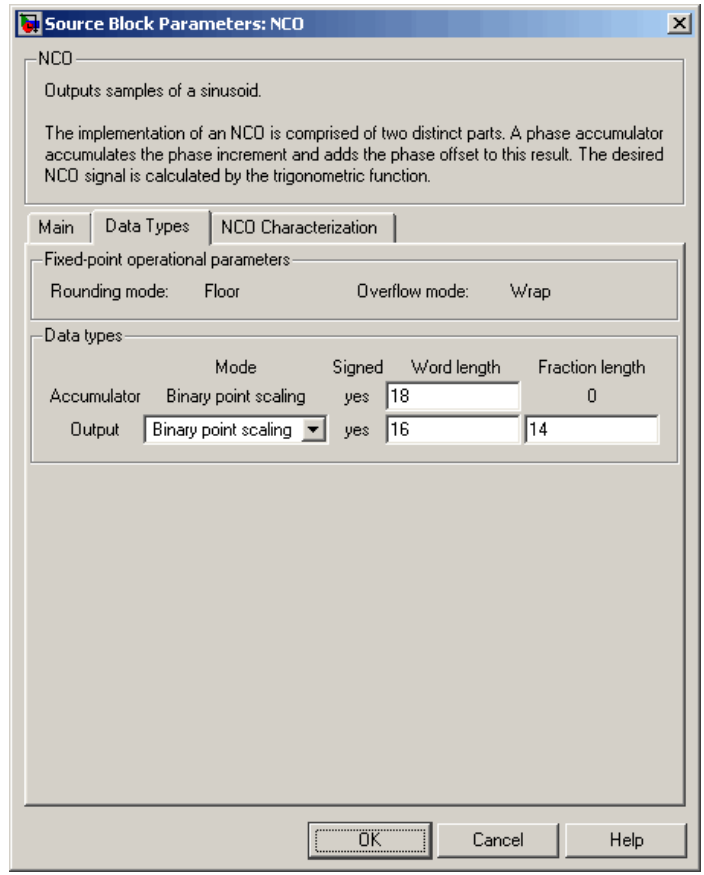
7

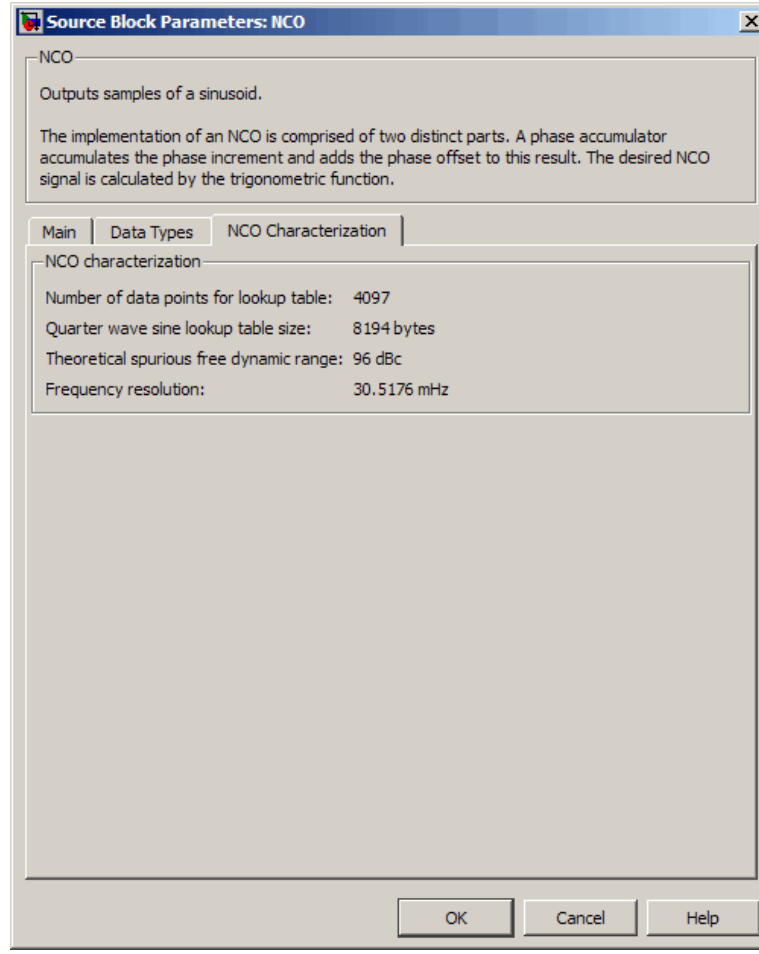
Type doc_nco_example at the MATLAB command line to open the following model:



The NCO block in the model is populated with the speci Þcations and quantities you just calculated. Th e output word length and fraction length depend on the constraints of your hardware; this example uses a word length of 16 and a fraction length of 14. The three panes of the block mask appear as follows.


Source Block Parameters: NCO
NCO
Outputs samples of a sinusoid.
The implementation of an NCO is comprised of two distinct parts. A phase accumulator accumulates the phase increment and adds the phase offset to this result. The desired NCO signal is calculated by the trigonometric function.
Main
Data Types
NCO Characterization
Phase adder parameters
Phase increment source: Specify via dialog
Phase increment: 16417
Phase offset source: Specify via dialog
Phase offset: 65536
Add internal dither
Number of dither bits: 4
Quantize phase
Number of quantized accumulator bits: 14
Show phase quantization error port
Output parameters
Output signal: Sine
Sample time: 1/8000
Samples per frame: 8000
OK
Cancel
Help


Source Block Parameters: NCO
NCO
Outputs samples of a sinusoid.
The implementation of an NCO is comprised of two distinct parts. A phase accumulator accumulates the phase increment and adds the phase offset to this result. The desired NCO signal is calculated by the trigonometric function.
Main
Data Types
NCO Characterization
Fixed-point operational parameters
Rounding mode: Floor
Overflow mode: Wrap
Data types
Mode
Signed
Word length
Fraction length
Accumulator
Binary point scaling
yes
18
0
Output
Binary point scaling
yes
16
14
OK
Cancel
Help



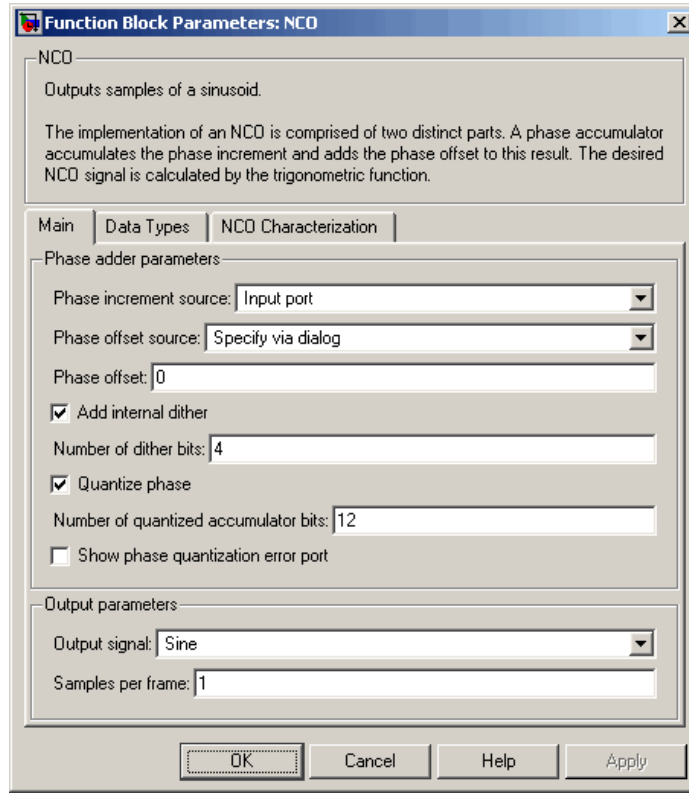
Looking at the NCO Characterization pane, you can verify that the specifications of this problem have been met.

8

Experiment with the model to observe the effects on the output shown on the Spectrum Scope. For example, try turning dithering on and off, and try changing the number of dither bits.

Dialog Box

The Main pane of the NCO dialog appears as follows.



Phase increment source

Choose how you specify the phase increment. The phase increment can come from an input port or from the dialog.

- If you select Input port , the inc port appears on the block icon.

- If you select Specify via dialog , the Phase increment parameter appears.

Phase increment

Specify the phase increment, whi ch must be a scalar or row vector with the same length as the phase offset. Each element of the vector is applied to a different channel, and therefore the vector length de Þnes the number of output channels. Only integer data types, including Þxed-point data types with zero fraction length, are allowed.

This parameter is visible only if Specify via dialog is selected for the Phase increment source parameter.

Phase offset source

Choose how you specify the phase offset. The phase offset can come from an input port or from the dialog.

- If you select Input port , the offset port appears on the block icon.
- If you select Specify via dialog , the Phase offset parameter appears.

Phase offset

Specify the phase offset, which must be a scalar or row vector with the same length as the phase increment. Each element of the vector is applied to a different channel, and therefore the vector length de Þnes the number of output channels. Only integer data types, including Þxed-point data types with zero fraction length, are allowed.

This parameter is visible only if Specify via dialog is selected for the Phase offset source parameter.

Add internal dither

Select to add internal dithering to the NCO algorithm. Dithering is added using the PN Sequence Generator from the Communications Blockset™ product.

Number of dither bits

Specify the number of dither bits.

This parameter is visible only if Add internal dither is selected.

Quantize phase

Select to enable quantization of the accumulated phase.

Number of quantized accumulator bits

Specify the number of quantized accumulator bits. This determines the number of entries in the lookup table. The number of quantized accumulator bits must be less than the accumulator word length.

This parameter is visible only if Quantize phase is selected.

Show phase quantization error port

Select to output the phase quantization error. When you select this, the Qerr port appears on the block icon.

This parameter is visible only if Quantize phase is selected.

Output signal

Choose whether the block should output a Sine, Cosine, Complex exponential, or Sine and cosine signals. If you select Sine and cosine, the two signals output on different ports.

Sample time

Specify the sample time in seconds when the block is acting as a source. When either the phase increment or phase offset come in via block input ports, the sample time is inherited and this parameter is not visible.

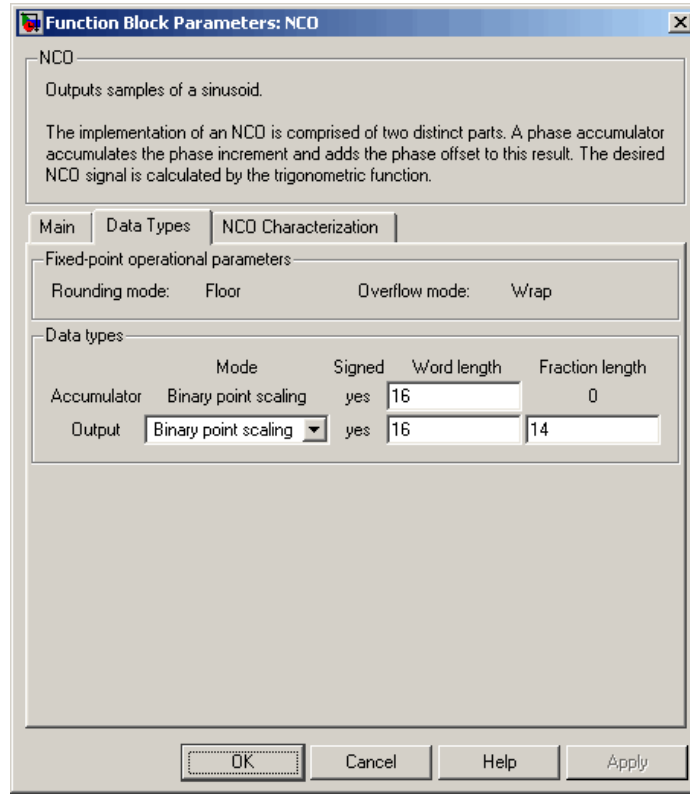
Samples per frame

Specify the number of samples per frame. When the value of this parameter is 1, the block outputs a sample-based signal. When the value is greater than 1, the block outputs a frame-based signal of the specified size. In frame-based mode, the phase increment and phase offset can vary from channel to channel and from frame to frame, but they are constant along each channel in a given frame.

When the phase offset input port exists, it has the same frame status as any output port prese nt. When the phase increment input port exists, it does not support frames.

This parameter is only visible if either Phase increment source or Phase offset source is set to Specify via dialog .

The Data Types pane of the NCO dialog appears as follows.



Rounding mode

The rounding mode used for this block when inputs are fixed point is always Floor.

Overflow mode

The overflow mode used for this block when inputs are fixed point is always Wrap.

Accumulator

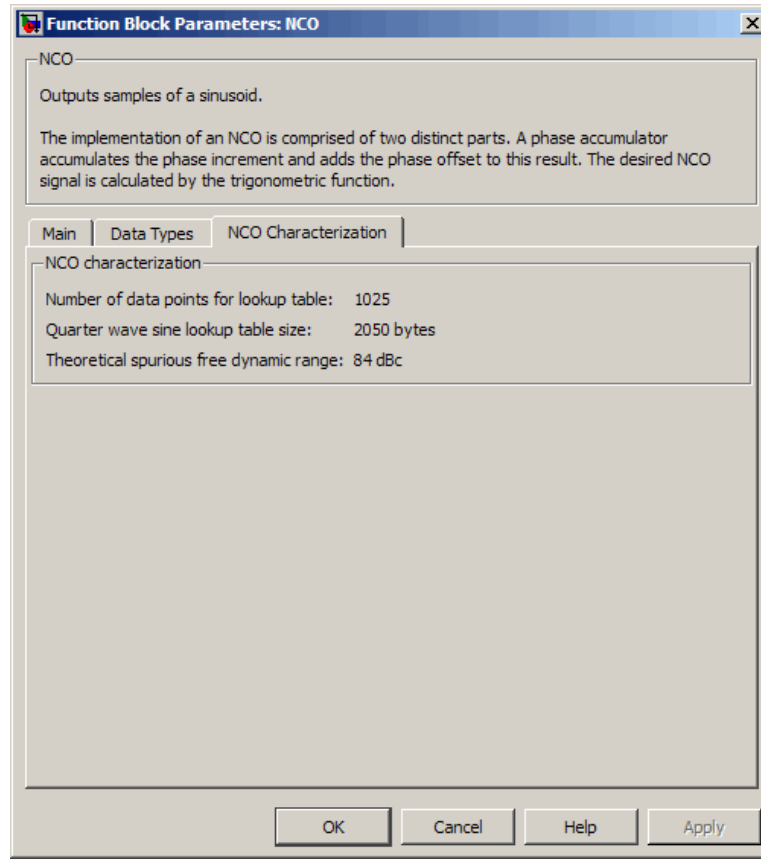
Specify the word length of the accumulator data type. The fraction length is always zero; this is an integer data type.

Output

Specify the output data type.

- Choose double or single for a floating-point implementation.
- When you select Binary point scaling, you can enter the word length and the fraction length of the output, in bits.

The NCO Characterization pane of the NCO dialog appears as follows.



The NCO Characterization pane does not have any parameters. Instead, it provides you with details on the NCO signal currently being implemented by the block:

- Number of data points for lookup table — The lookup table is implemented as a quarter-wave sine table. The number of lookup table data points is defined by

$$2^{\text{number of quantized accumulator bits}-2}+1$$

- Quarter wave sine lookup table size — The quarter wave sine lookup table size is de Þned by

$$\frac{(\text{number of data points for lookup table}) \cdot (\text{output word length})}{8} \text{ bytes}$$

- Theoretical spurious free dynamic range — The spurious free dynamic range (SFDR) is calculated as follows for a lookup table with 2^P entries:

$$SFDR = (6P) \text{ dB} \quad \text{without dither}$$
$$SFDR = (6P + 12) \text{ dB} \quad \text{with dither}$$

Supported Data Types

Port	Supported Data Types
inc	• Fixed point with zero fraction length • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
offset	• Fixed point with zero fraction length • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
sin	• Double-precision ßoating point • Single-precision ßoating point • Fixed point
Qerr	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

PN Sequence Generator	Communications Blockset
Sine Wave	Signal Processing Blockset

Normalization

Purpose

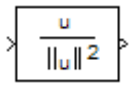
Perform vector normalization along rows, columns, or specified dimension

Library

Math Functions / Math Operations

dspmathops

Description



The Normalization block independently normalizes each row, column, or vector of the specified dimension of the input. The Normalization block accepts real and complex floating-point and fixed-point inputs except for complex unsigned fixed-point inputs. The block only accepts floating-point signals for the 2-norm mode, and both fixed-point and floating-point signals for the squared 2-norm mode. The output always has the same dimensions and frame status as the input.

This block treats an arbitrarily dimensioned input U as a collection of vectors oriented along the specified dimension. The block normalizes these vectors by either their norm or the square of their norm.

For example, consider a 3-dimensional input U(i,j,k) and assume that you want to normalize along the second dimension. First, define the 2-dimensional intermediate quantity V(i,k) such that each element of V is the norm of one of the vectors in U:

$$V(i,k) = \left(\sum_{j=1}^{J} U^2(i,j,k) \right)^{1/2}$$

Given V, the output of the block Y(i, j,k) in 2-norm mode is

$$Y(i,j,k) \frac{U(i,j,k)}{V(i,k) + b}$$

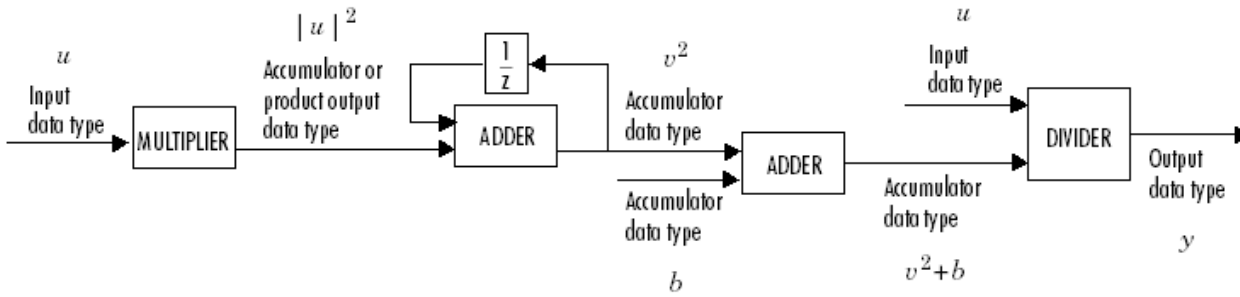
In squared 2-norm mode, the block output is

$$Y(i,j,k) = \frac{U(i,j,k)}{V(i,k)^2 + b}$$

The normalization bias, b, is typically chosen to be a small positive constant (for example, 1e-10) that prevents potential division by zero.

Fixed-Point Data Types

The following diagram shows the data types used within the Normalization block for fixed-point signals (squared 2-norm mode only).



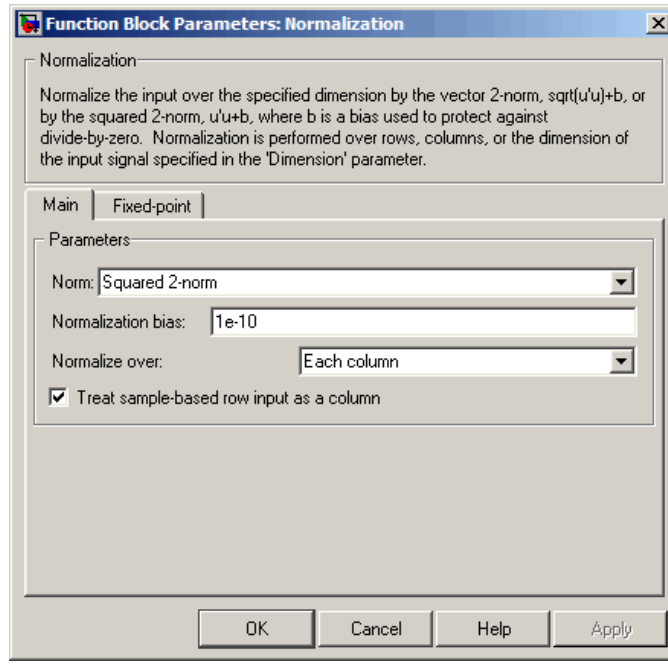
The output of the multiplier is in the product output data type when the input is real. When the input is complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, see "Multiplication Data Types". You can set the accumulator, product output, and output data types in the block dialog as discussed in "Dialog Box" on page 2-964.

Examples

See "Zero Algorithmic Delay" in the Signal Processing Blockset User's Guide for an example.

Dialog Box

The Main pane of the Normalization dialog appears as follows.



Norm

Specify the type of normalization to perform, 2-norm or Squared 2-norm. 2-norm mode supports floating-point signals only. Squared 2-norm supports both fixed-point and floating-point signals.

Normalization bias

Specify the real value b to be added in the denominator to avoid division by zero. Tunable.

Normalize over

Specify whether to normalize along rows, columns, or the dimension specified in the Dimension parameter.

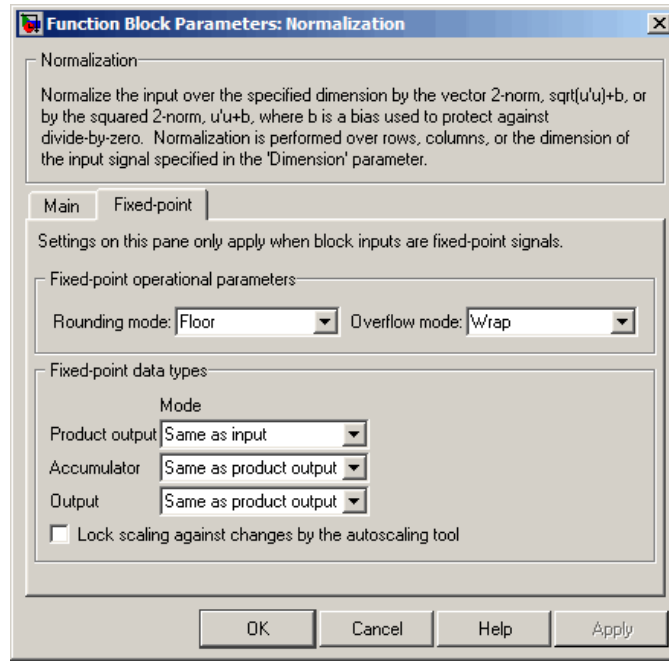
Dimension

Specify the one-based value of the dimension over which to normalize. The value of this parameter cannot exceed the number of dimensions in the input signal. This parameter is only visible if Specified dimension is selected for the Normalize over parameter.

Treat sample-based row input as column

Select to treat a sample-based row input as a column.

The Fixed-Point pane of the Normalization dialog appears as follows.



Note The parameters on this pane are only applicable to fixed-point signals when the block is in squared 2-norm mode. See “Fixed-Point Data Types” on page 2-963 for a diagram of how the product output, accumulator, and output data types are used in this case.

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths:

- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.
- When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block. The bias b is also quantized into the accumulator data type:

- When you select Same as product output, these characteristics match those of the product output.
- When you select Same as input, these characteristics match those of the input to the block.
- When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.

- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length:

- When you select Same as accumulator , these characteristics match those of the accumulator.
- When you select Same as product output , these characteristics match those of the product output.
- When you select Same as input , these characteristics match those of the input to the block.
- When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.
- When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any Þxed-point scaling you specify in this block mask f rom being overridden by the autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Array-Vector Multiply	Signal Processing Blockset
Reciprocal Condition	Signal Processing Blockset
norm	MATLAB

Purpose

Design Nyquist filter

Library

Filtering / Filter Design Toolbox

dspfdesign

Description

This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See “Nyquist Filter Design Dialog Box — Main Pane” in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset product.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point

Port	Supported Data Types
	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Purpose

Design octave filter

Library

Filtering / Filter Design Toolbox

dspfdesign

Description

This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See “Octave Filter Design Dialog Box — Main Pane” in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset product.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point

Port	Supported Data Types
	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Purpose

Truncate vectors by removing or k eeping beginning or ending values

Library

Signal Operations

dspsigops

Description

The Offset block removes or keeps values from the beginning or end of a vector and outputs the result in a vector of user-speci Þed length. The inputs to the In ports (In1, In2, ...) can be scalars or vectors, but they must be the same size and data type. The offset values are the inputs to the O ports (O1, O2, ...); they must be scalar values with the same data type. These offset values should be integer values because they determine the number of values the block discards or retains from each input vector. The block rounds any offset value that is a noninteger value to the nearest integer value. There is one output port for each pair of In and O ports. This block supports sample-based and frame-based signals.

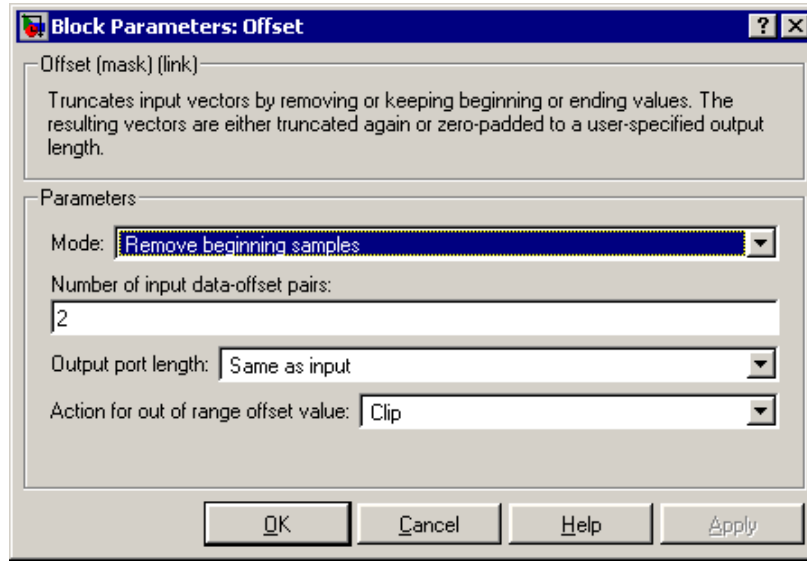
Use the Mode parameter to determine which values the block discards or retains from the input vector. To discard the initial values of the vector, select Remove beginning samples . To discard the Þnal values of the vector, select Remove ending samples . To retain the initial values of the vector, select Keep beginning samples . To retain the Þnal values of a vector, select Keep ending samples.

Use the Number of input data-offset pairs parameter to specify the number of inputs to the block. The number of input ports is twice the scalar value you enter. For example, if you enter 3, ports In1, O1, In2, O2, In3, and O3 appear on the block.

The block uses the Output port length parameter to determine the length of the output vectors. If you select Same as input , the block outputs vectors that are the same length as the input to the In ports. If you select User-defined , the Output length parameter appears. Enter a scalar that represents the desired length of the output vectors. If your desired output length is greater than the number of values you extracted from your input vector, the block zero-pads the end of the vector to reach the length you speci Þed.

Use the Action for out of range offset value parameter to determine how the block behaves when an offset value is not in the range 0 ≤ offset value ≤ N, where N is the input vector length. Select Clip if you want any offset values less than 0 to be set to 0 and any offset values greater than N to be set to N. Select Clip and warn if you want to be warned when any offset values less than 0 are set to 0 and any offset values greater than N are set to N. Select Error if you want the simulation to stop and display an error when the offset values are out of range.

Dialog Box



Mode

Use this parameter to determine which values the block discards or retains from the input vector. Your choices are Remove beginning samples, Remove ending samples, Keep beginning samples, and Keep ending samples.

Number of input data-offset pairs

Specify the number of inputs to the block. The number of input ports is twice the scalar value you enter.

Output port length

Use this parameter to specify the length of the output vectors. If you select Same as input , the output vectors are the same length as the input vectors. If you select User-defined , you can enter the desired length of the output vectors.

Output length

Enter a scalar that represents the desired length of the output vectors. This parameter is visible if, for the Output port length parameter, you select User-defined .

Action for out of range offset value

Use this parameter to determine how the block behaves when an offset value is not in the range such that $0 \leq$ offset value $\leq N$, where N is the input vector length. When you want any offset values less than 0 to be set to 0 and any offset values greater than N to be set to N, select Clip . When you want to be warned when any offset values less than 0 are set to 0 and any offset values greater than N are set to N, select Clip and warn . When you want the simulation to stop and display an error when the offset values are out of range, select Error .

Supported Data Types

Port	Supported Data Types
In	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
O	• Double-precision ßoating point • Single-precision ßoating point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Out	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

Purpose

Implement overlap-add method of frequency-domain filtering

Library

Filtering / Filter Designs

dsparch4

Description

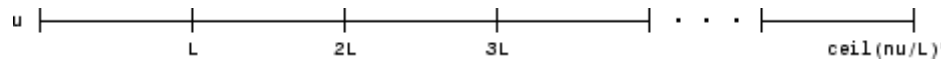
The Overlap-Add FFT Filter block uses an FFT to implement the overlap-add method, a technique that combines successive frequency-domain filtered sections of an input sequence.

Valid inputs to this block are 1-D vectors, sample-based vectors, frame-based vectors, and frame-based full matrices. All outputs are unbuffered into sample-based row vectors. The length of the output vector is equal to the number of channels in the input vector. An M-by-1 sample-based input has M channels, so it would result in a length-M sample-based output vector. An M-by-1 frame-based input has only one channel, so would result in a 1-by-1 (scalar) output.

The block's data output rate is M times faster than its data input rate, where M is the input frame-size. Thus, the block's data input and output rates are the same when the inputs are 1-D vectors, sample-based vectors, or frame-based row vectors. For frame-based column and frame-based full-matrix inputs, the block's data output rate is M times greater than the block's data input rate.

1-D vectors are treated as length-N sample-based vectors, and result in sample-based length-N row vectors.

The block breaks the scalar input sequence `u`, of length `nu`, into length-`L` nonoverlapping data sections,



which it linearly convolves with the filter's FIR coefficients,

$$H(z) = B(z) = b_1 + b_2 z^{-1} + \ldots + b_{n+1} z^{-n}$$

The numerator coefficients for H(z) are specified as a vector by the FIR coefficients parameter. The coefficient vector, `b = [b(1) b(2) ...`

b(n+1)] , can be generated by one of the filter design functions in the Signal Processing Toolbox product, such as fir1 . All filter states are internally initialized to zero.

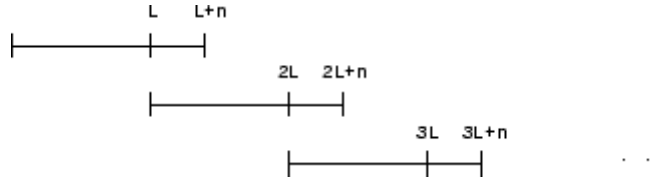
When either the filter coefficients or the inputs to the block are complex, the Output parameter should be set to Complex . Otherwise, the default Output setting, Real , instructs the block to take only the real part of the solution.

The block's overlap-add operation is equivalent to

```
y = ifft(fft(u(i:i+L-1),nfft) .* fft(b,nfft))
```

where you specify nfft in the FFT size parameter as a power-of-two value greater (typically much greater) than n+1. Values for FFT size that are not powers of two are rounded upwards to the nearest power-of-two value to obtain nfft .

The block overlaps successive output sections by n points and sums them.



The first L samples of each summation are output in sequence. The block chooses the parameter L based on the filter order and the FFT size.

```
L = nfft - n
```

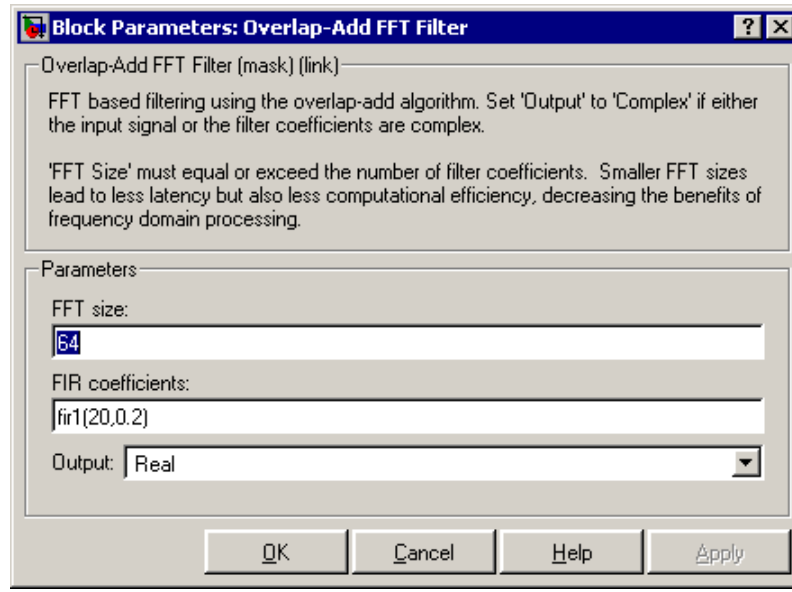
Latency

In single-tasking operation, the Overlap-Add FFT Filter block has a latency of nfft-n+1 samples. The first nfft-n+1 consecutive outputs from the block are zero; the first filtered input value appears at the output as sample nfft-n+2 .

In multitasking operation, the Overlap-Add FFT Filter block has a latency of 2*(nfft-n+1) samples. The first 2*(nfft-n+1) consecutive outputs from the block are zero; the first filtered input value appears at the output as sample 2*(nfft-n)+3 .

Note For more information on latency and the Simulink software tasking modes, see "Excess Algorithmic Delay (Tasking Latency)" and "Models with Multiple Sample Rates" in the Real-Time Workshop User's Guide.

Dialog Box



FFT size

The size of the FFT, which should be a power-of-two value greater than the length of the specified FIR filter.

FIR coefficients

The filter numerator coefficients.

Output

The complexity of the output; Real or Complex . When the input signal or the filter coefficients are complex, this should be set to Complex .

References

Oppenheim, A. V. and R. W. Schafer. Discrete-Time Signal Processing . Englewood Cliffs, NJ: Prentice Hall, 1989.

Proakis, J. and D. Manolakis. Digital Signal Processing. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

Overlap-Save FFT Filter Signal Processing Blockset product

Purpose

Implement overlap-save method of frequency-domain filtering

Library

Filtering / Filter Designs

dsparch4

Description

The Overlap-Save FFT Filter block uses an FFT to implement the overlap-save method, a technique that combines successive frequency-domain filtered sections of an input sequence.

Valid inputs to this block are 1-D vectors, sample-based vectors, frame-based vectors, and frame-based full matrices. All outputs are unbuffered into sample-based row vectors. The length of the output vector is equal to the number of channels in the input vector. An M-by-1 sample-based input has M channels, so it would result in a length-M sample-based output vector. An M-by-1 frame-based input has only one channel, so would result in a 1-by-1 (scalar) output.

The block's data output rate is M times faster than its data input rate, where M is the input frame-size. Thus, the block's data input and output rates are the same when the inputs are 1-D vectors, sample-based vectors, or frame-based row vectors. For frame-based column and frame-based full-matrix inputs, the block's data output rate is M times greater than the block's data input rate.

1-D vectors are treated as length-N sample-based vectors, and result in sample-based length-N row vectors.

Overlapping sections of input u are circularly convolved with the FIR filter coefficients

$$H(z) = B(z) = b_1 + b_2 z^{-1} + \ldots + b_{n+1} z^{-n}$$

The numerator coefficients for H(z) are specified as a vector by the FIR coefficients parameter. The coefficient vector, b = [b(1) b(2) ... b(n+1)], can be generated by one of the filter design functions in the Signal Processing Toolbox product, such as fir1. All filter states are internally initialized to zero.

When either the filter coefficients or the inputs to the block are complex, the Output parameter should be set to Complex. Otherwise, the default Output setting, Real, instructs the block to take only the real part of the solution.

The circular convolution of each section is computed by multiplying the FFTs of the input section and filter coefficients, and computing the inverse FFT of the product.

```
y = ifft(fft(u(i:i+(L-1)),nfft) .* fft(b,nfft))
```

where you specify nfft in the FFT size parameter as a power of two value greater (typically much greater) than n+1. Values for FFT size that are not powers of two are rounded upwards to the nearest power-of-two value to obtain nfft.

The first n points of the circular convolution are invalid and are discarded. The Overlap-Save FFT Filter block outputs the remaining nfft-n points, which are equivalent to the linear convolution.

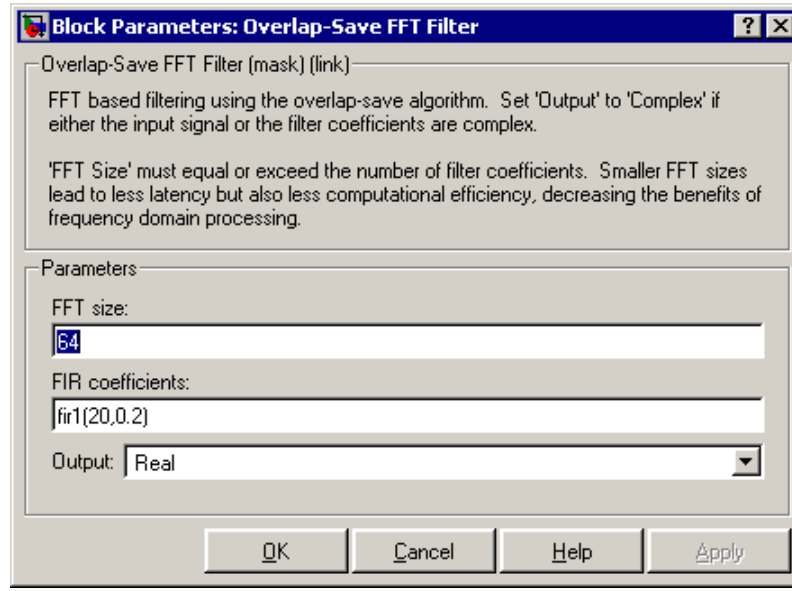
Latency

In single-tasking operation, the Overlap-Save FFT Filter block has a latency of nfft-n+1 samples. The first nfft-n+1 consecutive outputs from the block are zero; the first filtered input value appears at the output as sample nfft-n+2.

In multitasking operation, the Overlap-Save FFT Filter block has a latency of 2*(nfft-n+1) samples. The first 2*(nfft-n+1) consecutive outputs from the block are zero; the first filtered input value appears at the output as sample 2*(nfft-n)+3.

Note For more information on latency and the Simulink environment tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Models with Multiple Sample Rates” in the Real-Time Workshop User’s Guide.

Dialog Box



FFT size

The size of the FFT, which should be a power of two value greater than the length of the specified FIR filter.

FIR coefficients

The filter numerator coefficients.

Output

The complexity of the output; `Real` or `Complex`. When the input signal or the filter coefficients are complex, this should be set to `Complex`.

References

Oppenheim, A. V. and R. W. Schafer. Discrete-Time Signal Processing. Englewood Cliffs, NJ: Prentice Hall, 1989.

Proakis, J. and D. Manolakis. Digital Signal Processing. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

Overlap-Add FFT Filter

Signal Processing Blockset

Purpose

Overwrite submatrix or subdiagonal of input

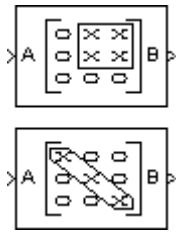
Library

- Math Functions / Matrices and Lin ear Algebra / Matrix Operations

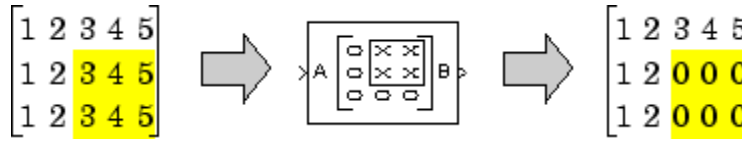
  dspmtrx3
- Signal Management / Indexing

  dspindex

Description



The Overwrite Values block overwrites a contiguous submatrix or subdiagonal of an input matrix. You can provide the overwriting values by typing them in a block paramete r, or through an additional input port, which is useful for providing overwriting values that change at each time step.



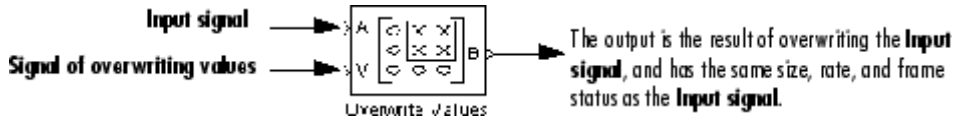
The block accepts both sample- and frame-based vectors and matrices. The output has the same size and frame status as the original input signal, not necessarily the same size and frame status as the signal containing the overwriting values. The input(s) and output of this block must have the same data type.

Specifying the Overwriting Values

The Source of overwriting value(s) parameter determines how you must provide the overwriting values, and has the following settings.

- Specify via dialog — You must provide the overwriting value(s) in the Overwrite with parameter. The block uses the same overwriting values to overwrite the speci Þed portion of the input at each time step. To learn how to speci fy valid overwriting values, see “Valid Overwriting Values” on page 2-986.
- Second input port — You must provide overwriting values through a second block input port, V. Use this setting to provide different

overwriting values at each time step. The output inherits its size, rate, and frame status from the input signal, not the overwriting values.



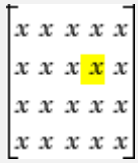
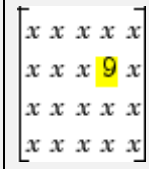
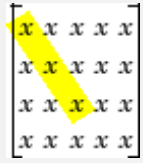
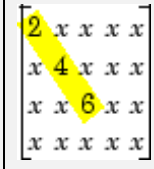
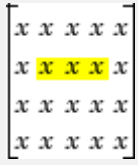
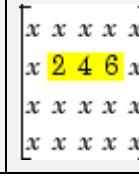
The rate at which you provide the overwriting values through input port V must match the rate at which the block receives each input matrix at input port A. The rate requirements depend on whether the input signal and overwriting values signal have the same frame status:

- When both signals are sample based, their sample rates must be the same.
- When both signals are frame based, their frame rates must be the same.
- When one signal is sample based and one signal is frame based, the sample rate of the sample-based signal must be the same as the frame rate of the frame-based signal.

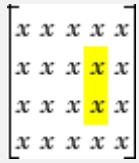
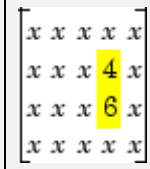
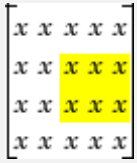
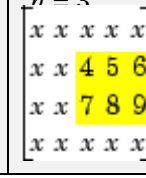
Valid Overwriting Values

The overwriting values can be a single constant, vector, or matrix, depending on the portion of the input you are overwriting, regardless of whether you provide the overwriting values through an input port or by providing them in the Overwrite with parameter.

Valid Overwriting Values

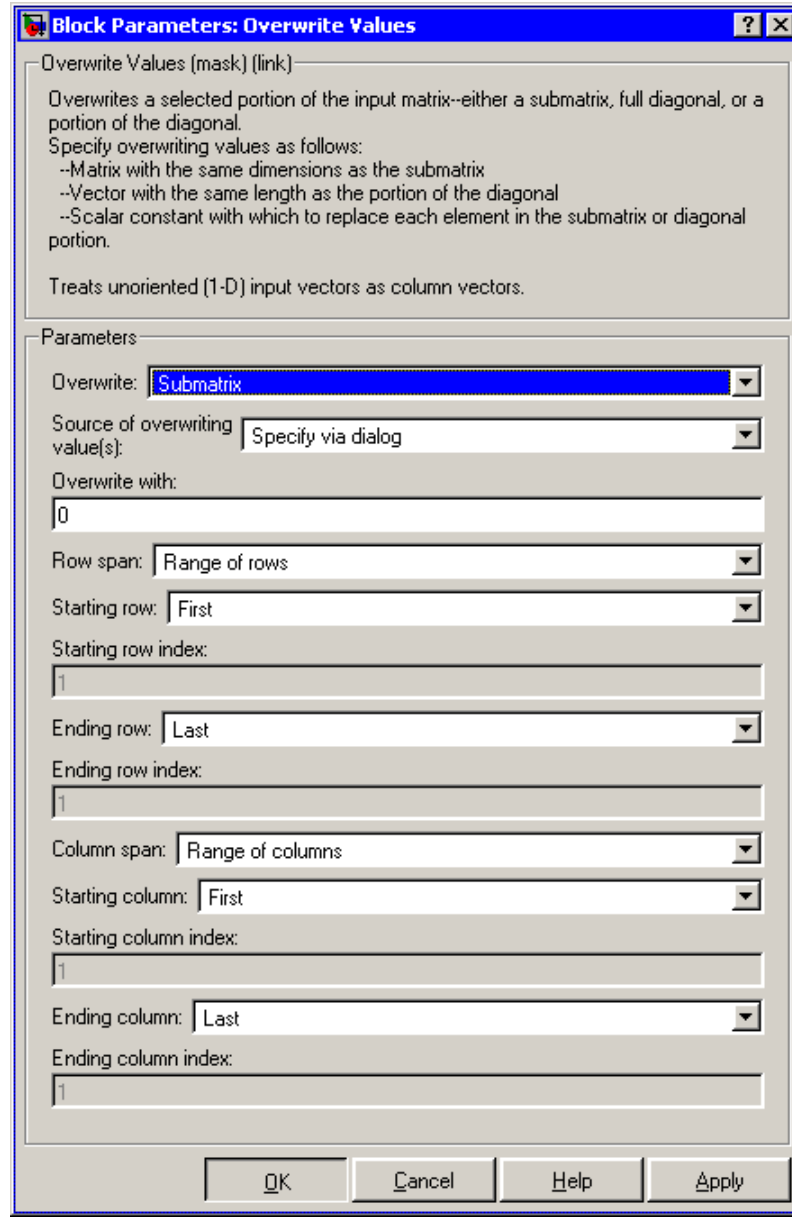
Portion of Input to Overwrite	Valid Overwriting Values	Example
A single element in the input 	Any constant value, v	v = 9 
A length- k portion of the diagonal 	Any length- k column or row vector, v	$k = 3 \quad v = [2 \quad 4 \quad 6]$ or $\begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$ 
A length- k portion of a row 	Any length- k row vector, v	$k = 3 \quad v = [2 \quad 4 \quad 6]$ 

Valid Overwriting Values (Continued)

Portion of Input to Overwrite	Valid Overwriting Values	Example
A length- k portion of a column 	Any length- k column vector, v	$k = 2 \qquad v = \begin{bmatrix} 4 \\ 6 \end{bmatrix}$ 
An m-by-n submatrix 	Any m-by-n matrix, v	$m = 2$ $n = 3$ $v = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ 

This block supports Simulink virtual buses.

Dialog Box


Block Parameters: Overwrite Values
Overwrite Values (mask) (link)
Overwrites a selected portion of the input matrix--either a submatrix, full diagonal, or a portion of the diagonal.
Specify overwriting values as follows:
--Matrix with the same dimensions as the submatrix
--Vector with the same length as the portion of the diagonal
--Scalar constant with which to replace each element in the submatrix or diagonal portion.
Treats unoriented (1-D) input vectors as column vectors.
Parameters
Overwrite: Submatrix
Source of overwriting value(s): Specify via dialog
Overwrite with:
0
Row span: Range of rows
Starting row: First
Starting row index:
1
Ending row: Last
Ending row index:
1
Column span: Range of columns
Starting column: First
Starting column index:
1
Ending column: Last
Ending column index:
1
OK
Cancel
Help
Apply

Note Only some of the following parameters are visible in the dialog box at any one time.

Overwrite
: Determines whether to overwrite a specified submatrix or a specified portion of the diagonal.

Source of overwriting value(s)
: Determines where you must provide the overwriting values: either through an input port, or by providing them in the Overwrite with parameter. For more information, see “Specifying the Overwriting Values” on page 2-985.

Overwrite with
: The value(s) with which to overwrite the specified portion of the input matrix. Enabled only when Source of overwriting value(s) is set to Specify via dialog. To learn how to specify valid overwriting values, see “Valid Overwriting Values” on page 2-986.

Row span
: The range of input rows to be overwritten. Options are All rows, One row, or Range of rows. For descriptions of these options, see “Dialog Box” on page 2-989.

Row/Starting row
: The input row that is the first row of the submatrix that the block overwrites. For a description of the options for the Row and Starting row parameters, see Settings for Row, Column, Starting Row, and Starting Column Parameters on page 2-995. Row is enabled when Row span is set to One row, and Starting row when Row span is set to Range of rows.

Row index/Starting row index
: Index of the input row that is the first row of the submatrix that the block overwrites. See how to use these parameters in Settings for Row, Column, Starting Row, and Starting Column Parameters

on page 2-995. Row index is enabled when Row is set to Index, and Starting row index when Starting row is set to Index.

Row offset/Starting row offset

The offset of the input row that is the first row of the submatrix that the block overwrites. See how to use these parameters in Settings for Row, Column, Starting Row, and Starting Column Parameters on page 2-995. Row offset is enabled when Row is set to Offset from middle or Offset from last, and Starting row offset is enabled when Starting row is set to Offset from middle or Offset from last.

Ending row

The input row that is the last row of the submatrix that the block overwrites. For a description of this parameter's options, see Settings for Ending Row and Ending Column Parameters on page 2-996. This parameter is enabled when Row span is set to Range of rows, and Starting row is set to any option but Last.

Ending row index

Index of the input row that is the last row of the submatrix that the block overwrites. See how to use this parameter in Settings for Ending Row and Ending Column Parameters on page 2-996. Enabled when Ending row is set to Index.

Ending row offset

The offset of the input row that is the last row of the submatrix that the block overwrites. See how to use this parameter in Settings for Ending Row and Ending Column Parameters on page 2-996. Enabled when Ending row is set to Offset from middle or Offset from last.

Column span

The range of input columns to be overwritten. Options are All columns, One column, or Range of columns. For descriptions of the analogous row options, see "Dialog Box" on page 2-989.

Column/Starting column

The input column that is the first column of the submatrix that the block overwrites. For a description of the options for the

Column and Starting column parameters, see Settings for Row, Column, Starting Row, and Starting Column Parameters on page 2-995. Column is enabled when Column span is set to One column, and Starting column when Column span is set to Range of columns.

Column index/Starting column index
: Index of the input column that is the first column of the submatrix that the block overwrites. See how to use these parameters in Settings for Row, Column, Starting Row, and Starting Column Parameters on page 2-995. Column index is enabled when Column is set to Index, and Starting column index when Starting column is set to Index.

Column offset/Starting column offset
: The offset of the input column that is the first column of the submatrix that the block overwrites. See how to use these parameters in Settings for Row, Column, Starting Row, and Starting Column Parameters on page 2-995. Column offset is enabled when Column is set to Offset from middle or Offset from last, and Starting column offset is enabled when Starting column is set to Offset from middle or Offset from last.

Ending column
: The input column that is the last column of the submatrix that the block overwrites. For a description of this parameter's options, see Settings for Ending Row and Ending Column Parameters on page 2-996. This parameter is enabled when Column span is set to Range of columns, and Starting column is set to any option but Last.

Ending column index
: Index of the input column that is the last column of the submatrix that the block overwrites. See how to use this parameter in Settings for Ending Row and Ending Column Parameters on page 2-996. This parameter is enabled when Ending column is set to Index.

Ending column offset

The offset of the input column that is the last column of the submatrix that the block overwrites. See how to use this parameter in Settings for Ending Row and Ending Column Parameters on page 2-996. This parameter is enabled when Ending column is set to Offset from middle or Offset from last.

Diagonal span

The range of diagonal elements to be overwritten. Options are All elements, One element, or Range of elements. For descriptions of these options, see "Overwriting a Subdiagonal" on page 2-999.

Element/Starting element

The input diagonal element that is the first element in the subdiagonal that the block overwrites. For a description of the options for the Element and Starting element parameters, see Element and Starting Element Parameters on page 2-999. Element is enabled when Element span is set to One element, and Starting element when Element span is set to Range of elements.

Element index/Starting element index

Index of the input diagonal element that is the first element of the subdiagonal that the block overwrites. See how to use these parameters in Element and Starting Element Parameters on page 2-999. Element index is enabled when Element is set to Index, and Starting element index when Starting element is set to Index.

Element offset/Starting element offset

The offset of the input diagonal element that is the first element of the subdiagonal that the block overwrites. See how to use these parameters in Element and Starting Element Parameters on page 2-999. Element offset is enabled when Element is set to Offset from middle or Offset from last, and Starting element offset is enabled when Starting element is set to Offset from middle or Offset from last.

Ending element
: The input diagonal element that is the last element of the subdiagonal that the block overw rites. For a description of this parameter's options, see Ending Element Parameters on page 2-1000. This parameter is enabled when Element span is set to Range of elements , and Starting element is set to any option but Last .

Ending element index
: Index of the input diagonal element that is the last element of the subdiagonal that the block overwrites. See how to use this parameter in Ending Element Parameters on page 2-1000. This parameter is enabled when Ending element is set to Index .

Ending element offset
: The offset of the input diagonal element that is the last element of the subdiagonal that the block overwrites. See how to use this parameter in Ending Element Parameters on page 2-1000. This parameter is enabled when Ending element is set to Offset from middle or Offset from last .

Examples

Overwriting a Submatrix

To overwrite a submatrix, follow these steps:

1. Set the Overwrite parameter to Submatrix .
2. Specify the overwriting values as described in "Specifying the Overwritin g Values" on page 2-985.
3. Specify which rows and columns of the input matrix are contained in the submatrix that you want to overwrite by setting the Row span parameter to one of the fol lowing options and the Column span to the analogous column-related options:
   - All rows — The submatrix contains all rows of the input matrix.
   - One row — The submatrix contains only one row of the input matrix, which you must specify in the Row parameter, as described in the following table.

- Range of rows — The submatrix contains one or more rows of the input, which you must specify in the Starting Row and Ending row parameters, as described in the following tables.

4 When you set Row span to One row or Range of rows, you need to further specify the row(s) contained in the submatrix by setting the Row or Starting row and Ending row parameters. Likewise, when you set Column span to One column or Range of columns, you must further specify the column(s) contained in the submatrix by setting the Column or Starting column and Ending column parameters. For descriptions of the settings for these parameters, see the following tables.

Settings for Row, Column, Starting Row, and Starting Column Parameters

Settings for Specifying the Submatrix's First Row or Column	First Row of Submatrix (Only row for Row span = One row)	First Column of Submatrix (Only row for Row span = One row)
First	First row of the input	First column of the input
Index	Input row specified in the Row index parameter	Input column specified in the Column index parameter
Offset from last	Input row with the index M - rowOffset where M is the number of input rows, and rowOffset is the value of the Row offset or Starting row offset parameter	Input column with the index N - colOffset where N is the number of input columns, and colOffset is the value of the Column offset or Starting column offset parameter
Last	Last row of the input	Last column of the input

Settings for Row, Column, Starting Row, and Starting Column Parameters (Continued)

Settings for Specifying the Submatrix's First Row or Column	First Row of Submatrix (Only row for Row span = One row)	First Column of Submatrix (Only row for Row span = One row)
Offset from middle	Input row with the index floor(M/2 + 1 - rowOffset) where M is the number of input rows, and rowOffset is the value of the Row offset or Starting row offset parameter	Input column with the index floor(N/2 + 1 - rowOffset) where N is the number of input columns, and colOffset is the value of the or Column offset or Starting column offset parameter
Middle	Input row with the index floor(M/2 + 1) where M is the number of input rows	Input columns with the index floor(N/2 + 1) where N is the number of input columns

Settings for Ending Row and Ending Column Parameters

Settings for Specifying the Submatrix's Last Row or Column	Last Row of Submatrix	Last Column of Submatrix
Index	Input row specified in the Ending row index parameter	Input column specified in the Ending column index parameter

Settings for Ending Row and Ending Column Parameters (Continued)

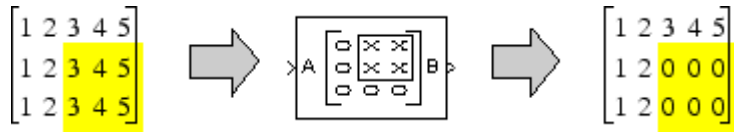
Settings for Specifying the Submatrix's Last Row or Column	Last Row of Submatrix	Last Column of Submatrix
Offset from last	Input row with the index M - rowOffset where M is the number of input rows, and rowOffset is the value of the Ending row offset parameter	Input column with the index N - colOffset where N is the number of input columns, and colOffset is the value of the Ending column offset parameter
Last	Last row of the input	Last column of the input
Offset from middle	Input row with the index floor(M/2 + 1 - rowOffset) where M is the number of input rows, and rowOffset is the value of the Ending row offset parameter	Input column with the index floor(N/2 + 1 - rowOffset) where N is the number of input columns, and colOffset is the value of the Ending column offset parameter
Middle	Input row with the index floor(M/2 + 1) where M is the number of input rows	Input columns with the index floor(N/2 + 1) where N is the number of input columns

For example, to overwrite the lower-right 2-by-3 submatrix of a 3-by-5 input matrix with all zeros, enter the following set of parameters:

- Overwrite = Submatrix
- Source of overwriting value(s) = Specify via dialog
- Overwrite with = 0
- Row span = Range of rows
- Starting row = Index

- Starting row index = 2
- Ending row = Last
- Column span = Range of columns
- Starting column = Offset from last
- Starting column offset = 2
- Ending column = Last

The following figure shows the block with the above settings overwriting a portion of a 3-by-5 input matrix.



There are often several possible parameter combinations that select the same submatrix from the input. For example, instead of specifying Last for Ending column, you could select the same submatrix by specifying

- Ending column = Index
- Ending column index = 5

Overwriting a Subdiagonal

To overwrite a subdiagonal, follow these steps:

1 Set the Overwrite parameter to Diagonal.

2 Specify the overwriting values as described in “Specifying the Overwriting Values” on page 2-985.

3 Specify the subdiagonal that you want to overwrite by setting the Diagonal span parameter to one of the following options:

- All elements — Overwrite the entire input diagonal.
- One element — Overwrite one element in the diagonal, which you must specify in the Element parameter (described below).
- Range of elements — Overwrite a portion of the input diagonal, which you must specify in the Starting element and Ending element parameters, as described in the following table.

4 When you set Diagonal span to One element or Range of elements, you need to further specify which diagonal element(s) to overwrite by setting the Element or Starting element and Ending element parameters. See the following tables.

Element and Starting Element Parameters

Settings for Element and Starting Element Parameters	First Element in Subdiagonal (Only element when Diagonal span = One element)
First	Diagonal element in first row of the input
Index	kth diagonal element, where k is the value of the Element index or Starting element index parameter

Element and Starting Element Parameters (Continued)

Settings for Element and Starting Element Parameters	First Element in Subdiagonal (Only element when Diagonal span = One element)
Offset from last	Diagonal element in the row with the index M- offset where M is the number of input rows, and offset is the value of the Element offset or Starting element offset parameter
Last	Diagonal element in the last row of the input
Offset from middle	Diagonal element in the input row with the index floor(M/2 + 1 - offset) where M is the number of input rows, and offset is the value of the Element offset or Starting element offset parameter
Middle	Diagonal element in the input row with the index floor(M/2 + 1) where M is the number of input rows

Ending Element Parameters

Settings for Ending Element Parameter	Last Element in Subdiagonal
Index	kth diagonal element, where k is the value of the Ending element index parameter
Offset from last	Diagonal element in the row with the index M- offset where M is the number of input rows, and offset is the value of the Ending element offset parameter
Last	Diagonal element in the last row of the input

Ending Element Parameters (Continued)

Settings for Ending Element Parameter	Last Element in Subdiagonal
Offset from middle	Diagonal element in the input row with the index floor(M/2 + 1 - offset) where M is the number of input rows, and offset is the value of the Ending element offset parameter
Middle	Diagonal element in the input row with the index floor(M/2 + 1) where M is the number of input rows

Supported Data Types

The input(s) and output of this block must have the same data type.

Port	Supported Data Types
A	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Port	Supported Data Types
V	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
B	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Reshape	Simulink
Selector	Simulink
Submatrix	Signal Processing Blockset
Variable Selector	Signal Processing Blockset
reshape	MATLAB

Purpose

Pad or truncate specified dimension(s)

Library

Signal Operations

dspsigops

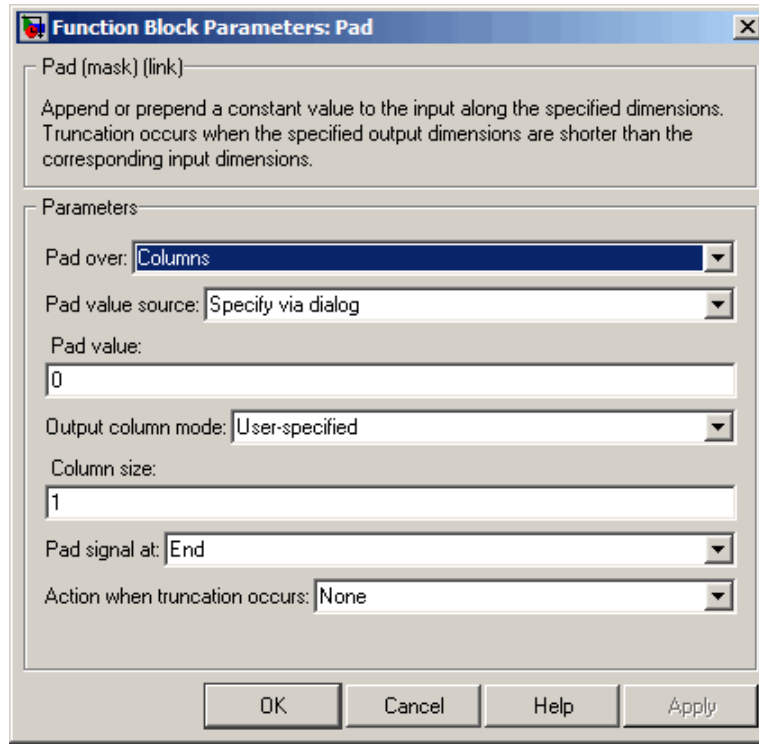
Description

The Pad block extends or crops the dimensions of the input by padding or truncating along its columns, rows, columns and rows, or any dimension(s) you specify. Truncation occurs when you specify output dimensions that are shorter than the corresponding input dimensions. If the input and output lengths are the same, the block is a pass-through.

You can enter the pad value in the block mask or via an input port. You can enter output sizes in the block mask, or have the block pad the specified dimensions until their length is the next highest power of two. The Pad signal at parameter controls whether the specified input dimensions are padded or truncated at their beginning, end, or both. For odd pad or truncation lengths, the extra pad value or truncation is applied to the end of the signal. When the block is in Specified dimensions mode, you can specify either the output size or the pad size.

You can have the block warn or error when an input signal is truncated using the Action when truncation occurs parameter.

Dialog Box



Pad over

Specify the dimensions over which to pad or truncate: Columns, Rows, Columns and rows, None, or Specified dimensions.

Dimensions to pad

Specify the one-based dimension(s) over which to pad or truncate. The value for this parameter can be a scalar or a vector. For example, specify 1 to pad columns. Specify [1 2] to pad columns and rows. Specify [1 3 5] to pad the first, third, and fifth dimensions.

This parameter is only visible when Specified dimensions is selected for the Pad over parameter.

Pad value source

Choose how you specify the pad value. The pad value can come from an input port or from the dialog:

- If you select Input port , the PVal port appears on the block icon.
- If you select Specify via dialog , the Pad value parameter appears.

Pad value

Specify the constant scalar value with which to pad the input. Tunable.

This parameter is only visible when Specify via dialog is selected for the Pad value source parameter.

Output column mode

Choose how you specify the column length of the output:

- If you select User-specified , the Column size parameter appears.
- If you select Next power of two , the block pads the output columns until their length is the next highest power of two. If the column length is already a p ower of two, the columns are not padded.

This parameter is only visible when Columns or Columns and rows is selected for the Pad over parameter.

Column size

Specify the column length of the output. If the speci Þed column length is longer than the input column length, the columns are padded. If the speci Þed column length is shorter than the input column length, the columns are truncated. This parameter is only visible when User-specified is selected for the Output column mode parameter.

Output row mode

Choose how you specify the output row length of the output:

- If you select User-specified , the Row size parameter appears.
- If you select Next power of two , the block pads the output rows until their length is the next highest power of two. If the row length is already a power of two, the rows are not padded.

This parameter is only visible when Rows or Columns and rows is selected for the Pad over parameter.

Row size

Specify the row length of the output. If the specified row length is longer than the input row length, the rows are padded. If the specified row length is shorter than the input row length, the rows are truncated. This parameter is only visible when User-specified is selected for the Output row mode parameter.

Specify

Choose whether you want to control the output length of the specified dimensions by specifying the pad size or the output size.

This parameter is only visible when Specified dimensions is selected for the Pad over parameter.

Pad size at beginning

Specify how many values to add to the beginning of the input signal along the specified dimension(s). This parameter must be a scalar or a vector with the same number of elements as the Dimensions to pad parameter. Each element in the Pad size at beginning parameter gives the pad length for the beginning of the corresponding dimension in the Dimensions to pad parameter. Values of this parameter must be zero or a positive integer.

This parameter is only visible if Pad size is selected for the Specify parameter.

Pad size at end

Specify how many values to add to the end of the input signal along the specified dimension(s). This parameter must be a scalar or a vector with the same number of elements as the Dimensions to pad parameter. Each element in the Pad size at end parameter gives the pad length for the end of the corresponding dimension in the Dimensions to pad parameter. Values of this parameter must be zero or a positive integer.

This parameter is only visible if Pad size is selected for the Specify parameter.

Output size mode

Choose how you specify the output length of the specified dimensions:

- If you select User-specified, the Output size parameter appears.
- If you select Next power of two, the block pads the specified dimensions until their length is the next highest power of two. If the dimension length is already a power of two, no padding occurs in that dimension.

This parameter is only visible if Output size is selected for the Specify parameter.

Output size

Specify the output length of the specified dimension(s). This parameter must be a scalar or a vector with the same number of elements as the Dimensions to pad parameter. Each element in the Output size vector gives the output length for the corresponding dimension in the Dimensions to pad vector. If the specified length is longer than the input length for a given dimensions, that dimension is padded. If the specified length is shorter than the input length for a given dimension, that dimension is truncated.

This parameter is only visible if Output size is selected for the Specify parameter.

Pad signal at

Specify whether to pad or truncate the signal at the Beginning, End, or Beginning and end of the specified dimension(s). When you select Beginning and end, half the pad length is added to the beginning of the signal, and half is added to the end of the signal. For an odd pad length, the extra value is added to the end of the signal. This also applies to truncation. In this mode, an equal number of values are truncated from the beginning and the end of the signal. In the case of an odd truncation length, the extra value is removed from the end of the signal.

Action when truncation occurs

Choose None when you do not want to be notified that the input is truncated. Select Warning to display a warning when the input is truncated. Choose Error when to display an error and terminate the simulation when the input is truncated.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating-point • Single-precision floating-point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating-point • Single-precision floating-point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Concatenate	Simulink
Repeat	Signal Processing Blockset
Submatrix	Signal Processing Blockset
Upsample	Signal Processing Blockset
Variable Selector	Signal Processing Blockset

Parametric Equalizer

Purpose

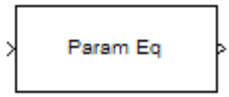
Design parametric equalizer

Library

Filtering / Filter Design Toolbox

dspfdesign

Description



This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have Filter Design Toolbox product.

Dialog Box

See “Parametric Equalizer Filter Design Dialog Box — Main Pane” in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset product.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point

Port	Supported Data Types
	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Peak Finder

Purpose

Determine whether each value of input signal is local minimum or maximum

Library

Signal Operations

dspsigops

Description

The Peak Finder block outputs the number of local extrema in the input signal at the Cnt port. Optionally, it can also output the extrema indices, the extrema values, and a binary indicator of whether or not the extrema are maxima or minima.

The Peak Finder block compares the current signal value to the previous and next values to determine if the current value is an extremum. Use the Peak type(s) parameter to specify whether you are looking for maxima, minima, or both.

If you select the Output peak indices check box, the Idx port appears on the block. The block outputs the zero-based extrema indices at the Idx port. If you select the Output peak values check box, the Val port appears on the block. The block outputs the extrema values at the Val port. If you select either of these check boxes and Maxima and Minima is selected for the Peak type(s) , the Pol port also appears on the block. If the signal value is a maximum, the block outputs a 1 at the Pol ("Polarity") port. If the signal value is a minimum, the block outputs a 0 at the Pol port.

Note that nothing is output at the Idx, Val, and Pol ports for an input signal value that is not an extremum.

Use the Maximum number of peaks to find parameter to specify how many extrema to look for in each input signal. The block stops searching the input signal once this maximum number of extrema has been found.

If you select the Ignore peaks within threshold of neighboring values check box, the block no longer detects low-amplitude peaks. This feature allows the block to ignore noise within a threshold value that you de Þne. Enter a threshold value for the Threshold parameter. Now, the current value is a maximum i f (current – previous) > threshold

and (current – next) > threshold. The current value is a minimum if (current – previous) < –threshold a nd (current – next) < –threshold.

This block supports single-channe l, multichannel, sa mple-based, and frame-based inputs. These input signals must be real-valued Þxed-point or ßoating-point scalars or vectors.

Examples

Example 1

Consider the input vector

[9 6 10 3 4 5 0 12]

The table below shows the analysi s made by the Peak Finder block. Note that the Þrst and last input signal values are not considered:

Previous, current, and next values	9 6 10	6 10 3	10 3 4	3 4 5	4 5 0	5 0 12
Current value if it is an extremum	6	10	3	—	5	0
Index of current value if it is an extremum	1	2	3	—	5	6
Polarity of current value if it is an extremum	0	1	0	—	1	0

Therefore, f or this example the outputs at the block ports are

Cnt: 5

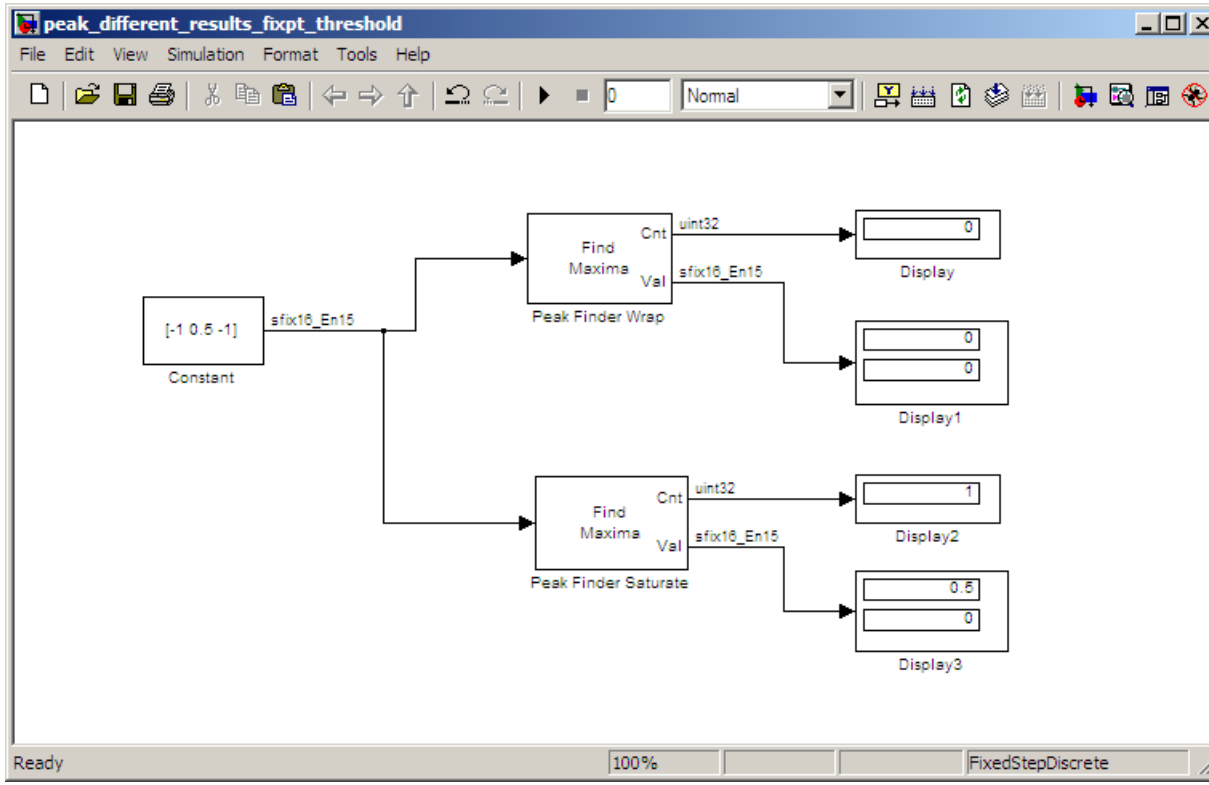
Idx: [1 2 3 5 6]

Val: [6 10 3 5 0]

Pol: [0 1 0 1 0]

Example 2

Note that the Over flow mode parameter can affect the output of the block when the input is Þxed point. Consider the following model:



In this model, the settings in the Constant block are

- Constant value — [-1 0.5 -1]
- Interpret vector parameters as 1–D — not selected
- Sampling mode — Sample based
- Sample time — 1
- Output data type — <data type expression>
- Mode — Fixed point
- Sign — Signed

- Scaling — Binary point
- Word length — 16
- Fraction length — 15

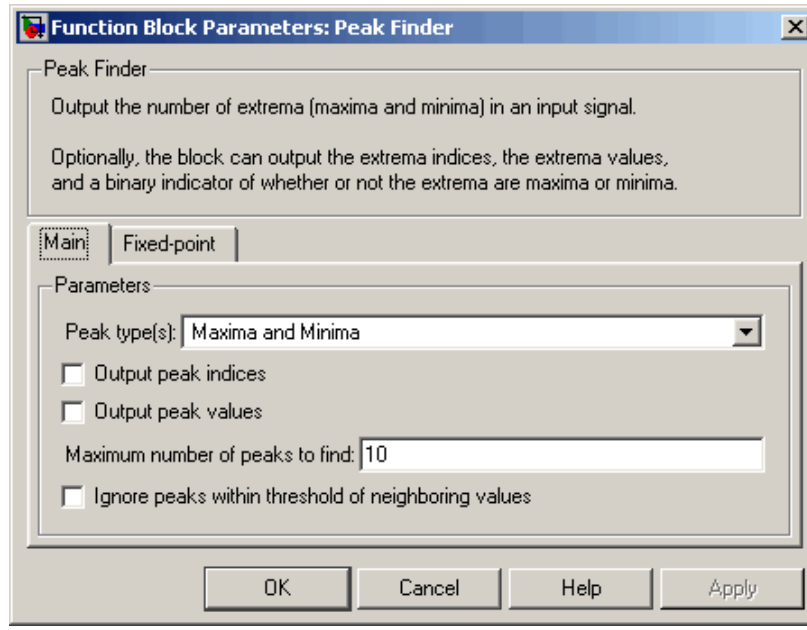
The settings in the Peak Finder blocks are

- Peak type(s) — Maxima
- Output peak indices — not selected
- Output peak values — selected
- Maximum number of peaks to find — 2
- Ignore peaks within threshold of neighboring values — selected
- Threshold — 0.25
- Rounding mode — Floor
- Overflow mode — Wrap for Peak Finder Wrap, Saturate for Peak Finder Saturate

Setting the Overflow mode parameter of the Peak Finder Wrap block to Wrap causes the calculations (current – previous) > threshold and (current – next) > threshold to wrap on overflow, thereby causing the maximum to be missed.

Dialog Box

The Main pane of the Peak Finder block dialog appears as follows.



Peak type(s)
: Specify whether you are looking for maxima, minima, or both.

Output peak indices
: Select this check box if you want the block to output the extrema indices at the Idx port.

Output peak values
: Select this check box if you want the block to output the extrema values at the Val port.

Maximum number of peaks to find
: Enter the number of extrema to look for in each input signal. The block stops searching the input signal for extrema once the maximum number of extrema has been found. The value of this parameter must be an integer greater than or equal to one.

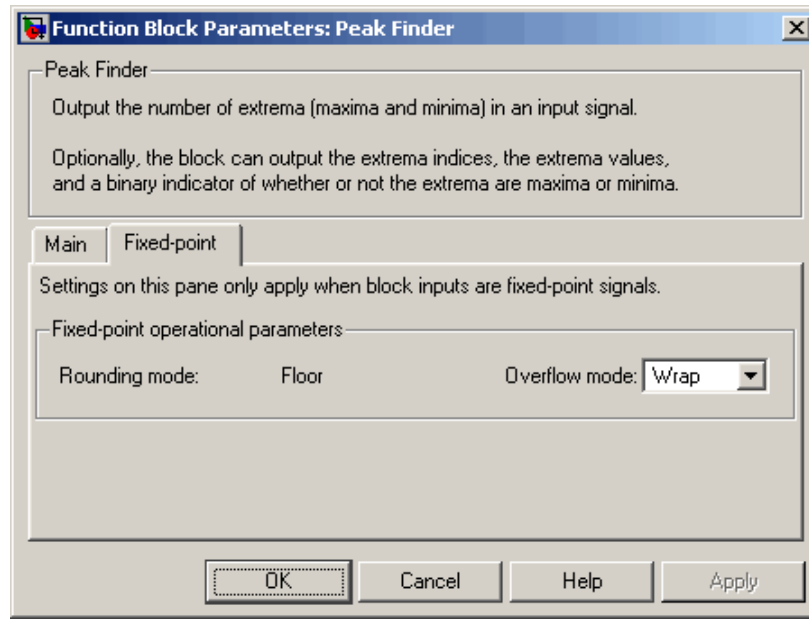
Ignore peaks within threshold of neighboring values

Select this check box if you want to eliminate the detection of peaks whose amplitudes are within a speci Þed threshold of neighboring values.

Threshold

Enter your threshold value. This parameter appears if you select the Ignore peaks within threshold of neighboring values check box.

The Fixed-point pane of the Peak Finder block dialog appears as follows.



Rounding mode

The rounding mode of this block is always Floor .

Over flow mode

Select the overflow mode to be used when block inputs are fixed point.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Cnt	• 32-bit unsigned integers
Idx	• 32-bit unsigned integers
Val	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Pol	• Boolean

See Also

Maximum	Signal Processing Blockset
Minimum	Signal Processing Blockset

Purpose

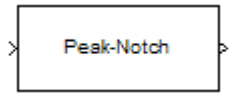
Design peak or notch filter

Library

Filtering / Filter Design Toolbox

dspfdesign

Description



This block brings the functionality of the Filter Design Toolbox filterbuilder function to the Simulink environment. You must have a Filter Design Toolbox product license to design filters with this block. However, you can run models containing this block without a license. This allows you to run a model sent to you by a colleague who has designed a filter using this block, even if you do not have the Filter Design Toolbox product.

Dialog Box

See “Peak/Notch Filter Design Dialog Box — Main Pane” in the Filter Design Toolbox documentation for more information about the parameters of this block. The Data Types pane is not available for Filter Design Toolbox blocks in the Signal Processing Blockset product.

Parameters of this block that do not change filter order or structure are tunable. Tunable parameters are enabled during simulation; nontunable parameters are not.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point

Port	Supported Data Types
	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Purpose

Compute nonparametric estimate of spectrum

Library

Estimation / Power Spectrum Estimation

dspspect3

Description

The Periodogram block computes a nonparametric estimate of the spectrum. The block averages the squared magnitude of the FFT computed over windowed sections of the input and normalizes the spectral average by the square of the sum of the window samples.

Both an M-by-N frame-based matrix input and an M-by-N sample-based matrix input are treated as M sequential time samples from N independent channels. The block computes a separate estimate for each of the N independent channels and generates an N_{fft}-by-N matrix output. When you select the Inherit FFT length from input dimensions check box, N_{fft} is specified by the frame size of the input, which must be a power of 2. When you clear the Inherit FFT length from input dimensions check box, N_{fft} is specified as a power of 2 by the FFT length parameter, and the block zero pads or wraps the input to N_{fft} before computing the FFT.

Each column of the output matrix contains the estimate of the corresponding input column's power spectral density at N_{fft} equally spaced frequency points in the range [0,F_s), where F_s is the signal's sample frequency. The output is always sample based.

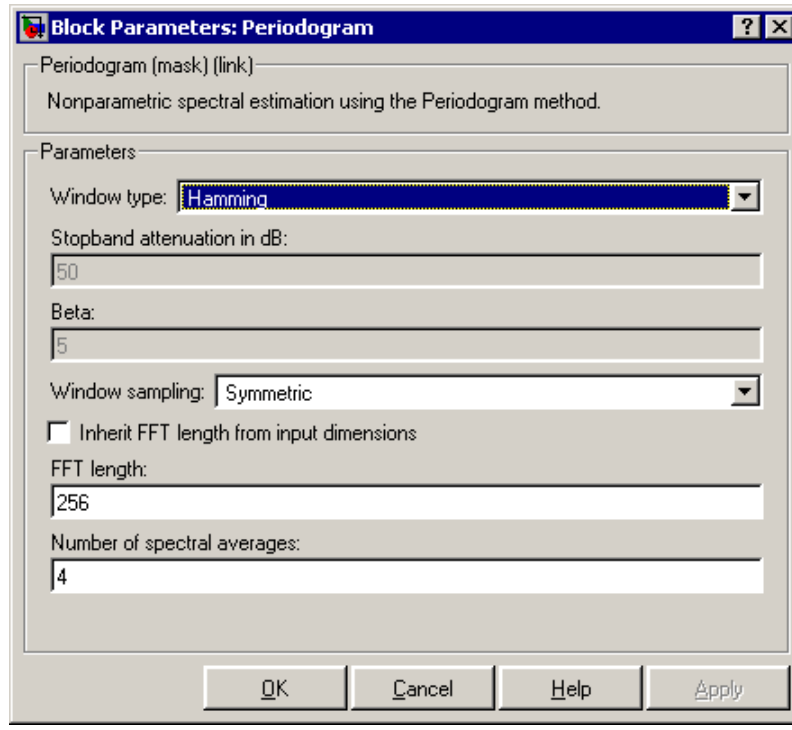
The Number of spectral averages specifies the number of spectra to average. Setting this parameter to 1 effectively disables averaging.

The Window type, Stopband ripple, Beta, and Window sampling parameters all apply to the specification of the window function; see the Window Function block reference page for more details on these four parameters.

Example

The dspstfft demo provides an illustration of using the Periodogram and Matrix Viewer blocks to create a spectrogram. The dspsacomp demo compares the Periodogram block with several other spectral estimation methods.

Dialog Box



Window type

Enter the type of window to apply. See the Window Function block reference page for more details. Tunable.

Stopband attenuation in dB

Enter the level, in dB, of stopband attenuation, R_s, for the Chebyshev window. This parameter is enabled if, for the Window type parameter, you choose Chebyshev. Tunable.

Beta

Enter the ß parameter for the Kaiser window. This parameter is enabled if, for the Window type parameter, you chose Kaiser. Increasing Beta widens the mainlobe and decreases the

amplitude of the window sidelobes in the window's frequency magnitude response. Tunable.

Window sampling

From the list, choose Symmetric or Periodic . Tunable.

Inherit FFT length from input dimensions

When you select this check box, the block uses the input frame size as the number of data points, N_{fft}, on which to perform the FFT.

FFT length

Enter the number of data points on which to perform the FFT, N_{fft}. When N_{fft} is larger than the input frame size, each frame is zero-padded as needed. When N_{fft} is smaller than the input frame size, each frame is wrapped as needed. This parameter is enabled when you clear the Inherit FFT length from input dimensions check box.

Number of spectral averages

Enter the number of spectra to average; setting this parameter to 1 disables averaging.

References

Oppenheim, A. V. and R. W. Schafer. Discrete-Time Signal Processing . Englewood Cliffs, NJ: Prentice Hall, 1989.

Orfanidis, S. J. Introduction to Signal Processing . Englewood Cliffs, NJ: Prentice-Hall, 1995.

Proakis, J. and D. Manolakis. Digital Signal Processing. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point
Output	• Double-precision floating point • Single-precision floating point

See Also

Burg Method	Signal Processing Blockset
Inverse Short-Time FFT	Signal Processing Blockset
Magnitude FFT	Signal Processing Blockset
Short-Time FFT	Signal Processing Blockset
Spectrum Scope	Signal Processing Blockset
Window Function	Signal Processing Blockset
Yule-Walker Method	Signal Processing Blockset
pwelch	Signal Processing Toolbox

See “Power Spectrum Estimatio n” for related information.

Purpose

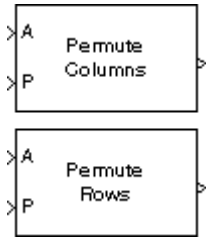
Reorder matrix rows or columns

Library

Math Functions / Matrices and Linear Algebra / Matrix Operations

dspmtrx3

Description



The Permute Matrix block reorders the rows or columns of M-by-N input matrix A as specified by indexing input P.

When the Permute parameter is set to Rows, the block uses the rows of A to create a new matrix with the same column dimension. Input P is a length-L vector whose elements determine where each row from A should be placed in the L-by-N output matrix.

```
% Equivalent MATLAB code
y = [A(P(1),:) ; A(P(2),:) ; A(P(3),:) ; ... ; A(P(end),:)]
```

For row permutation, a length-M 1-D vector input at the A port is treated as a M-by-1 matrix.

When the Permute parameter is set to Columns, the block uses the columns of A to create a new matrix with the same row dimension. Input P is a length-L vector whose elements determine where each column from A should be placed in the M-by-L output matrix.

```
% Equivalent MATLAB code
y = [A(:,P(1)) A(:,P(2)) A(:,P(3)) ... A(:,P(end))]
```

For column permutation, a length-N 1-D vector input at the A port is treated as a 1-by-N matrix.

When an index value in input P references a nonexistent row or column of matrix A, the block reacts with the behavior specified by the Invalid permutation index parameter. The following options are available:

- Clip index — Clip the index to the nearest valid value (1 or M for row permutation, and 1 or N for column permutation), and do not issue an alert. Example: For a 3-by-7 input matrix, a column index of 9 is clipped to 7, and a row index of -2 is clipped to 1.

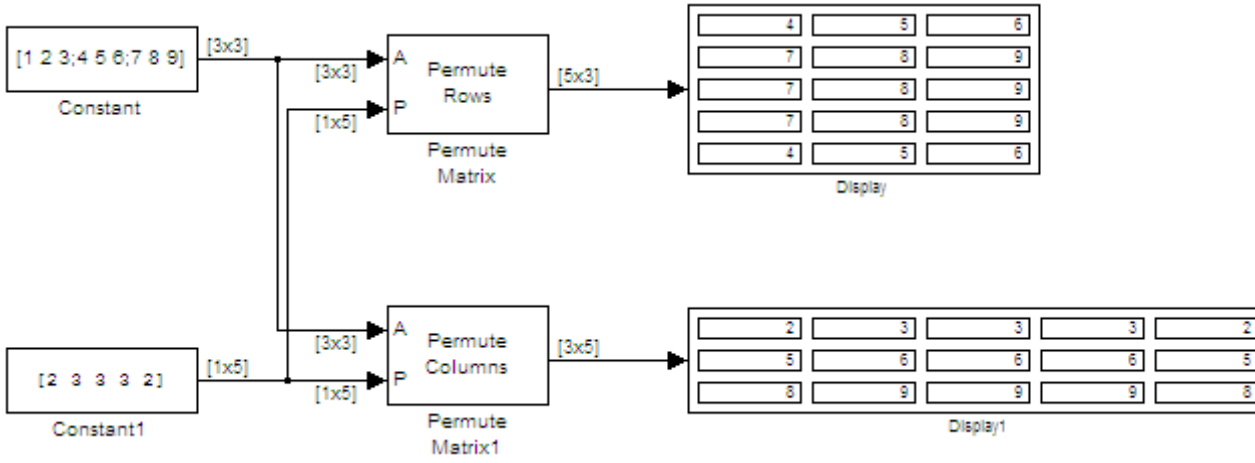
- Clip and warn — Display a warning message in the MATLAB command window, and clip the index as described above.
- Generate error — Display an error dialog box and terminate the simulation.

When length of the permutation vector P is not equal to the number of rows or columns of the input matrix A, you can choose to get an error dialog box and terminate the simulation by selecting Error when length of P is not equal to Permute dimension size .

When input A is frame based, the output is frame based; otherwise, the output is sample based.

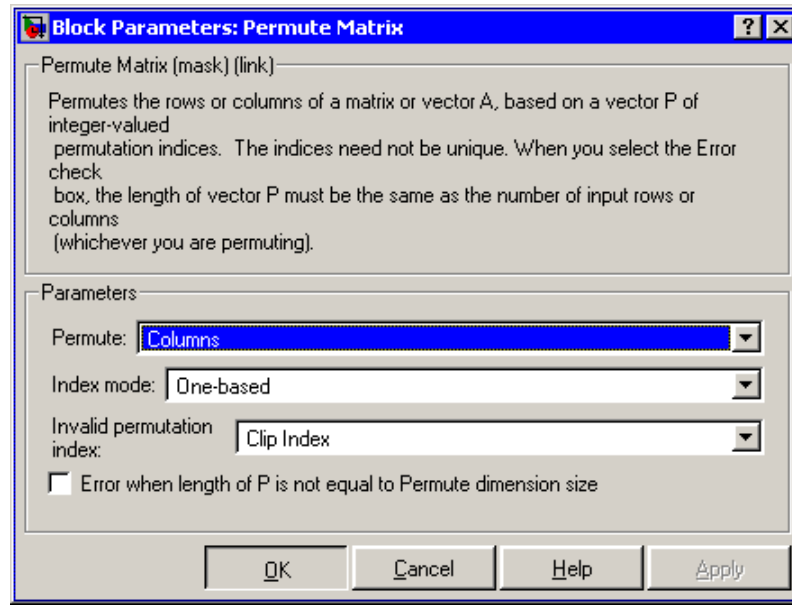
Examples

In the model below, the top Permute Matrix block places the second row of the input matrix in both the Þrst and Þfth rows of the output matrix, and places the third row of the input matrix in the three middle rows of the output matrix. The bottom Permu te Matrix block places the second column of the input matrix in both the Þrst and Þfth columns of the output matrix, and places the third column of the input matrix in the three middle columns of the output matrix.



As shown in the example above, rows and columns of A can appear any number of times in the output, or not at all.

Dialog Box



Permute

Method of constructing the output matrix; by permuting rows or columns of the input.

Index mode

When set to One-based , a value of 1 in the permutation vector P refers to the Þrst row or column of the input matrix A. When set to Zero-based , a value of 0 in P refers to the Þrst row or column of A.

Invalid permutation index

Response to an invalid index value. Tunable.

Error when length of P is not equal to Permute dimension size
Option to display an error dialog box and terminate the simulation when the length of the permutation vector P is not equal to the number of rows or columns of the input matrix A.

Supported Data Types

Port	Supported Data Types
A	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
P	• Double-precision ßoating point • Single-precision ßoating point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Submatrix Signal Processing Blockset

Transpose Signal Processing Blockset

Variable Selector	Signal Processing Blockset
permute	MATLAB

See “Reordering Channels in Multichannel Frame-Based Signals” for related information.

Polynomial Evaluation

Purpose

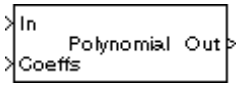
Evaluate polynomial expression

Library

Math Functions / Polynomial Functions

dsppolyfun

Description



The Polynomial Evaluation block applies a polynomial function to the real or complex input at the In port.

```
y = polyval(u)          % Equivalent MATLAB code
```

The Polynomial Evaluation block performs these types of operation more efficiently than the equivalent construction using Simulink Sum and Math Function blocks.

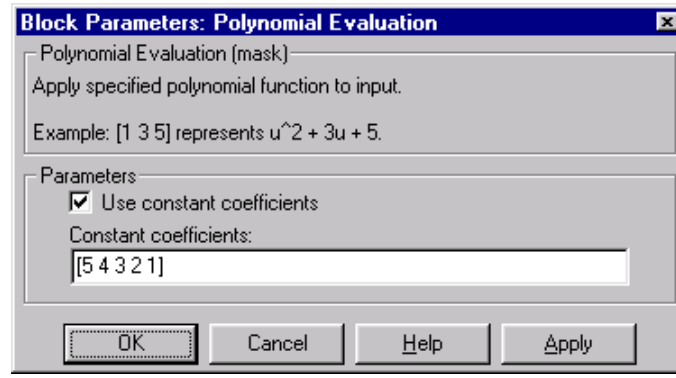
When you select the Use constant coefficients check box, you specify the polynomial expression in the Constant coefficients parameter. When you do not select Use constant coefficients, a variable polynomial expression is specified by the input to the Coeffs port. In both cases, the polynomial is specified as a vector of real or complex coefficients in order of descending exponents.

The table below shows some examples of the block's operation for various coefficient vectors.

Coefficient Vector	Equivalent Polynomial Expression
[1 2 3 4 5]	$y = u^4 + 2u^3 + 3u^2 + 4u + 5$
[1 0 3 0 5]	$y = u^4 + 3u^2 + 5$
[1 2+i 3 4-3i 5i]	$y = u^4 + (2+i)u^3 + 3u^2 + (4-3i)u + 5i$

Each element of a vector or matrix input to the In port is processed independently, and the output size and frame status are the same as the input.

Dialog Box



Use constant coefficients

Select to enable the Constant coefficients parameter and disable the Coeffs input port.

Constant coefficients

Specify the vector of polynomial coefficients to apply to the input, in order of descending exponents. This parameter is enabled when you select the Use constant coefficients check box.

Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

Least Squares Polynomial Fit	Signal Processing Blockset
Math Function	Simulink
Sum	Simulink
polyval	MATLAB

Polynomial Stability Test

Purpose

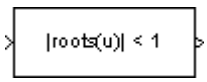
Use Schur-Cohn algorithm to determine whether all roots of input polynomial are inside unit circle

Library

Math Functions / Polynomial Functions

dsppolyfun

Description



The Polynomial Stability Test block uses the Schur-Cohn algorithm to determine whether all roots of a polynomial are within the unit circle.

```
y = all(abs(roots(u)) < 1)          % Equivalent MATLAB code
```

Each column of the M-by-N input matrix u contains M coefficients from a distinct polynomial,

$$f(x) = u_1 x^{M-1} + u_2 x^{M-2} + \ldots + u_M$$

arranged in order of descending exponents, u_1, u_2, ..., u_M. The polynomial has order M-1 and positive integer exponents.

Inputs can be frame based or sample based, and both represent the polynomial coefficients as shown above. For convenience, a length-M 1-D vector input is treated as an M-by-1 matrix.

The output is a 1-by-N matrix with each column containing the value 1 or 0. The value 1 indicates that the polynomial in the corresponding column of the input is stable; that is, the magnitudes of all solutions to f(x) = 0 are less than 1. The value 0 indicates that the polynomial in the corresponding column of the input might be unstable; that is, the magnitude of at least one solution to f(x) = 0 is greater than or equal to 1.

The output is always sample based.

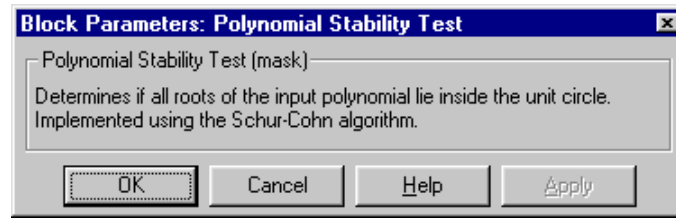
Applications

This block is most commonly used to check the pole locations of the denominator polynomial, A(z), of a transfer function, H(z).

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_1 + b_2 z^{-1} + \ldots + b_m z^{-(m-1)}}{a_1 + a_2 z^{-1} + \ldots + a_n z^{-(n-1)}}$$

The poles are the n-1 roots of the denominator polynomial, A(z). When any poles are located outside the unit circle, the transfer function H(z) is unstable. As is typical in DSP applications, the transfer function above is specified in descending powers of z^{-1} rather than z.

Dialog Box



Supported Data Types

- Double-precision floating point
- Single-precision floating point
- Boolean — Block outputs are always Boolean.

See Also

Least Squares Polynomial Fit

Polynomial Evaluation

polyfit

Signal Processing Blockset

Signal Processing Blockset

MATLAB

Pseudoinverse

Purpose

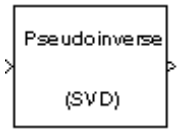
Compute Moore-Penrose pseudoinverse of matrix

Library

Math Functions / Matrices and Lin ear Algebra / Matrix Inverses

dspinverses

Description



The Pseudoinverse block computes t he Moore-Penrose pseudoinverse of input matrix A.

```
[U,S,V] = svd(A,0)          % Equivalent MATLAB code
```

The pseudoinverse of A is the matrix $A^{\dagger}$ such that

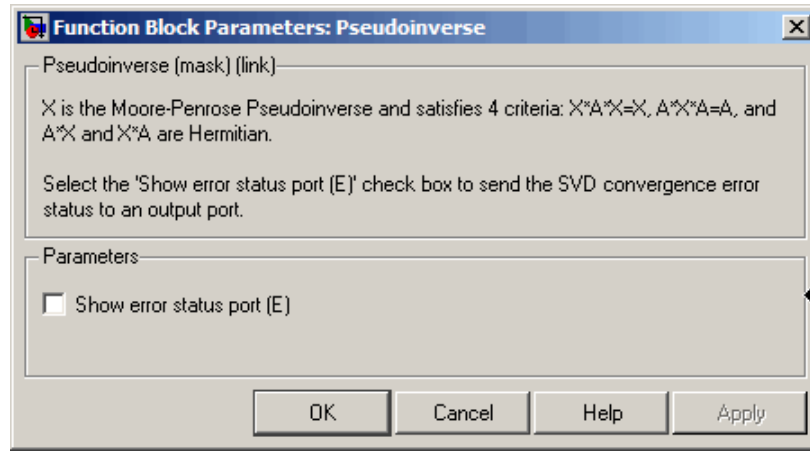
$$A^{\dagger} = VS^{\dagger}U^{*}$$

where U and V are orthogonal matrices, and S is a diagonal matrix. The pseudoinverse has the following properties:

- $AA^{\dagger} = (AA^{\dagger})^{*}$
- $A^{\dagger}A = (A^{\dagger}A)^{*}$
- $AA^{\dagger}A = A$
- $A^{\dagger}AA^{\dagger} = A^{\dagger}$

The output is always sample based.

Dialog Box



Show error status port

Select to enable the E output port, which reports a failure to converge. The possible values you can receive on the port are:

- 0 — The pseudoinverse calculation converges.
- 1 — The pseudoinverse calculation does not converge.

If the pseudoinverse calculation fails to converge, the output at port X is an unde Þned matrix of the correct size.

References

Golub, G. H., and C. F. Van Loan. Matrix Computations . 3rd ed. Baltimore, MD: Johns Hopkin s University Press, 1996.

Supporte d Data Types

Port	Supported Data Types
A	• Double-precision ßoating point • Single-precision ßoating point

Port	Supported Data Types
X	• Double-precision ßoating point • Single-precision ßoating point
E	• Boolean

See Also

Cholesky Inverse	Signa l Processing Blockset
LDL Inverse	Signal Processing Blockset
LU Inverse	Signal Processing Blockset
Singular Value Decomposition	Signal Processing Blockset
inv	MATLAB

See “Matrix Inverses” for related information.

Purpose

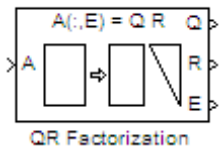
Factor arbitrary matrix into unitary and upper triangular components

Library

Math Functions / Matrices and Linea r Algebra / Matrix Factorizations

dspfactors

Description



The QR Factorization block uses a sequence of Householder transformations to triangularize the input matrix A. The block factors a column permutation of the M-by-N input matrix A as

$$A_e = QR$$

The column-pivoted matrix A_e contains the columns of A permuted as indicated by the contents of length- N permutation vector E.

```
Ae = A(:,E)        % Equivalent MATLAB code
```

The block selects a colu mn permutation vector E, which ensures that the diagonal elements of matrix R are arranged in order of decreasing magnitude.

$$\left|r_{i+1,j+1}\right| < \left|r_{i,j}\right| \qquad i = j$$

The size of matrices Q and R depends on the setting of the Output size parameter:

- When you select Economy for the output size, Q is an M-by-min(M,N) unitary matrix, and R is a min(M,N)-by-N upper-triangular matrix.

  ```
  [Q R E] = qr(A,0)    % Equivalent MATLAB code
  ```

- When you select Full for the output size, Q is an M-by-M unitary matrix, and R is a M-by-N upper-triangular matrix.

  ```
  [Q R E] = qr(A)        % Equivalent MATLAB code
  ```

A length- M 1–D vector input is always treated as an M-by-1 matrix.

QR factorization is an important tool for solving linear systems of equations because of good error propagation properties and the invertability of unitary matrices:

$$Q^{-1} = Q'$$

where Q' is the complex conjugate transpose of Q.

Unlike LU and Cholesky factorizations, the matrix A does not need to be square for QR factorization. How ever, QR factorization requires twice as many operations as LU Factorization (Gaussian elimination).

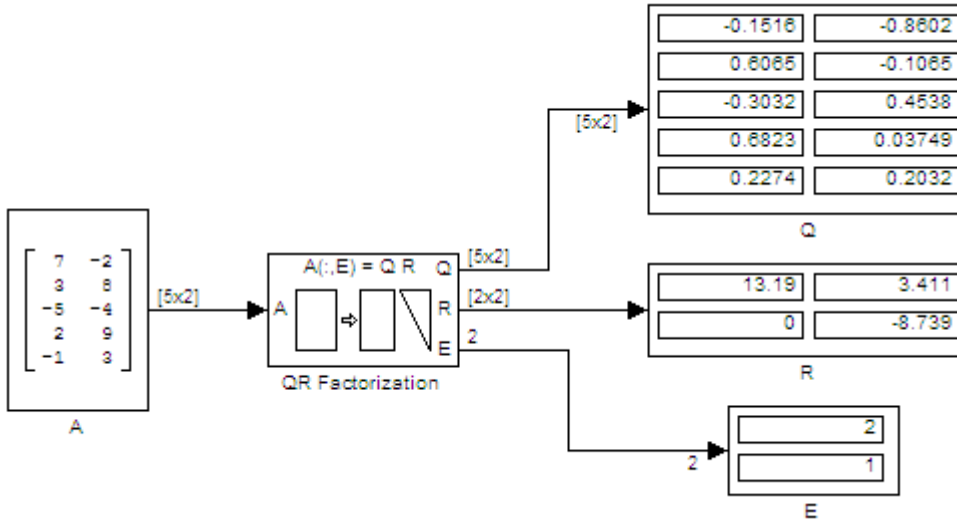
Examples

The Output size parameter of the QR factorization block has two settings: Economy and Full . When the M-by-N input matrix A has dimensions such that M > N, the dimensions of output matrices Q and R differ depending on the setting of the Output size parameter. If, however, the size of the input matrix A is such that $M \leq N$, output matrices Q and R have the same dimensions, regardless of whether the Output size is set to Economy or Full .

The input to the QR Factorization block in the following model is a 5-by-2 matrix A. When you change the setting of the Output size parameter from Economy to Full , the dimensions of the output given by the QR Factorization block also change.

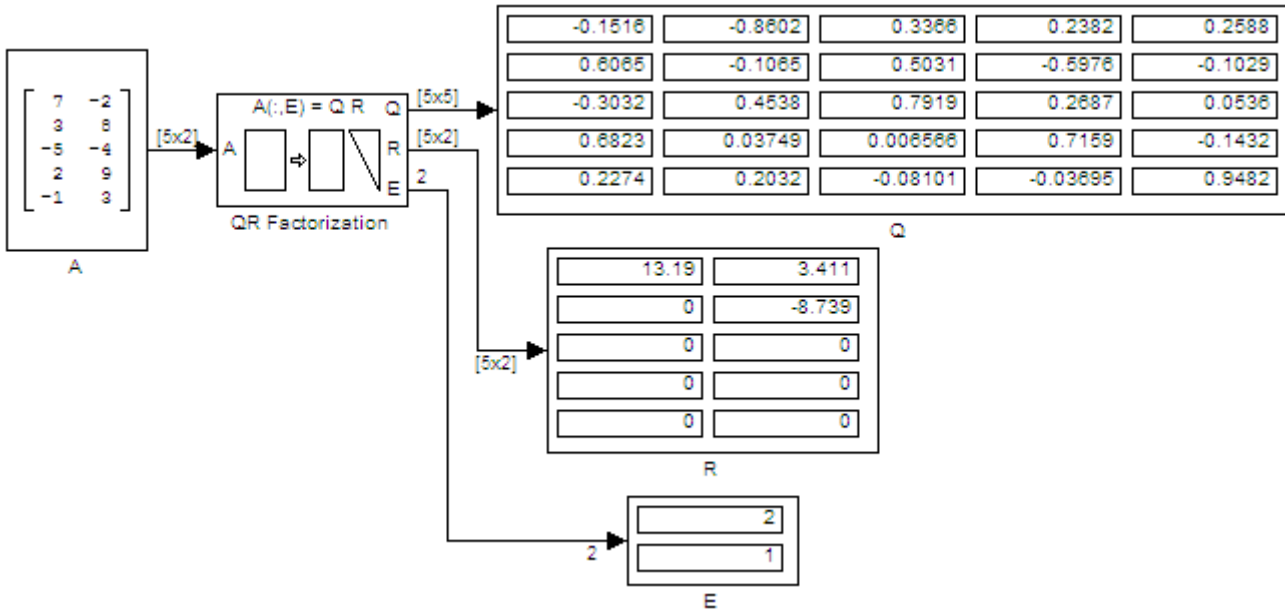
1 Open the model by typing doc_qrfactorization_ref at the MATLAB command line.

2 Double-click the QR Factorization block, set the Output size parameter to Economy, and run the model.



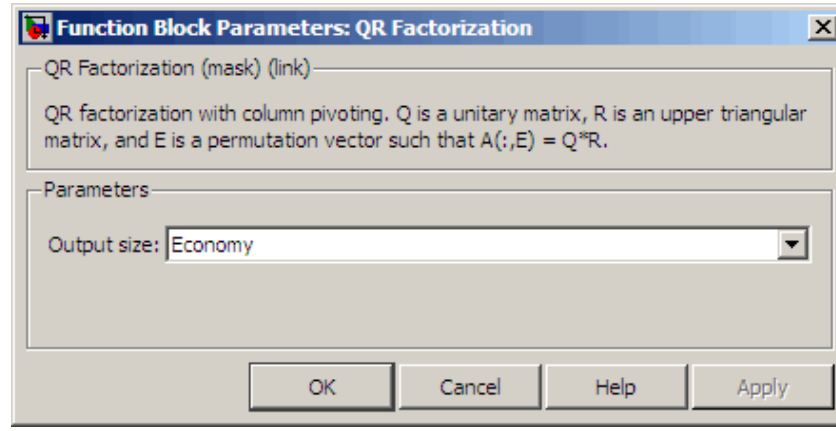
The QR Factorization block outputs a 5-by-2 matrix Q and a 2-by-2 matrix R.

3 Change the Output size parameter of the QR Factorization block to Full and rerun the model.



The QR Factorization block outputs a 5-by-5 matrix Q and a 5-by-2 matrix R.

Dialog Box



Output size

Specify the size of output matrices Q and R:

- Economy — When this output size is selected, the block outputs an M-by-min(M,N) unitary matrix Q and a min(M,N)-by-N upper-triangular matrix R.
- Full — When this output size is selected, the block outputs an M-by-M unitary matrix Q and a M-by-N upper-triangular matrix R.

References

Golub, G. H., and C. F. Van Loan. Matrix Computations . 3rd ed. Baltimore, MD: Johns Hopkin s University Press, 1996.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point
Output	• Double-precision ßoating point • Single-precision ßoating point

See Also

Cholesky Factorization	Signal Processing Blockset
LU Factorization	Signal Processing Blockset
QR Solver	Signal Processing Blockset
Singular Value Decomposition	Signal Processing Blockset
qr	MATLAB

See “Matrix Factorizations” for related information.

QR Solver

Purpose

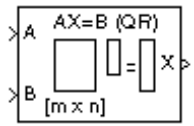
Find minimum-norm-residual solution to A X=B

Library

Math Functions / Matrices and Linea r Algebra / Linear System Solvers

dspsolvers

Description



The QR Solver block solves the linear system A X=B, which can be overdetermined, underdetermined , or exactly determined. The system is solved by applying QR factorizat ion to the M-by-N matrix, A, at the A port. The input to the B port is the right side M-by-L matrix, B. A length-M 1-D vector input at either port is treated as an M-by-1 matrix.

The output at the x port is the N-by-L matrix, X. X is always sample based, and is chosen to minimize the sum of the squares of the elements of B-AX. When B is a vector, this solutio n minimizes the vector 2-norm of the residual (B-AX is the residual). When B is a matrix, this solution minimizes the matrix Frobenius nor m of the residual. In this case, the columns of X are the solutions to the L corresponding systems $AX_k=B_k$, where B_k is the kth column of B, and X_k is the kth column of X.

X is known as the minimum-norm-residual solution to AX=B. The minimum-norm-residual solutio n is unique for overdetermined and exactly determined linear systems, but it is not unique for underdetermined linear systems. Thus when the QR Solver is applied to an underdetermined system, the output X is chosen such that the number of nonzero entries in X is minimized.

Algorithm

QR factorization factors a c olumn-permuted variant (A_e) of the M-by-N input matrix A as

$$A_e = QR$$

where Q is a M-by-min(M,N) unitary matrix, and R is a min(M,N)-by-N upper-triangular matrix.

The factored matrix is substituted for A_e in

$$A_eX = B_e$$

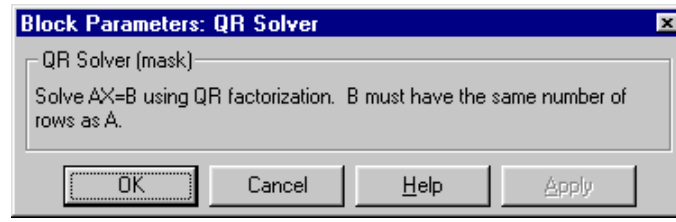
and

$$QRX = B_e$$

is solved for X by noting that $Q^{-1} = Q^*$ and substituting $Y = Q^*B_e$. This requires computing a matrix multiplication for Y and solving a triangular system for X.

$$RX = Y$$

Dialog Box



Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

Levinson-Durbin	Signal Processing Blockset
LDL Solver	Signal Processing Blockset
LU Solver	Signal Processing Blockset
QR Factorization	Signal Processing Blockset
SVD Solver	Signal Processing Blockset

See “Linear System Solvers” for related information.

Quantizer

Purpose

Discretize input at speci Þed interval

Library

Quantizers

dspquant2

Description

The Quantizer block is an implementation of the Simulink Quantizer block. See Quantizer for more information.

Purpose

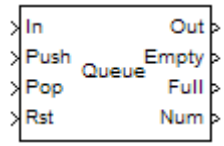
Store inputs in FIFO register

Library

Signal Management / Buffers

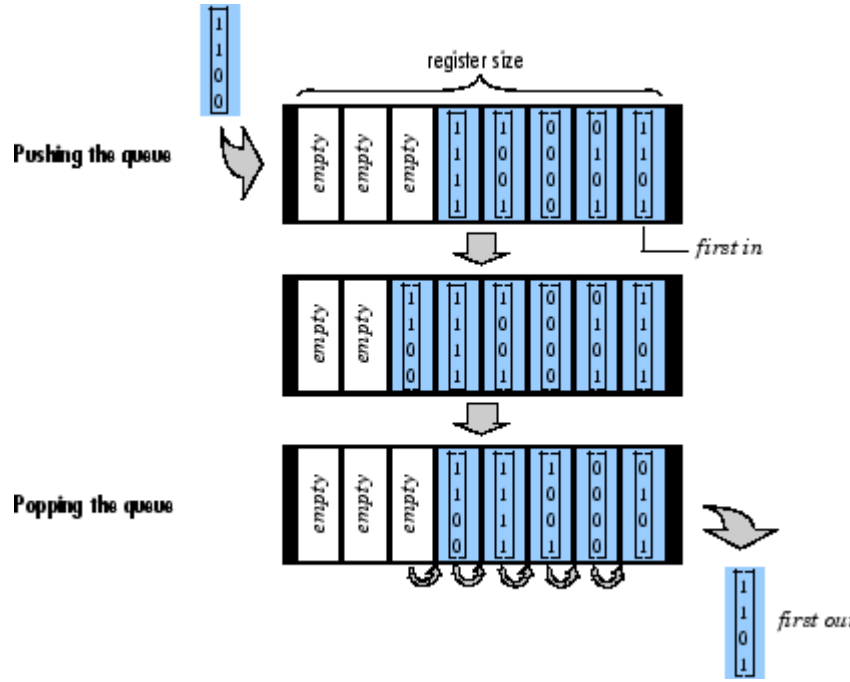
dspbuff3

Description



The Queue block stores a sequence of input samples in a Þrst in, Þrst out (FIFO) register. The register capacity is set by the Register size parameter, and inputs can be scalars, vectors, or matrices.

The block pushes the input at the In port onto the end of the queue when a trigger event is received at the Push port. When a trigger event is received at the Pop port, the block pops the Þrst element off the queue and holds the Out port at that value. The Þrst input to be pushed onto the queue is always the Þrst to be popped off.



A trigger event at the optional Rst port empties the queue contents. When you select Clear output port on reset, then a trigger event at the Rst port empties the queue and sets the value at the Out port to zero. This setting also applies when a disabled subsystem containing the Queue block is reenabled; the Out port value is only reset to zero in this case when you select Clear output port on reset.

When two or more of the control input ports are triggered at the same time step, the operations are executed in the following order:

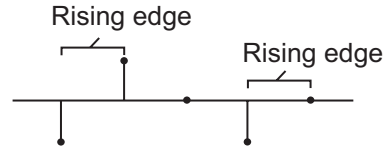
1 Rst

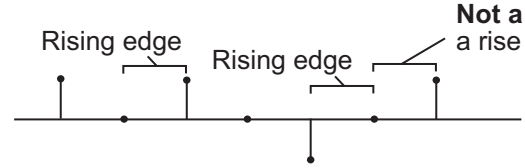
2 Push

3 Pop

The rate of the trigger signal must be the same as the rate of the data signal input. You specify the triggering event for the Push, Pop, and Rst ports by the Trigger type pop-up menu:

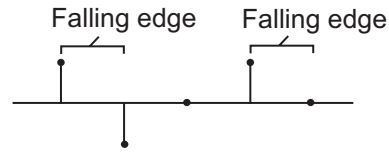
- Rising edge — Triggers execution of the block when the trigger input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero; see the following figure

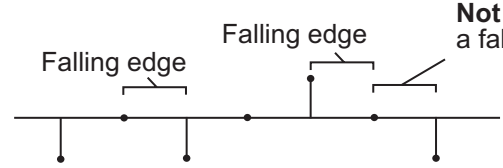




Not a rising edge because it continues a rise from a negative value to zero.

- Falling edge — Triggers execution of the block when the trigger input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positi ve value to zero; see the following Þgure





Not a falling edge because it continues a fall from a positive value to zero.

- Either edge — Triggers execution of the block when the trigger input is a Rising edge or Falling edge (as described above).
- Non-zero sample — Triggers execution of the block at each sample time that the trigger input is not zero.

Note When running simulations in the Simulink MultiTasking mode, sample-based trigger signals have a one-sample latency, and frame-based trigger signals have one frame of latency. Thus, there is a one-sample or one-frame delay between the time the block detects a trigger event, and when it applies the trigger. For more information on latency and the Simulink tasking modes, see "Excess Algorithmic Delay (Tasking Latency)" and "Models with Multiple Sample Rates" in the Real-Time Workshop User's Guide .

Note If your model contains any referenced models that use the Queue block, you cannot simulate the top-level model in Simulink Accelerator mode.

The Push onto full register parameter specifies the block's behavior when a trigger is received at the Push port but the register is full. The Pop empty register parameter specifies the block's behavior when a trigger is received at the Pop port but the register is empty. The following options are available for both cases:

- Ignore — Ignore the trigger event, and continue the simulation.
- Warning — Ignore the trigger event, but display a warning message in the MATLAB Command Window.
- Error — Display an error dialog box and terminate the simulation.

Note The Push onto full register and Pop empty register parameters are diagnostic parameters. Like all diagnostic parameters on the Configuration Parameters dialog box, they are set to Ignore in the code generated for this block by Real-Time Workshop code generation software.

The Push onto full register parameter additionally offers the Dynamic reallocation option, which dynamically resizes the register to accept as many additional inputs as memory permits. To find out how many elements are on the queue at a given time, enable the Num output port by selecting the Show number of register entries port parameter.

Note When Dynamic reallocation is selected, the System target file parameter on the Real-Time Workshop pane of the Configuration Parameters dialog box must be set to grt_malloc.tlc Generic Real-Time Target with dynamic memory allocation .

Examples

Example 1

The table below illustrates the Queue block's operation for a Register size of 4, Trigger type of Either edge , and Clear output port on reset enabled. Because the block triggers on both rising and falling edges in this example, each transition from 1 to 0 or 0 to 1 in the Push, Pop, and Rst columns below represents a distinct trigger event. A 1 in the Empty column indicates an empty queue, while a 1 in the Full column indicates a full queue.

In	Push	Pop	Rst	Queue	Out	Empty	Full	Num
1	0	0	0	top bottom	0	1	0	0
2	1	0	0	top 2 bottom	0	0	0	1
3	0	0	0	top 3 2 bottom	0	0	0	2
4	1	0	0	top 4 3 2 bottom	0	0	0	3
5	0	0	0	top 5 4 3 2 bottom	0	0	1	4

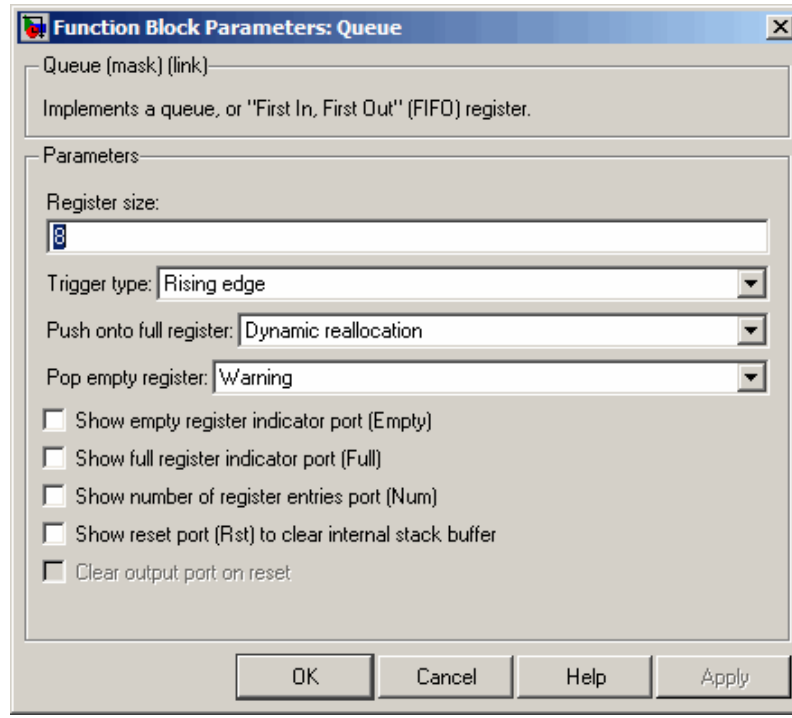
In	Push	Pop	Rst	Queue	Out	Empty	Full	Num
6	0	1	0	top 5 4 3 bottom	2	0	0	3
7	0	0	0	top 5 4 bottom	3	0	0	2
8	0	1	0	top 5 bottom	4	0	0	1
9	0	0	0	top bottom	5	1	0	0
10	1	0	0	top 10 bottom	5	0	0	1
11	0	0	0	top 11 10 bottom	5	0	0	2
12	1	0	1	top 12 bottom	0	0	0	1

Note that at the last step shown, the Push and Rst ports are triggered simultaneously. The Rst trigger takes precedence, and the queue is Þrst cleared and then pushed.

Example 2

The dspqdemo demo provides another example of the operation of the Queue block.

Dialog Box



Register size

The number of entries that the FIFO register can hold.

Trigger type

The type of event that triggers the block's execution. The rate of the trigger signal must be the same as the rate of the data signal input.

Push onto full register

Response to a trigger received at the Push port when the register is full. Inputs to this port must have the same built-in data type as inputs to the Pop and Rst input ports.

When Dynamic reallocation is selected, the System target file parameter on the Real-Time Workshop pane of the Configuration Parameters dialog box must be set to grt_malloc.tlc Generic Real-Time Target with dynamic memory allocation .

Pop empty register
: Response to a trigger received at the Pop port when the register is empty. Inputs to this port must have the same built-in data type as inputs to the Push and Rst input ports.

Show empty register indicator port
: Enable the Empty output port, which is high (1) when the queue is empty, and low (0) otherwise.

Show full register indicator port
: Enable the Full output port, which is high (1) when the queue is full, and low (0) otherwise. The Full port remains low when you select Dynamic reallocation from the Push onto full register parameter.

Show number of register entries port
: Enable the Num output port, which tracks the number of entries currently on the queue. When inputs to the In port are double-precision values, the outputs from the Num port are double-precision values. Otherwise, the outputs from the Num port are 32-bit unsigned integer values.

Show reset port to clear internal stack buffer
: Enable the Rst input port, which empties the queue when the trigger specified by the Trigger type is received. Inputs to this port must have the same built-in data type as inputs to the Push and Pop input ports.

Clear output port on reset
: Reset the Out port to zero, in addition to clearing the queue, when a trigger is received at the Rst input port.

Supported Data Types

Port	Supported Data Types
In	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Push	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers Inputs to this port must have the same built-in data type as inputs to the Pop and Rst input ports
Pop	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers Inputs to this port must have the same built-in data type as inputs to the Push and Rst input ports.

Port	Supported Data Types
Rst	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers Inputs to this port must have the same built-in data type as inputs to the Push and Pop input ports.
Out	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Empty	• Double-precision ßoating point • Boolean
Full	• Double-precision ßoating point • Boolean
Num	• Double-precision ßoating point The block outputs a double-precision ßoating-point value at this port when the data type of the In port is double-precision ßoating-point. • 32-bit unsigned integers The block outputs a 32-bit unsigned integer value at this port when the data type of the In port is anything other than double-precision ßoating-point.

See Also

Buffer	Signal Processing Blockset
Delay Line	Signal Processing Blockset
Stack	Signal Processing Blockset

Random Source

Purpose

Generate randomly distributed values

Library

Signal Processing Sources

dspsrcs4

Description

The Random Source block generates a frame of M values drawn from a uniform or Gaussian pseudorandom distribution, where you specify M in the Samples per frame parameter.

This reference page contains a detailed discussion of the following Random Source block topics:

- "Distribution Type" on page 2-1056
- "Output Complexity" on page 2-1057
- "Output Repeatability" on page 2-1059
- "Specifying the Initial Seed" on page 2-1059
- "Sample Period" on page 2-1060
- "Dialog Box" on page 2-1061
- "Supported Data Types" on page 2-1064
- "See Also" on page 2-1064

Distribution Type

When the Source type parameter is set to Uniform, the output samples are drawn from a uniform distribution whose minimum and maximum values are specified by the Minimum and Maximum parameters, respectively. All values in this range are equally likely to be selected. A length-N vector specified for one or both of these parameters generates an N-channel output (M-by-N matrix) containing a unique random distribution in each channel.

For example, specify

- Minimum = [0 0 -3 -3]

- Maximum = [10 10 20 20]

to generate a four-channel output whose first and second columns contain random values in the range [0, 10], and whose third and fourth columns contain random values in the range [-3, 20]. When you specify only one of the Minimum and Maximum parameters as a vector, the block scalar expands the other parameter so it is the same length as the vector.

When the Source type parameter is set to Gaussian, you must also set the Method parameter, which determines the method by which the block computes the output, and has the following settings:

- Ziggurat — Produces Gaussian random values by using the Ziggurat method, which is the same method used by the MATLAB randn function.
- Sum of uniform values — Produces Gaussian random values by adding and scaling uniformly distributed random signals based on the central limit theorem. This theorem states that the probability distribution of the sum of a sufficiently high number of random variables approaches the Gaussian distribution. You must set the Number of uniform values to sum parameter, which determines the number of uniformly distributed random numbers to sum to produce a single Gaussian random value.

For both settings of the Method parameter, the output samples are drawn from the normal distribution defined by the Mean and Variance parameters. A length-N vector specified for one or both of the Mean and Variance parameters generates an N-channel output (M-by-N frame matrix) containing a distinct random distribution in each column. When you specify only one of these parameters as a vector, the block scalar expands the other parameter so it is the same length as the vector.

Output Complexity

The block's output can be either real or complex, as determined by the Real and Complex options in the Complexity parameter. These settings control all channels of the output, so real and complex data

cannot be combined in the same output. For complex output with a Uniform distribution, the real and imaginary components in each channel are both drawn from the same uniform random distribution, deÞned by the Minimum and Maximum parameters for that channel.

For complex output with a Gaussian distribution, the real and imaginary components in each channel are drawn from normal distributions with different means. In this case, the Mean parameter for each channel should specify a complex value; the real component of the Mean parameter speciÞes the mean of the real components in the channel, while the imaginary component speciÞes the mean of the imaginary components in the channel. When either the real or imaginary component is omitted from the Mean parameter, a default value of 0 is used for the mean of that component.

For example, a Mean parameter setting of [5+2i 0.5 3i] generates a three-channel output with the following means.

Channel 1 mean	real = 5	imaginary = 2
Channel 2 mean	real = 0.5	imaginary = 0
Channel 3 mean	real = 0	imaginary = 3

For complex output, the Variance parameter, σ^2, speciÞes the total variance for each output channel. This is the sum of the variances of the real and imaginary components in that channel.

$$\sigma^2 = \sigma_{\mathrm{Re}}^2 + \sigma_{\mathrm{Im}}^2$$

The speciÞed variance is equally divided between the real and imaginary components, so that

$$\sigma_{\mathrm{Re}}^2 = \frac{\sigma^2}{2}$$

$$\sigma_{\mathrm{Im}}^2 = \frac{\sigma^2}{2}$$

Output Repeatability

The Repeatability parameter determines whether or not the block outputs the same signal each time you run the simulation. You can set the parameter to one of the following options:

- Repeatable — Outputs the same signal each time you run the simulation. The first time you run the simulation, the block randomly selects an initial seed. The block reuses these same initial seeds every time you rerun the simulation.
- Specify seed — Outputs the same signal each time you run the simulation. Every time you run the simulation, the block uses the initial seed(s) specified in the Initial seed parameter. Also see “Specifying the Initial Seed” on page 2-1059.
- Not repeatable — Does not output the same signal each time you run the simulation. Every time you run the simulation, the block randomly selects an initial seed.

Specifying the Initial Seed

When you set the Repeatability parameter to Specify seed, you must set the Initial seed parameter. The Initial seed parameter specifies the initial seed for the pseudorandom number generator. The generator produces an identical sequence of pseudorandom numbers each time it is executed with a particular initial seed.

Specifying Initial Seeds for Real Outputs

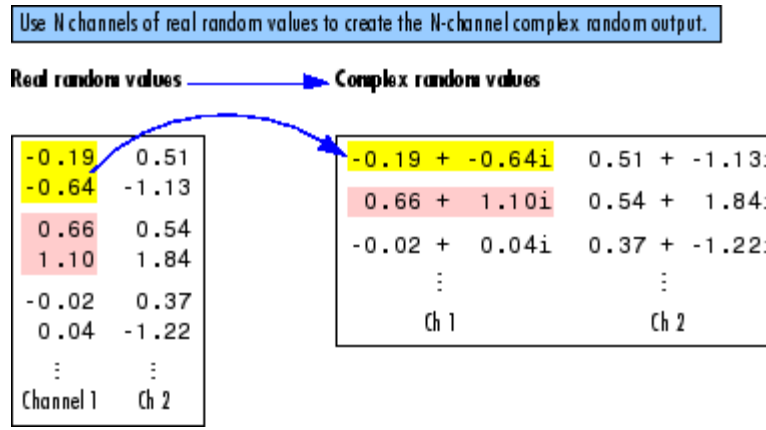
To specify the N initial seeds for an N-channel real-valued output, Complexity parameter set to Real, provide one of the following in the Initial seed parameter:

- Length-N vector of initial seeds — Uses each vector element as an initial seed for the corresponding channel in the N-channel output.
- Single scalar — Uses the scalar to generate N random values, which it uses as the seeds for the N-channel output.

Specifying Initial Seeds for Complex Outputs

To specify the initial seeds for an N-channel complex-valued output, Complexity parameter set to Complex , provide one of the following in the Initial seed parameter:

- Length-N vector of initial seeds — Uses each vector element as an initial seed for generating N channels of real random values. The block uses pairs of adjacent values in each of these channels as the real and imaginary components of the Þnal output, as illustrated in the following Þgure.
- Single scalar — Uses the scalar to generate N random values, which it uses as the seeds for generating N channels of real random values. The block uses pairs of adjacent values in each of these channels as the real and imaginary components of the Þnal output, as illustrated in the following Þgure.



Sample Period

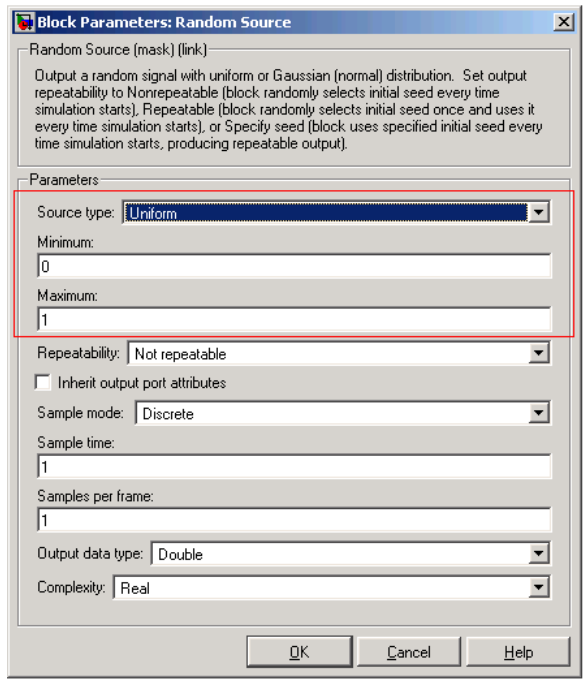
The Sample time parameter value, T_s, speciÞes the random sequence sample period when the Sample mode parameter is set to Discrete . In this mode, the block generates the number of samples speci Þed by the Samples per frame parameter value, M, and outputs this frame

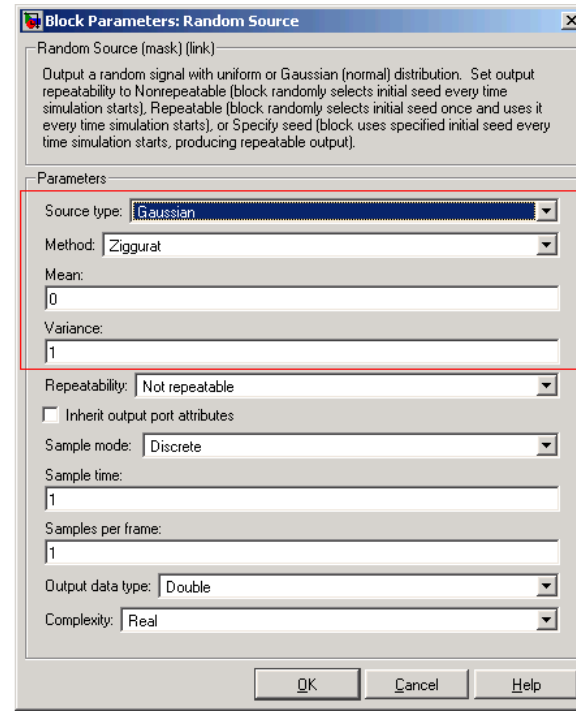
with a period of M*T_s. For M=1, the output is sample based; otherwise, the output is frame based.

When Sample mode is set to Continuous, the block is configured for continuous-time operation, and the Sample time and Samples per frame parameters are disabled. Note that many Signal Processing Blockset blocks do not accept continuous-time inputs.

Dialog Box

Only some of the parameters described below are visible in the dialog box at any one time.





Opening this dialog box causes a running simulation to pause. See "Changing Source Block Parameters During Simulation" in the online Simulink documentation for details.

Source type

The distribution from which to draw the random values, Uniform or Gaussian . For more information, see “Distribution Type” on page 2-1056.

Method

The method by which the block computes the Gaussian random values, Ziggurat or Sum of uniform values . This parameter is enabled when Source type is set to Gaussian . For more information, see “Distribution Type” on page 2-1056.

Minimum

The minimum value in the uniform distribution. This parameter is enabled when you select Uniform from the Source type parameter. Tunable.

Maximum

The maximum value in the uniform distribution. This parameter is enabled when you select you select Uniform from the Source type parameter. Tunable.

Number of uniform values to sum

The number of uniformly distributed random values to sum to compute a single number in a Gaussian random distribution. This parameter is enabled when the Source type parameter is set to Gaussian , and the Method parameter is set to Sum of uniform values . For more information, see “Distribution Type” on page 2-1056.

Mean

The mean of the Gaussian (normal) distribution. This parameter is enabled when you select Gaussian from the Source type parameter. Tunable.

Variance

The variance of the Gaussian (n ormal) distribution. This parameter is enabled when you select Gaussian from the Source type parameter. Tunable.

Repeatability

The repeatability of the block output: Not repeatable, Repeatable, or Specify seed. In the Repeatable and Specify seed settings, the block outputs the same signal every time you run the simulation. For details, see “Output Repeatability” on page 2-1059.

Initial seed

The initial seed(s) to use for the random number generator when you set the Repeatability parameter to Specify seed. For details, see “Specifying the Initial Seed” on page 2-1059. Tunable.

Inherit output port attributes

When you select this check box, block inherits the sample mode, sample time, output data type, complexity, and signal dimensions of a sample-based signal from a downstream block. When you select this check box, the Sample mode, Sample time, Samples per frame, Output data type, and Complexity parameters are disabled.

Suppose you want to back propagate a 1-D vector. The output of the Random Source block is a length M sample-based 1-D vector, where length M is inherited from the downstream block. When the Minimum, Maximum, Mean, or Variance parameter specifies N channels, the 1-D vector output contains M/N samples from each channel. An error occurs in this case when M is not an integer multiple of N.

Suppose you want to back propagate a M-by-N signal. When N>1, your signal has N channels. When N = 1, your signal has M channels. The value of the Minimum, Maximum, Mean, or Variance parameter can be a scalar or a vector of length equal to the number of channels. You can specify these parameters as either row or column vectors, except when the signal is a row vector. In this case, the Minimum, Maximum, Mean, or Variance parameter must also be specified as a row vector.

Sample mode

The sample mode, Continuous or Discrete. This parameter is enabled when the Inherit output port attributes check box is cleared.

Sample time

The sample period, T_s, of the random output sequence. The output frame period is $M*T_s$. This parameter is enabled when the Inherit output port attributes check box is cleared.

Samples per frame

The number of samples, M, in each output frame. When the value of this parameter is 1, the block outputs a sample-based signal.

This parameter is enabled when the Inherit output port attributes check box is cleared.

Output data type

The data type of the output, single- precision or double-precision. This parameter is enabled when the Inherit output port attributes check box is cleared.

Output complexity

The complexity of the output, Real or Complex. This parameter is enabled when the Inherit output port attributes check box is cleared.

Supported Data Types

- Double-precision floating-point
- Single-precision floating-point

See Also

Discrete Impulse	Signal Processing Blockset
Maximum	Signal Processing Blockset
Minimum	Signal Processing Blockset

Signal From Workspace	Signal Processing Blockset
Standard Deviation	Signal Processing Blockset
Variance	Signal Processing Blockset
Constant	Simulink
Random Number	Simulink
Signal Generator	Simulink
rand	MATLAB
randn	MATLAB

Real Cepstrum

Purpose

Compute real cepstrum of input

Library

Transforms

dspxfrm3

Description

The Real Cepstrum block computes the real cepstrum of each channel in the real-valued M-by-N input matrix, `u`. For both sample-based and frame-based inputs, the block assumes that each input column is a frame containing M consecutive samples from an independent channel. The block does not accept complex-valued inputs.

The output is a real M_o-by-N matrix, where you specify M_o in the FFT length parameter. Each output column contains the length-M_o real cepstrum of the corresponding input column.

```
y = real(ifft(log(abs(fft(u,Mo)))))
```

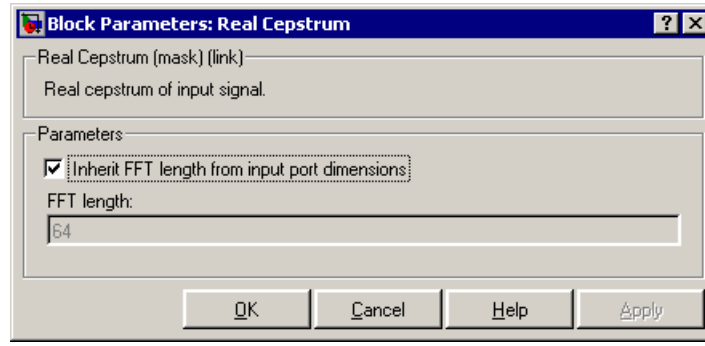
or, more compactly,

```
y = rceps(u,Mo)
```

When you select the Inherit FFT length from input port dimensions check box, the output frame size matches the input frame size (M_o=M). In this case, a sample-based length-M row vector input is processed as a single channel, that is, as an M-by-1 column vector, and the output is a length-M row vector. A 1-D vector input is always processed as a single channel, and the output is a 1-D vector.

The output is always sample based, and the output port rate is the same as the input port rate.

Dialog Box



Inherit FFT length from input port dimensions

When selected, matches the outpu t frame size to the input frame size.

FFT length

The number of frequency points at which to compute the FFT, which is also the output frame size, M_o. This parameter is available when you do not select Inherit FFT length from input port dimensions .

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Complex Cepstrum	Signal Processing Blockset
DCT	Signal Processing Blockset
FFT	Signal Processing Blockset
rceps	Signal Processing Toolbox

Reciprocal Condition

Purpose

Compute reciprocal condition of square matrix in 1-norm

Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

dspmtrx3

Description

The Reciprocal Condition block computes the reciprocal of the condition number for a square input matrix A.

```
y = rcond(A)        % Equivalent MATLAB code
```

or

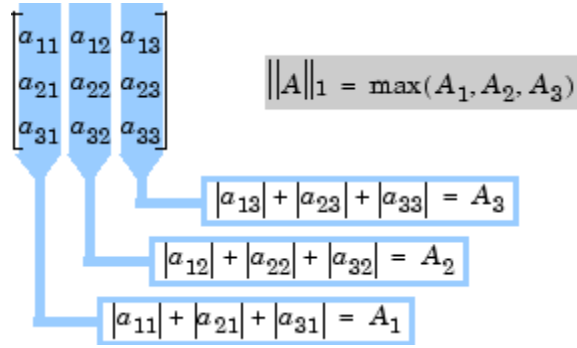
$$y = \frac{1}{\kappa} = \frac{1}{\left\|A^{-1}\right\|_1 \left\|A\right\|_1}$$

where κ is the condition number ($\kappa \geq 1$), and y is the scalar sample-based output (0 $\leq$ y < 1).

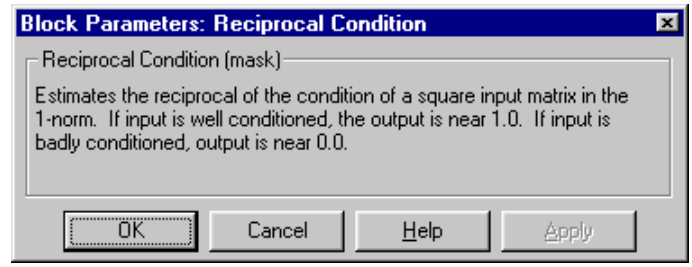
The matrix 1-norm, $\|A\|_1$, is the maximum column-sum in the M-by-M matrix A.

$$\|A\|_1 = 1 \leq \max_{j \leq M} \sum_{i=1}^{M} \left|a_{ij}\right|$$

For a 3-by-3 matrix:



Dialog Box



References

Golub, G. H., and C. F. Van Loan. Matrix Computations . 3rd ed. Baltimore, MD: Johns Hopkin s University Press, 1996.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Matrix 1-Norm	Signal Processing Blockset
Normalization	Signal Processing Blockset
rcond	MATLAB

Remez FIR Filter Design (Obsolete)

Purpose

Design and apply equiripple FIR filter

Library

dspobslib

Description

Note The Remez FIR Filter Design block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the Digital Filter block.

The Remez FIR Filter Design block implements the Parks-McClellan algorithm to design and apply a linear-phase filter with an arbitrary multiband magnitude response. The filter design, which uses the Signal Processing Toolbox firpm function, minimizes the maximum error between the desired frequency response and the actual frequency response. Such filters are called equiripple due to the equiripple behavior of their approximation error. The block applies the filter to a discrete-time input using the Direct-Form II Transpose Filter block.

An M-by-N sample-based matrix input is treated as M*N independent channels, and an M-by-N frame-based matrix input is treated as N independent channels. In both cases, the block filters each channel independently over time, and the output has the same size and frame status as the input.

The Filter type parameter allows you to specify one of the following filters:

- Multiband

  The multiband filter has an arbitrary magnitude response and linear phase.

- Differentiator

  The differentiator filter approximates the ideal differentiator. Differentiators are antisymmetric FIR filters with approximately linear magnitude responses. To obtain the correct derivative, scale

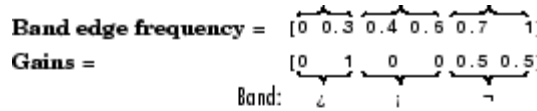
the Gains at these frequencies vector by πF_s rad/s, where F_s is the sample frequency in Hertz.

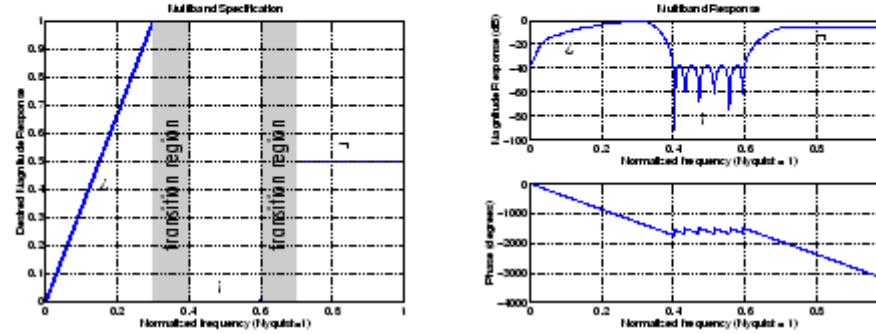
- Hilbert Transformer

  The Hilbert transformer filter approximates the ideal Hilbert transformer. Hilbert transformers are antisymmetric FIR filters with approximately constant magnitude.

The Band-edge frequency vector parameter is a vector of frequency points in the range 0 to 1, where 1 corresponds to half the sample frequency. Each band is defined by the two bounding frequencies, so this vector must have even length . Frequency points must appear in ascending order. The Gains at these frequencies parameter is a vector of the same size containing the desired magnitude response at the corresponding points in the Band-edge frequency vector .

Each odd-indexed frequency-amplitude pair defines the left endpoint of a line segment representing the desired magnitude response in that frequency band. The corresponding even-indexed frequency-amplitude pair defines the right endpoint. Between the frequency bands specified by these end-points, there may be undefined sections of the specified frequency response. These are called “don’t care” or “transition” regions, and the magnitude response in these areas is a by-product of the optimization in the other specified frequency ranges.





The Weights parameter is a vector that specifies the emphasis to be placed on minimizing the error in certain frequency bands relative to others. This vector specifies one weight per band, so it is half the length of the Band-edge frequency vector and Gains at these frequencies vectors.

In most cases, differentiators and Hilbert transformers have only a single band, so the weight is a scalar value that does not affect the final filter. However, the Weights parameter is useful when using the block to design an antisymmetric multiband filter, such as a Hilbert transformer with stopbands.

Examples

Example 1: Multiband

Consider a lowpass filter with a transition band in the normalized frequency range 0.4 to 0.5, and 10 times greater error minimization in the stopband than in the passband.

In this case:

- Filter type = Multiband

- Band-edge frequency vector = [0 0.4 0.5 1]
- Gains at these frequencies = [1 1 0 0]
- Weights = [1 10]

Example 2: Differentiator

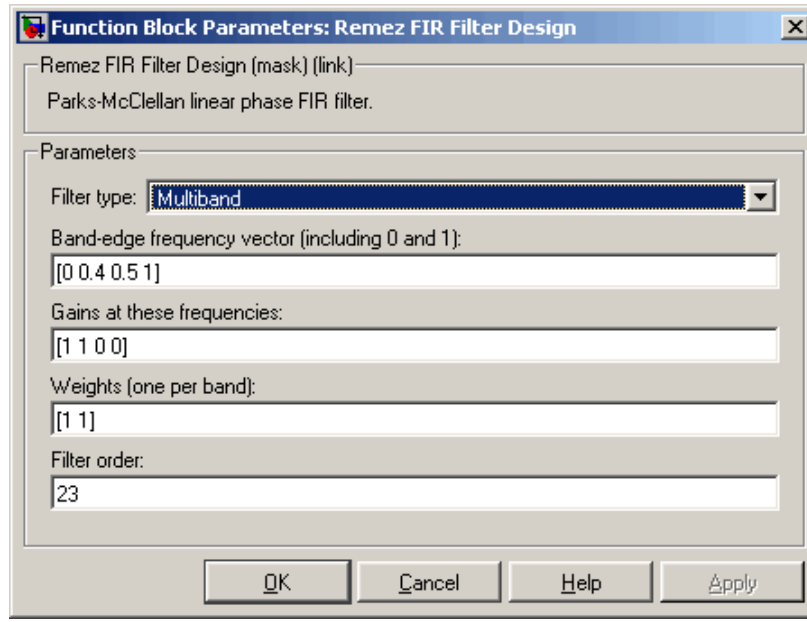
Assume the specifications for a differentiator filter require it to have order 21. The "ramp" response extends over the entire frequency range. In this case, specify:

- Filter type = Differentiator
- Band-edge frequency vector = [0 1]
- Gains at these frequencies = [0 pi*Fs]
- Filter order = 21

For a type III even order filter, the differentiation band should stop short of half the sample frequency. For example, if the filter order is 20, you could specify the block parameters as follows:

- Filter type = Differentiator
- Band-edge frequency vector = [0 0.9]
- Gains at these frequencies = [0 0.9*pi*Fs]
- Filter order = 20

Dialog Box



Filter type
: The Þlter type. Tunable.

Band-edge frequency vector
: A vector of frequency points, in ascending order, in the range 0 to 1. The value 1 corresponds to half the s ample frequency. This vector must have even length. Tunable.

Gains at these frequencies
: A vector of frequency-response m agnitudes corresponding to the points in the Band-edge frequency vector. This vector must be the same length as the Band-edge frequency vector. Tunable.

Weights
: A vector containing one weight for each frequency band. This vector must be half the length of the Band-edge frequency and Gains at these frequencies vectors. Tunable.

Filter order
: The filter order.

References

Oppenheim, A. V. and R. W. Schafer. Discrete-Time Signal Processing . Englewood Cliffs, NJ: Prentice Hall, 1989.

Proakis, J. and D. Manolakis. Digital Signal Processing. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

Repeat

Purpose

Resample input at higher rate by repeating values

Library

Signal Operations

dspsigops

Description

The Repeat block upsamples each channel of the M_i-by-N input to a rate L times higher than the input sample rate by repeating each consecutive input sample L times at the output. You specify the integer L in the Repetition count parameter.

This block supports trigg ered subsystems if, for Frame-based mode, you select Maintain input frame rate .

Sample-Based Operation

When the input is sample based, the block treats each of the M*N matrix elements as an independ ent channel, and upsamples each channel over time. The Frame-based mode parameter must be set to Maintain input frame size . The output sample rate is L times higher than the input sample rate ($T_{so} = T_{si}/L$), and the input and output sizes are identical.

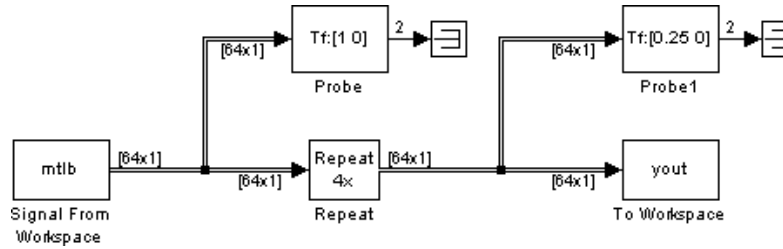
Frame-Based Operation

When the input is frame based, the block treats each of the N input columns as a frame containing M_i sequential time samples from an independent channel. The block upsamples each channel independently by repeating each row of the input matrix L times at the output. The Frame-based mode parameter determines how the block adjusts the rate at the output to accommodate the repeated rows. There are two available options:

- Maintain input frame size

  The block generates the output at the faster (upsa mpled) rate by using a proportionally shorter frame period at the output port than at the input port. For L repetitions of the input, the output frame period is L times shorter than the input frame period ($T_{fo} = T_p/L$), but the input and output frame sizes are equal.

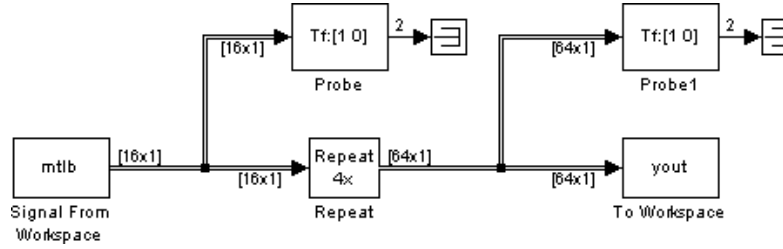
The model below shows a single-channel input with a frame period of 1 second being upsampled through 4-times repetition to a frame period of 0.25 second. The input and output frame sizes are identical.



- Maintain input frame rate

  The block generates the output at the faster (upsampled) rate by using a proportionally larger frame size than the input. For L repetitions of the input, the output frame size is L times larger than the input frame size ($M_o = M_i * L$), but the input and output frame rates are equal.

  The model below shows a single-channel input of frame size 16 being upsampled through 4-times repetition to a frame size of 64. The input and output frame rates are identical.



Zero Latency

The Repeat block has zero-tasking latency for all single-rate operations. The block is single-rate for the particular combinations of sampling mode and parameter settings shown in the table below.

Sampling Mode	Parameter Settings
Sample based	Repetition count parameter, L, is 1.
Frame based	Repetition count parameter, L, is 1, or Frame-based mode parameter is Maintain input frame rate.

The block also has zero latency for all multirate operations in the Simulink single-tasking mode.

Zero-tasking latency means that the block repeats the first input (received at t=0) for the first L output samples, the second input for the next L output samples, and so on. The Initial condition parameter value is not used.

Nonzero Latency

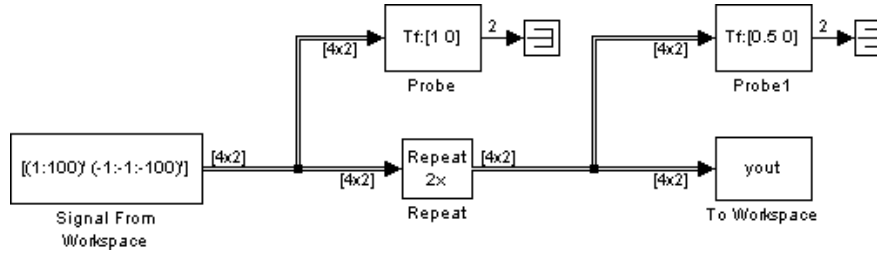
The Repeat block has tasking latency only for multirate operation in the Simulink multitasking mode:

- In sample-based mode, the initial condition for each channel is repeated for the first L output samples. The channel's first input appears as output sample L+1. The Initial condition value can be an M_i-by-N matrix containing one value for each channel, or a scalar to be applied to all signal channels.
- In frame-based mode, the first row of the initial condition matrix is repeated for the first L output samples, the second row of the initial condition matrix is repeated for the next L output samples, and so on. The first row of the first input matrix appears in the output as sample M_iL+1. The Initial condition value can be an M_i-by-N matrix, or a scalar to be repeated across all elements of the M_i-by-N matrix. See the example below for an illustration of this case.

Note For more information on latency and the Simulink tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Models with Multiple Sample Rates” in the Real-Time Workshop User’s Guide .

Examples

Construct the frame-bas ed model shown below.



Adjust the block parameters as follows.

- ConÞgure the Signal From Workspace block to generate a two-channel signal with frame si ze of 4 and sample period of 0.25. This represents an output frame period of 1 (0.25*4). The Þrst channel should contain the positive ramp signal 1, 2, ..., 100, and the second channel should contain the negative ramp signal -1, -2, ..., -100.
  - Signal = [(1:100)' (-1:-1:-100)']
  - Sample time = 0.25
  - Samples per frame = 4
- ConÞgure the Repeat block to upsample the two-channel input by increasing the output frame rate by a factor of 2 relative to the input frame rate. Set an initial condition matrix of

$$\begin{bmatrix} 11 & -11 \\ 12 & -12 \\ 13 & -13 \\ 14 & -14 \end{bmatrix}$$

- Repetition count = 2
- Initial condition = [11 -11;12 -12;13 -13;14 -14]
- Frame-based mode = Maintain input frame size

• Conþgure the Probe blocks by clearing the Probe width and Probe complex signal check boxes (if desired).

This model is multirate because the re are at least two distinct sample rates, as shown by the two Probe blocks. To run this model in the Simulink multitasking mode, in the Solver pane of the Conþguration Parameters dialog box, set the Type list to Fixed-step and set the Solver list to Discrete (no continuous states) . For the Tasking mode for periodic sample times parameter, select MultiTasking . Also set the Stop time to 30.

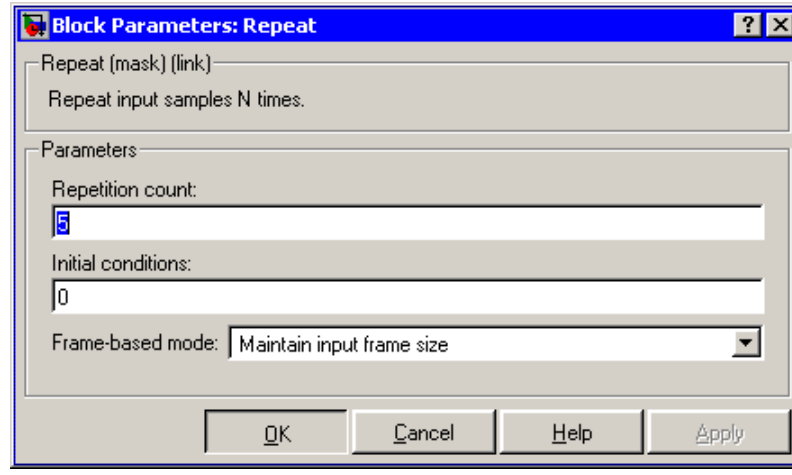
Run the model and look at the output, yout . The þrst few samples of each channel are shown below.

```
yout =
     11    -11
     11    -11
     12    -12
     12    -12
     13    -13
     13    -13
     14    -14
     14    -14
      1     -1
      1     -1
      2     -2
      2     -2
```

3 -3
3 -3
4 -4
4 -4
5 -5
5 -5

Since we ran this frame-based multi rate model in multitasking mode, the block repeats each row of the ini tial condition matrix for L output samples, where L is the Repetition count of 2. The Þrst row of the Þrst input matrix appears in the output as sample 9 ,that is, sample M¡L+1, where M ¡ is the input frame size.

Dialog Box



Repetition count

The integer number of times, L, that the input value is repeated at the output. This is the factor by which the output frame size or sample rate is increased.

Initial conditions

The value with which the block is initialized for cases of nonzero latency; a scalar or matrix.

Frame-based mode

For frame-based operation, the method by which to implement the repetition (upsampling): Maintain input frame size that is, increase the frame rate, or Maintain input frame rate , that is, increase the frame size. The Frame-based mode parameter must be set to Maintain input frame size for sample-base inputs.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

FIR Interpolation	Signal Processing Blockset
Upsample	Signal Processing Blockset

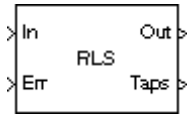
Purpose

Compute filter estimates for input using RLS adaptive filter algorithm

Library

dspobslib

Description



Note The RLS Adaptive Filter block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the RLS Filter block.

The RLS Adaptive Filter block recursively computes the recursive least squares (RLS) estimate of the FIR filter coefficients.

The corresponding RLS filter is expressed in matrix form as

$$k(n) = \frac{\lambda^{-1}P(n-1)u(n)}{1+\lambda^{-1}u^{H}(n)P(n-1)u(n)}$$
$$y(n) = \hat{w}^{H}(n-1)u(n)$$
$$e(n) = d(n) - y(n)$$
$$\hat{w}(n) = \hat{w}(n-1) + k(n)e^{*}(n)$$
$$P(n) = \lambda^{-1}P(n-1) - \lambda^{-1}k(n)u^{H}(n)P(n-1)$$

where λ^{-1} denotes the reciprocal of the exponential weighting factor. The variables are as follows

Variable	Description
n	The current algorithm iteration
u(n)	The buffered input samples at step n
P(n)	The inverse correlation matrix at step n
k(n)	The gain vector at step n
$\hat{w}(n)$	The vector of filter-tap estimates at step n

Variable	Description
y(n)	The filtered output at step n
e(n)	The estimation error at step n
d(n)	The desired response at step n
λ	The exponential memory weighting factor

The block icon has port labels corresponding to the inputs and outputs of the RLS algorithm. Note that inputs to the In and Err ports must be sample-based scalars. The signal at the Out port is a scalar, while the signal at the Taps port is a sample-based vector.

Block Ports	Corresponding Variables
In	u, the scalar input, which is internally buffered into the vector u(n)
Out	y(n), the filtered scalar output
Err	e(n), the scalar estimation error
Taps	$\hat{w}(0)$, the vector of filter-tap estimates

An optional Adapt input port is added when you select the Adapt input check box in the dialog box. When this port is enabled, the block continuously adapts the filter coefficients while the Adapt input is nonzero. A zero-valued input to the Adapt port causes the block to stop adapting, and to hold the filter coefficients at their current values until the next nonzero Adapt input.

The implementation of the algorithm in the block is optimized by exploiting the symmetry of the inverse correlation matrix P(n). This decreases the total number of computations by a factor of two.

The FIR filter length parameter specifies the length of the filter that the RLS algorithm estimates. The Memory weighting factor corresponds to λ in the equations, and specifies how quickly the filter

"forgets" past sample information. Setting $\lambda=1$ specifies an infinite memory; typically, $0.95 \leq \lambda \leq 1$.

The Initial value of filter taps specifies the initial value $\hat{w}(0)$ as a vector, or as a scalar to be repeated for all vector elements. The initial value of P(n) is

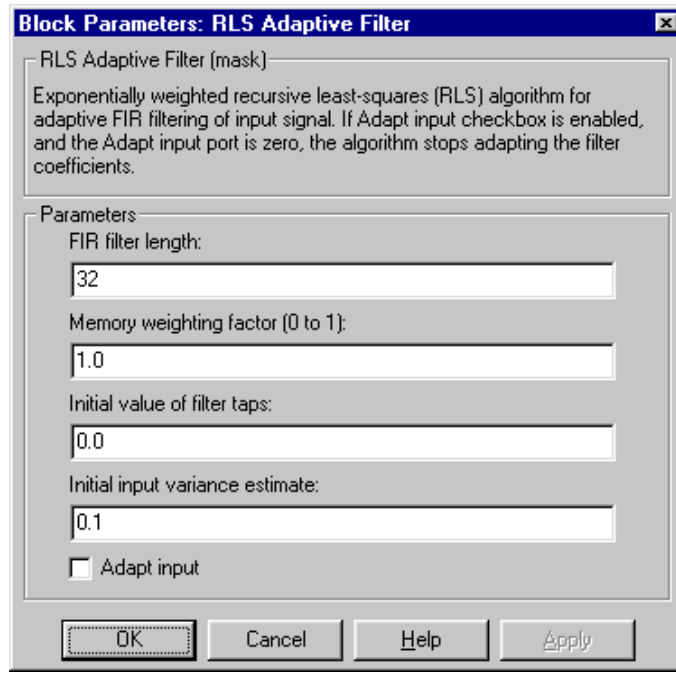
$$I \frac{1}{\hat{\sigma}^2}$$

where you specify $\hat{\sigma}^2$ in the Initial input variance estimate parameter.

Examples

The rlsdemo demo illustrates a noise cancellation system built around the RLS Adaptive Filter block.

Dialog Box



FIR filter length
: The length of the FIR filter.

Memory weighting factor
: The exponential weighting factor, in the range [0,1]. A value of 1 specifies an infinite memory. Tunable.

Initial value of filter taps
: The initial FIR filter coefficients.

Initial input variance estimate
: The initial value of 1/P(n).

Adapt input
: Enables the Adapt port.

References

Haykin, S. Adaptive Filter Theory . 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1996.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Kalman Adaptive Filter (Obsolete)	Signal Processing Blockset
LMS Adaptive Filter (Obsolete)	Signal Processing Blockset

See “Adaptive Filters” for related information.

Purpose

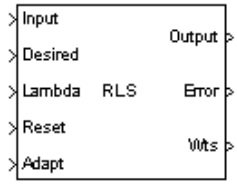
Compute filtered output, filter error, and filter weights for given input and desired signal using RLS adaptive filter algorithm

Library

Filtering / Adaptive Filters

dspadpt3

Description



RLS Filter

The RLS Filter block recursively computes the least squares estimate (RLS) of the FIR filter weights. The block estimates the filter weights, or coefficients, needed to convert the input signal into the desired signal. Connect the signal you want to filter to the Input port. This input signal can be a sample-based scalar or a single-channel frame-based signal. Connect the signal you want to model to the Desired port. The desired signal must have the same data type, frame status, complexity, and dimensions as the input signal. The Output port outputs the filtered input signal, which can be sample or frame based. The Error port outputs the result of subtracting the output signal from the desired signal.

The corresponding RLS filter is expressed in matrix form as

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^{H}(n)\mathbf{P}(n-1)\mathbf{u}(n)}$$
$$y(n) = \mathbf{w}(n-1)\mathbf{u}(n)$$
$$e(n) = \boldsymbol{d}(n) - y(n)$$
$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}^{H}(n)e(n)$$
$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^{H}(n)\mathbf{P}(n-1)$$

where λ^{-1} denotes the reciprocal of the exponential weighting factor. The variables are as follows

Variable	Description
n	The current time index
u(n)	The vector of buffered input samples at step n

Variable	Description
P(n)	The inverse correlation matrix at step n
k(n)	The gain vector at step n
$\mathbf{w}(n)$	The vector of filter-tap estimates at step n
y(n)	The filtered output at step n
e(n)	The estimation error at step n
d(n)	The desired response at step n
λ	The forgetting factor

The implementation of the algorithm in the block is optimized by exploiting the symmetry of the inverse correlation matrix P(n). This decreases the total number of computations by a factor of two.

Use the **Filter length** parameter to specify the length of the filter weights vector.

The **Forgetting factor (0 to 1)** parameter corresponds to λ in the equations. It specifies how quickly the filter "forgets" past sample information. Setting $\lambda=1$ specifies an infinite memory. Typically, $1-\frac{1}{2L}<\lambda<1$, where L is the filter length. You can specify a forgetting factor using the input port, Lambda, or enter a value in the **Forgetting factor (0 to 1)** parameter in the Block Parameters: RLS Filter dialog box.

Enter the initial filter weights, $\hat{w}(0)$, as a vector or a scalar for the **Initial value of filter weights** parameter. When you enter a scalar, the block uses the scalar value to create a vector of filter weights. This vector has length equal to the filter length and all of its values are equal to the scalar value.

The initial value of P(n) is

$$\frac{1}{\sigma^2} I$$

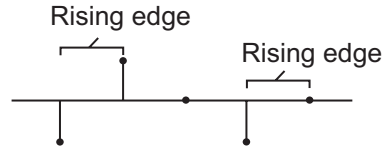
where you specify σ^2 in the Initial input variance estimate parameter.

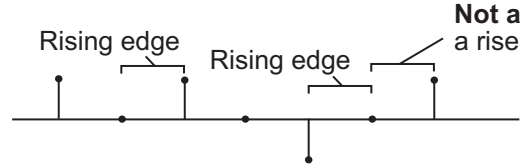
When you select the Adapt port check box, an Adapt port appears on the block. When the input to this port is nonzero, the block continuously updates the filter weights. When the input to this port is zero, the filter weights remain at their current values.

When you want to reset the value of the filter weights to their initial values, use the Reset input parameter. The block resets the filter weights whenever a reset event is detected at the Reset port. The reset signal rate must be the same rate as the data signal input.

From the Reset input list, select None to disable the Reset port. To enable the Reset port, select one of the following from the Reset input list:

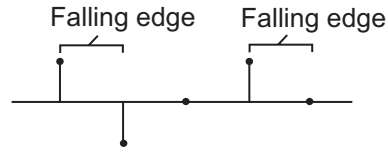
- Rising edge — Triggers a reset operation when the Reset input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero; see the following figure

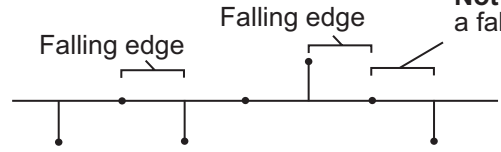




Not a rising edge because it continues a rise from a negative value to zero.

- Falling edge — Triggers a reset operation when the Reset input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero; see the following figure





Not a falling edge because it continues a fall from a positive value to zero.

- Either edge — Triggers a reset operation when the Reset input is a Rising edge or Falling edge, as described above
- Non-zero sample — Triggers a reset operation at each sample time that the Reset input is not zero

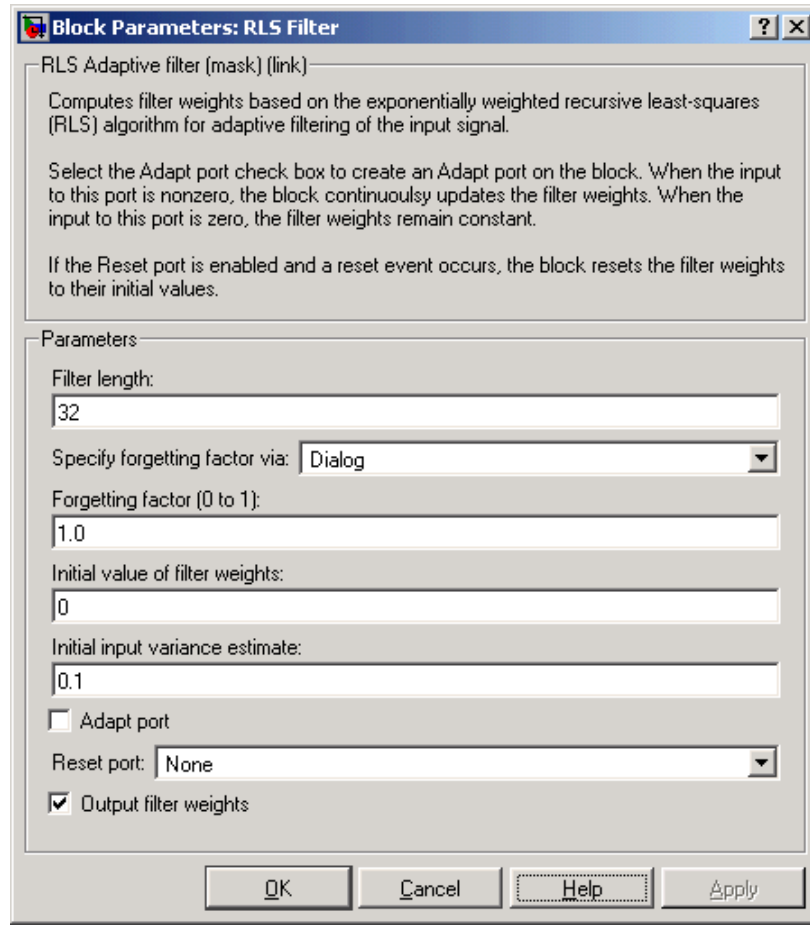
Note When running simulations in the Simulink MultiTasking mode, sample-based reset signals have a one-sample latency, and frame-based reset signals have one frame of latency. Thus, there is a one-sample or one-frame delay between the time the block detects a reset event, and when it applies the reset. For more information on latency and the Simulink tasking modes, see "Excess Algorithmic Delay (Tasking Latency)" and "Models with Multiple Sample Rates" in the Real-Time Workshop User's Guide .

Select the Output filter weights check box to create a Wts port on the block. For each iteration, the block outputs the current updated filter weights from this port.

Examples

The rlsdemo demo illustrates a noise cancellation system built around the RLS Filter block.

Dialog Box



Filter length

Enter the length of the FIR filter weights vector.

Specify forgetting factor via

Select Dialog to enter a value for the forgetting factor in the Block parameters: RLS Filter dialog box. Select Input port to specify the forgetting factor using the Lambda input port.

Forgetting factor (0 to 1)
: Enter the exponential weighting factor in the range $0 \leq \lambda \leq 1$. A value of 1 specifies an infinite memory. Tunable.

Initial value of filter weights
: Specify the initial values of the FIR filter weights.

Initial input variance estimate
: The initial value of 1/P(n).

Adapt port
: Select this check box to enable the Adapt input port.

Reset input
: Select this check box to enable the Reset input port.

Output filter weights
: Select this check box to export the filter weights from the Wts port.

References

Hayes, M.H. Statistical Digital Signal Processing and Modeling. New York: John Wiley & Sons, 1996.

Supported Data Types

- Double-precision floating point
- Single-precision floating point

See Also

Kalman Adaptive Filter (Obsolete)	Signal Processing Blockset
LMS Filter	Signal Processing Blockset
Block LMS Filter	Signal Processing Blockset
Fast Block LMS Filter	Signal Processing Blockset

See “Adaptive Filters” for related information.

Purpose

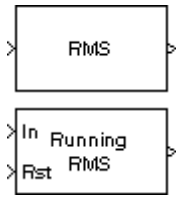
Compute root-mean-square value of input or sequence of inputs

Library

Statistics

dspstat3

Description



The RMS block computes the RMS value of each row or column of the input, along vectors of a specified dimension of the input, or of the entire input. The RMS block can also track the RMS value in a sequence of inputs over a period of time. The Running RMS parameter selects between basic operation and running operation.

Basic Operation

When you do not select the Running RMS check box, the block computes the RMS value of each row or column of the input, along vectors of a specified dimension of the input, or of the entire input at each individual sample time, and outputs the array y. Each element in y is the RMS value of the corresponding column, row, vector, or entire input. The output y depends on the setting of the Find the RMS value over parameter. For example, consider a 3-dimensional input signal of size M-by-N-by-P:

- Entire input — The output at each sample time is a scalar that contains the RMS value of the entire input. In this mode, the output is always sample based.
- Each row — The output at each sample time consists of an M-by-1-by-P array, where each element contains the RMS value of each vector over the second dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is an M-by-1 column vector. In this mode, the frame status of the output is the same as that of the input.
- Each column — The output at each sample time consists of a 1-by-N-by-P array, where each element contains the RMS value of each vector over the first dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is a 1-by-N

row vector. In this mode, the fram e status of the output is the same as that of the input.

For convenience, length- M 1-D vector inputs are treated as M-by-1 column vectors when the block is in this mode. Sample-based length- M row vector inputs are also treated as M-by-1 column vectors when the Treat sample-based row input as a column check box is selected.

- Specified dimension — The output at each sample time depends on Dimension . If Dimension is set to 1, the output is the same as that when you select Each column . If Dimension is set to 2, the output is the same as when you select Each row . If Dimension is set to 3, the output at each sample time is an M-by-N matrix containing the RMS value of each vector over the third dimension of the input. In this mode, the frame status of the output is the same as that of the input.

The RMS value of the jth column of an M-by-N input matrix u is given by

$$y_j = \sqrt{\frac{\sum_{i=1}^{M} \left| u_{ij} \right|^2}{M}} \qquad 1 \le j \le N$$

```
y = sqrt(sum(u.*conj(u))/size(u,1))        % Equivalent MATLAB code
```

Running Operation

When you select the Running RMS check box, the block tracks the RMS value of successive inputs to the block. For sample-based M-by-N inputs, the output is a sample-based M-by-N matrix, with each element y_{ij} containing the RMS value of element u_{ij} over all inputs since the last reset. For frame-based M-by-N inputs, the output is a frame-based M-by-N matrix with each element y_{ij} containing the RMS value of the jth column over all inputs since the last reset, up to and including element u_{ij} of the current input.

N-D signals cannot be frame based. When the Running RMS check box is selected, each element of the N-D signal is treated as a separate channel.

There are $\prod d_i$ channels, where d_i is the size of the ith dimension.

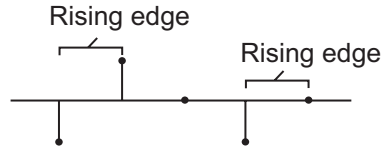
Resetting the Running RMS

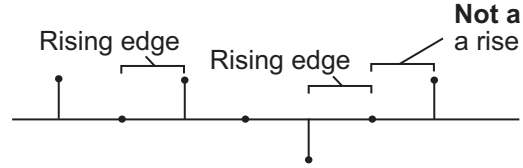
The block resets the running RMS whenever a reset event is detected at the optional Rst port. The reset sample time must be a positive integer multiple of the input sample time.

When the block is reset for sample-based inputs, the running RMS for each channel is initialized to the value in the corresponding channel of the current input. For frame-based inputs, the running RMS for each channel is initialized to the earliest value in each channel of the current input.

You specify the reset event in the Reset port parameter:

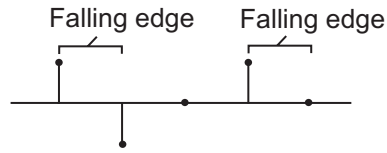
- None disables the Rst port.
- Rising edge — Triggers a reset operation when the Rst input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following figure)

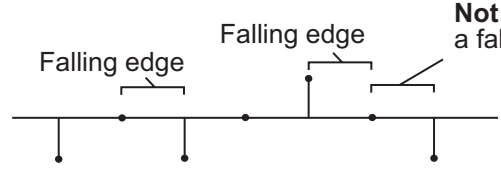




Not a rising edge because it continues a rise from a negative value to zero.

- Falling edge — Triggers a reset operation when the Rst input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)





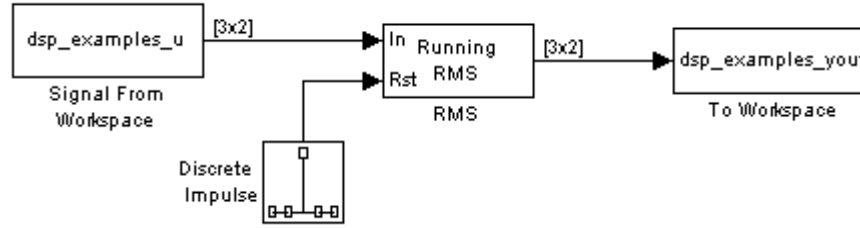
Not a falling edge because it continues a fall from a positive value to zero.

- Either edge — Triggers a reset o peration when the Rst input is a Rising edge or Falling edge , as described earlier
- Non-zero sample — Triggers a reset opera tion at each sample time that the Rst input is not zero

Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see "Excess Algorithmic Delay (Tasking Latency)" and "Models with Multiple Sample Rates" in the Real-Time Workshop User's Guide.

Examples

The RMS block in the next model calculates the running RMS of a frame-based 3-by-2 (two-channel) matrix input, u. The running RMS is reset at t=2 by an impulse to the block's Rst port.



The RMS block has the following settings:

- Running RMS = Select this check box.
- Reset port = Non-zero sample

The Signal From Workspace block has the following settings:

- Signal = dsp_examples_u
- Sample time = 1/3
- Samples per frame = 3

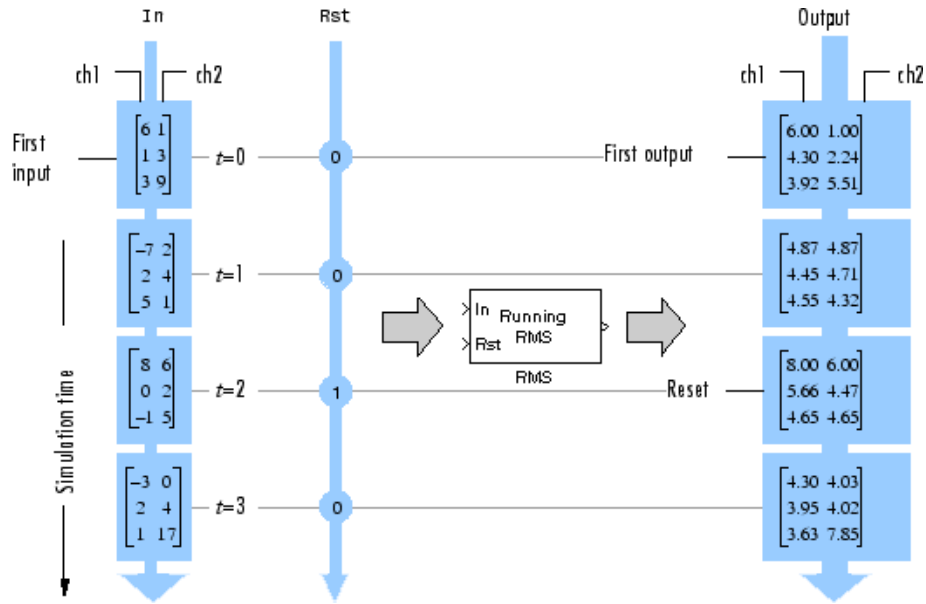
where

```
dsp_examples_u = [6 1 3 -7 2 5 8 0 -1 -3 2 1;1 3 9 2 4 1 6 2 5 0 4 17]'
```

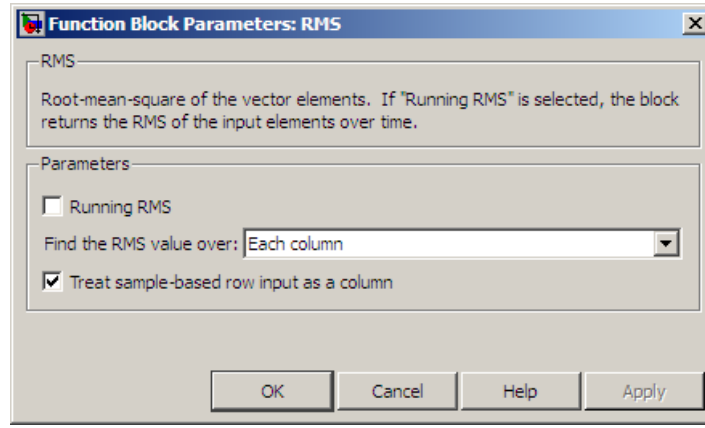
The Discrete Impulse block has the following settings:

- Delay (samples) = 2
- Sample time = 1
- Samples per frame = 1

The block's operation is shown in the next Þgure.



Dialog Box



Running RMS

Enables running operation when selected.

Reset port

Determines the reset event that causes the block to reset the running RMS. The reset signal rate must be a positive integer multiple of the rate of the data signal input. This parameter is enabled only when you set the Running RMS parameter. For more information, see “Resetting the Running RMS” on page 2-1097.

Find the RMS value over

Specify whether to find the RMS value along rows, columns, entire input, or the dimension specified in the Dimension parameter. For more information, see “Basic Operation” on page 2-1095.

Treat sample-based row input as a column

Select to treat sample-based length- M row vector inputs as M-by-1 column vectors. This parameter is only visible when the Find the RMS value over parameter is set to Each column .

Dimension

Specify the dimension (one-based value) of the input signal, over which the RMS value is computed. The value of this parameter

cannot exceed the number of dimensions in the input signal. This parameter is only visible when the Find the RMS value over parameter is set to Specified dimension .

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Boolean — The block accepts Boolean inputs to the Rst port.

See Also

Mean	Signal Processing Blockset
Variance	Signal Processing Blockset

Purpose

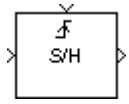
Sample and hold input signal

Library

Signal Operations

dspsigops

Description



The Sample and Hold block acquires the input at the signal port whenever it receives a trigger event at the trigger port (marked by). The block then holds the output at the acquired input value until the next triggering event occurs. When the acquired input is frame based, the output is frame based; otherwise, the output is sample based.

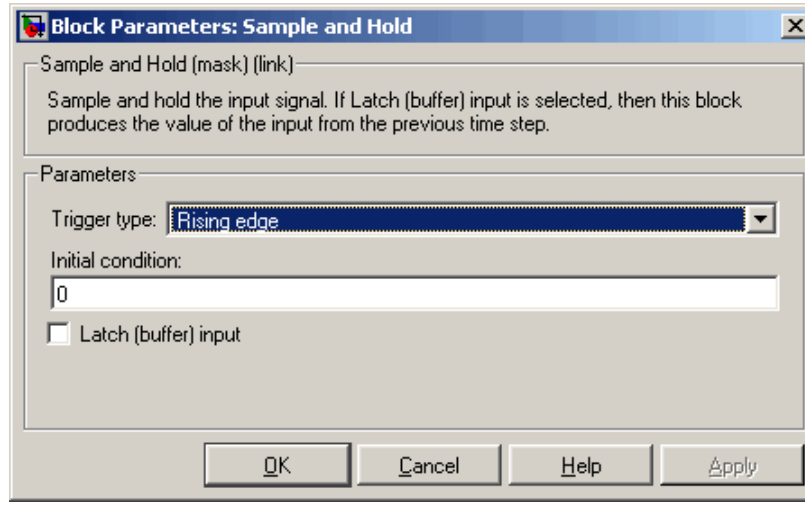
The trigger input must be a sample-based scalar with sample rate equal to the input frame rate at the signal port. You specify the trigger event in the Trigger type pop-up menu:

- Rising edge triggers the block to acquire the signal input when the trigger input rises from a negative value or zero to a positive value.
- Falling edge triggers the block to acquire the signal input when the trigger input falls from a positive value or zero to a negative value.
- Either edge triggers the block to acquire the signal input when the trigger input either rises from a negative value or zero to a positive value or falls from a positive value or zero to a negative value.

You specify the block's output prior to the first trigger event using the Initial condition parameter. When the acquired input is an M-by-N matrix, the Initial condition can be an M-by-N matrix, or a scalar to be repeated across all elements of the matrix. When the input is a length-M 1-D vector, the Initial condition can be a length-M row or column vector, or a scalar to be repeated across all elements of the vector.

If you select the Latch (buffer) input check box, the block outputs the value of the input from the previous time step until the next triggering event occurs. To use this block in a loop, select this check box.

Dialog Box



Trigger type
: The type of event that triggers the block to acquire the input signal.

Initial condition
: The block's output prior to the first trigger event.

Latch (buffer) input
: If you select this check box, the block outputs the value of the input from the previous time step until the next triggering event occurs.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Trigger	• Any data type supported by the Trigger block
Outputs	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Downsample Signal Processing Blockset

N-Sample Switch Signal Processing Blockset

Scalar Quantizer (Obsolete)

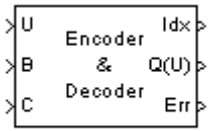
Purpose

Convert input signal into set of quantized output values or index values, or convert set of index values into quantized output signal

Library

dspobslib

Description



Scalar Quantizer

Note The Scalar Quantizer block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the Scalar Quantizer Encoder block or the Scalar Quantizer Decoder block.

The Scalar Quantizer block has three modes of operation. In Encoder mode, the block maps each input value to a quantization region by comparing the input value to the quantizer boundary points defined in the Boundary points parameter. The block outputs the index of the associated region. In Decoder mode, the block transforms the input index values into quantized output values, defined in the Codebook parameter. In the Encoder and Decoder mode, the block performs both the encoding and decoding operations. The block outputs the index values and the quantized output values.

You can select how you want to enter the Boundary points and/or Codebook values using the Source of quantizer parameters. When you select Specify via dialog, type the parameters into the block parameters dialog box. Select Input ports, and port B and/or C appears on the block. In Encoder and Encoder and decoder mode, the input to port B is used as the Boundary points. In Decoder and Encoder and decoder mode, the input to port C is used as the Codebook.

In Encoder and Encoder and decoder mode, the Boundary points are the values used to break up the input signal into regions. Each region is specified by an index number. When your first boundary point is -inf and your last boundary point is inf, your quantizer is unbounded. When your first and last boundary point is finite, your

quantizer is bounded. When only your first or last boundary point is -inf or inf, your quantizer is semi-bounded.

For instance, when your input signal ranges from 0 to 11, you can create a bounded quantizer using the following boundary points:

```
[0  0.5  3.7  5.8  6.0  11]
```

The boundary points can have equal or varied spacing. Any input values between 0 and 0.5 would correspond to index 0. Input values between 0.5 and 3.7 would correspond to index 1, and so on.

Suppose you wanted to create an unbounded quantizer with the following boundary points:

```
[-inf  0  2  5.5  7.1  10  inf]
```

When your input signal has values less than 0, these values would be assigned to index 0. When your input signal has values greater than 10, these values would be assigned to index 6.

When an input value is the same as a boundary point, the Tie-breaking rule parameter defines the index to which the value is assigned. When you want the input value to be assigned to the lower index value, select Choose the lower index. To assign the input value with the higher index, select Choose the higher index.

In Decoder and Encoder and decoder mode, the Codebook is a vector of quantized output values that correspond to each index value.

In Encoder and Encoder and decoder mode, the Searching method determines how the appropriate quantizer index is found. Select Linear and the Scalar Quantizer block compares the input value to the first region defined by the first two boundary points. When the input value does not fall within this region, the block then compares the input value to the next region. This process continues until the input value is determined to be within a region and is associated with the appropriate index value. The computational cost of this process is of the order P, where P is the number of boundary points.

Select Binary for the Searching method and the block compares the input value to the middle value of the boundary points vector. When the input value is larger than this boundary point, the block discards the boundary points that are lower than this middle value. The block then compares the input value to the middle boundary point of the new range, defined by the remaining boundary points. This process continues until the input value is associated with the appropriate index value. The computational cost of this process is of the order $\log_2 P$, where P is the number of boundary points. In most cases, the Binary option is faster than the Linear option.

In Decoder mode, the input to this block is a vector of index values, where $0 \leq \text{index} < N$ and N is the length of the codebook vector. Use the Action for out of range input parameter to determine what happens when an input index value is out of this range. When you want any index values less than 0 to be set to 0 and any index values greater than or equal to N to be set to N -1, select Clip . When you want to be warned when any index values less than 0 are set to 0 and any index values greater than or equal to N are set to N -1, select Clip and warn . When you want the simulation to stop and display an error when the index values are out of range, select Error .

In Encoder and decoder mode, you can select the Output the quantization error check box. The quantization error is the difference between the input value and the quantized output value. Select this check box to output the quantization error for each input value from the Err port on this block.

Data Type Support

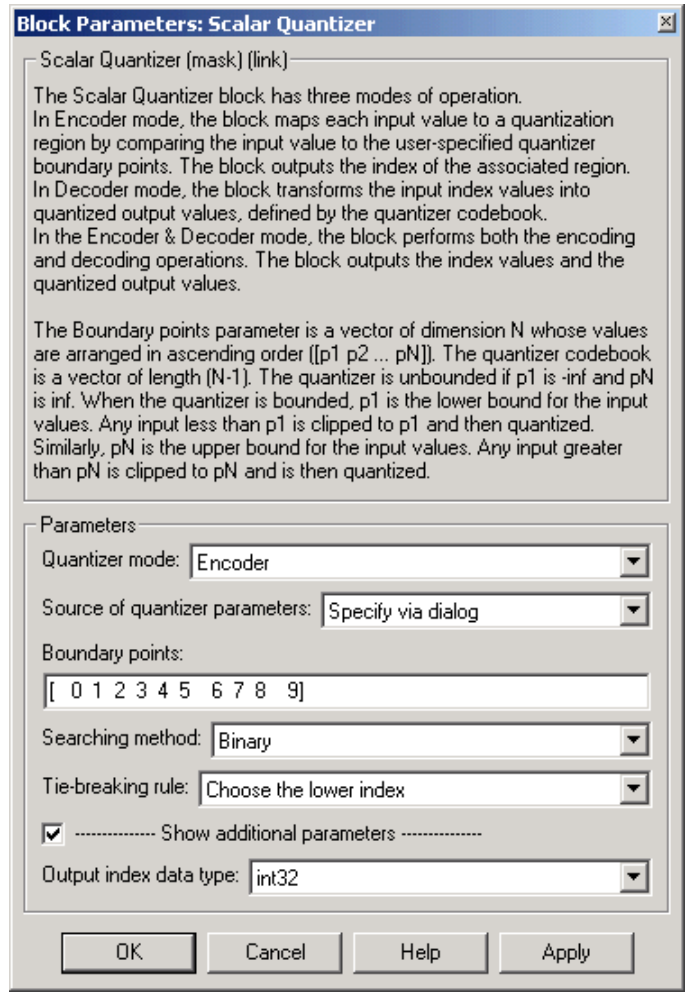
In Encoder mode, the input data values and the boundary points can be the input to the block at ports U and B. Similarly, in Encoder and decoder mode, the codebook values can also be the input to the block at port C. The data type of the input data values, boundary points, and codebook values can be double , single , uint8 , uint16 , uint32 , int8 , int16 , or int32 . In Decoder mode, the input to the block can be the index values and the codebook values. The data type of the index input to the block at port Idx can be uint8 , uint16 , uint32 , int8 , int16 , or

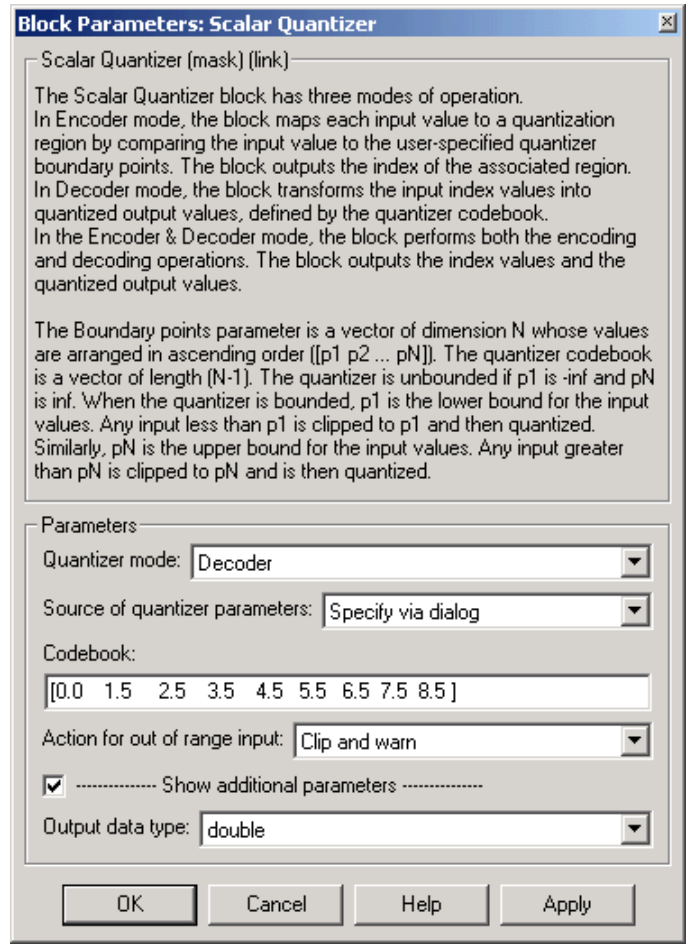
int32 . The data type of the codebook values can be double , single , uint8 , uint16 , uint32 , int8 , int16 , or int32 .

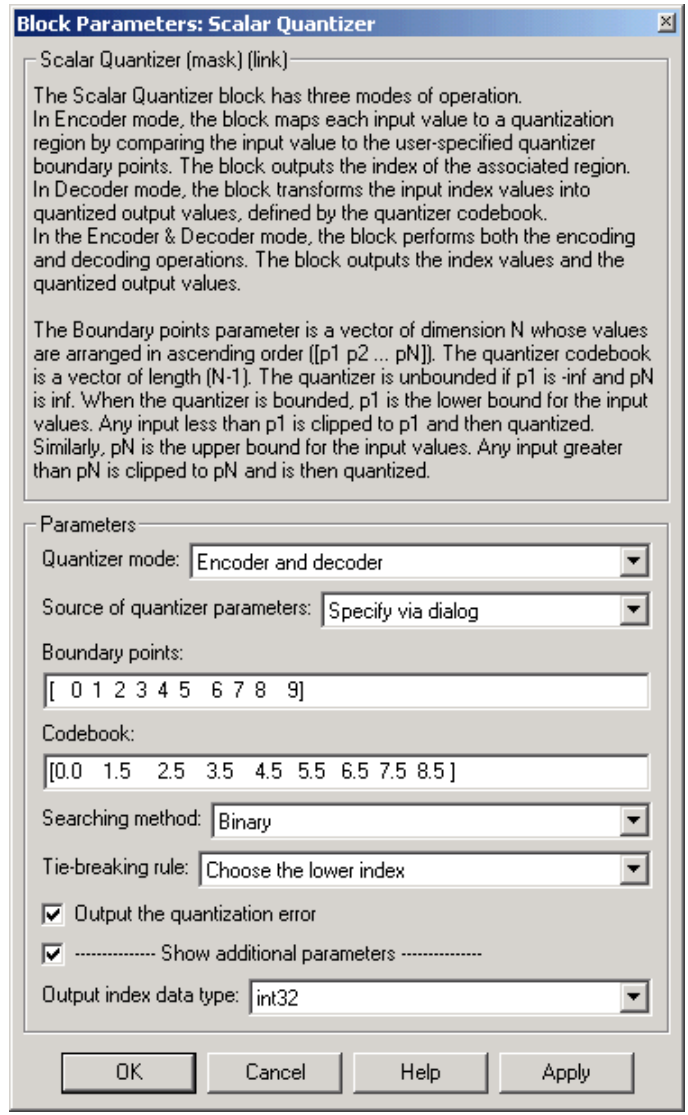
In Encoder mode, the output of the block is the index values. In Encoder and decoder mode, the output can also include the quantized output values and the quantization error. In Encoder and Encoder and decoder mode, use the Output index data type parameter to specify the data type of the index output from the block at port Idx. The data type of the index output can be uint8 , uint16 , uint32 , int8 , int16 , or int32 . The data type of the quantized output and the quantization error can be double , single , uint8 , uint16 , uint32 , int8 , int16 , or int32 . In Decoder mode, the output of the block is the quantized output values. Use the Output data type parameter to specify the data type of the quantized output values. The data type can be double , single , uint8 , uint16 , uint32 , int8 , int16 , int32 .

Note The input data, codebook values, b oundary points, quantization error, and the quantized output values must have the same data type whenever they are present in any of the quantizer modes.

Dialog Box


Block Parameters: Scalar Quantizer
Scalar Quantizer (mask) (link)
The Scalar Quantizer block has three modes of operation.
In Encoder mode, the block maps each input value to a quantization region by comparing the input value to the user-specified quantizer boundary points. The block outputs the index of the associated region.
In Decoder mode, the block transforms the input index values into quantized output values, defined by the quantizer codebook.
In the Encoder & Decoder mode, the block performs both the encoding and decoding operations. The block outputs the index values and the quantized output values.
The Boundary points parameter is a vector of dimension N whose values are arranged in ascending order ([p1 p2 ... pN]). The quantizer codebook is a vector of length (N-1). The quantizer is unbounded if p1 is -inf and pN is inf. When the quantizer is bounded, p1 is the lower bound for the input values. Any input less than p1 is clipped to p1 and then quantized. Similarly, pN is the upper bound for the input values. Any input greater than pN is clipped to pN and is then quantized.
Parameters
Quantizer mode: Encoder
Source of quantizer parameters: Specify via dialog
Boundary points:
[0 1 2 3 4 5 6 7 8 9]
Searching method: Binary
Tie-breaking rule: Choose the lower index
---------- Show additional parameters ----------
Output index data type: int32
OK
Cancel
Help
Apply


Block Parameters: Scalar Quantizer
Scalar Quantizer (mask) (link)
The Scalar Quantizer block has three modes of operation.
In Encoder mode, the block maps each input value to a quantization region by comparing the input value to the user-specified quantizer boundary points. The block outputs the index of the associated region.
In Decoder mode, the block transforms the input index values into quantized output values, defined by the quantizer codebook.
In the Encoder & Decoder mode, the block performs both the encoding and decoding operations. The block outputs the index values and the quantized output values.
The Boundary points parameter is a vector of dimension N whose values are arranged in ascending order ([p1 p2 ... pN]). The quantizer codebook is a vector of length (N-1). The quantizer is unbounded if p1 is -inf and pN is inf. When the quantizer is bounded, p1 is the lower bound for the input values. Any input less than p1 is clipped to p1 and then quantized. Similarly, pN is the upper bound for the input values. Any input greater than pN is clipped to pN and is then quantized.
Parameters
Quantizer mode: Decoder
Source of quantizer parameters: Specify via dialog
Codebook:
[0.0 1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5]
Action for out of range input: Clip and warn
---------- Show additional parameters ----------
Output data type: double
OK
Cancel
Help
Apply


Block Parameters: Scalar Quantizer
Scalar Quantizer (mask) (link)
The Scalar Quantizer block has three modes of operation.
In Encoder mode, the block maps each input value to a quantization region by comparing the input value to the user-specified quantizer boundary points. The block outputs the index of the associated region.
In Decoder mode, the block transforms the input index values into quantized output values, defined by the quantizer codebook.
In the Encoder & Decoder mode, the block performs both the encoding and decoding operations. The block outputs the index values and the quantized output values.
The Boundary points parameter is a vector of dimension N whose values are arranged in ascending order ([p1 p2 ... pN]). The quantizer codebook is a vector of length (N-1). The quantizer is unbounded if p1 is -inf and pN is inf. When the quantizer is bounded, p1 is the lower bound for the input values. Any input less than p1 is clipped to p1 and then quantized. Similarly, pN is the upper bound for the input values. Any input greater than pN is clipped to pN and is then quantized.
Parameters
Quantizer mode: Encoder and decoder
Source of quantizer parameters: Specify via dialog
Boundary points:
[0 1 2 3 4 5 6 7 8 9]
Codebook:
[0.0 1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5]
Searching method: Binary
Tie-breaking rule: Choose the lower index
Output the quantization error
---------- Show additional parameters ----------
Output index data type: int32
OK
Cancel
Help
Apply

Quantizer mode

Specify Encoder , Decoder , or Encoder and decoder as a mode of operation.

Source of quantizer parameters

Choose Specify via dialog to type the parameters into the block parameters dialog box. Select Input ports to specify the parameters using the block's input ports. In Encoder and Encoder and decoder mode, input the Boundary points using port B. In Decoder and Encoder and decoder mode, input the Codebook values using port C.

Boundary points

Enter a vector of values that represent the boundary points of the quantizer regions. Tunable.

Codebook

Enter a vector of quantized output values that correspond to each index value. Tunable.

Searching method

Select Linear and the block Þnds the region in which the input value is located using a linear search. Select Binary and the block Þnds the region in which the input value is located using a binary search.

Tie-breaking rule

Set this parameter to determin e the behavior of the block when the input value is the same as the boundary point. When you select Choose the lower index , the input value is assigned to lower index value. When you select Choose the higher index , the value is assigned to the higher index.

Action for out of range input

Choose the block's behavior when an input index value is out of range, where $0 \leq \text{index} < N$ and N is the length of the codebook vector. Select Clip , when you want any index values less than 0 to be set to 0 and any index values greater than or equal to N to be set to N -1. Select Clip and warn , when you want to be warned when any index values less than 0 are set to 0 and any

index values greater than or equal to N are set to N -1. Select Error , when you want the simulation to stop and display an error when the index values are out of range.

Output the quantization error

In Encoder and decoder mode, select this check box to output the quantization error from the Err port on this block.

Output index data type

In Encoder and Encoder and decoder mode, specify the data type of the index output from the block at port Idx. The data type can be uint8 , uint16 , uint32 , int8 , int16 , or int32 . This parameter becomes visible when you select the Show additional parameters check box.

Output data type

In Decoder mode, specify the data type of the quantized output. The data type can be uint8 , uint16 , uint32 , int8 , int16 , int32 , single , or double . This parameter becomes visible when you select Specify via dialog for the Source of quantizer parameters and you select the Show additional parameters check box.

References

Gersho, A. and R. Gray. Vector Quantization and Signal Compression . Boston: Kluwer Academic Publishers, 1992.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Fixed point
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

For more information on what data types are supported for each quantizer mode, see “Data Type Support” on page 2-1108.

See Also

Quantizer	Simulink
Scalar Quantizer Decoder	Signal Processing Blockset
Scalar Quantizer Design	Signal Processing Blockset
Scalar Quantizer Encoder	Signal Processing Blockset
Uniform Encoder	Signal Processing Blockset
Uniform Decoder	Signal Processing Blockset

Scalar Quantizer Decoder

Purpose

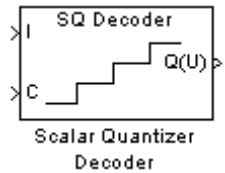
Convert each index value into quantized output value

Library

Quantizers

dspquant2

Description



The Scalar Quantizer Decoder block transforms the zero-based input index values into quantized output values. The set of all possible quantized output values is de Þned by the Codebook values parameter.

Use the Codebook values parameter to specify a m atrix containing all possible quantized output values. You can select how you want to enter the codebook values using the Source of codebook parameter. When you select Specify via dialog , type the codebook values into the block parameters dialog bo x. When you select Input port , port C appears on the block. The block uses the input to port C as the Codebook values parameter.

The input to this block is a vector of integer index values, where 0 $\leq$ index < N and N is the number of distinct codeword vectors in the codebook matrix. Use the Action for out of range index value parameter to determine what happens when an input index value is outside this range. When you want any index value less than 0 to be set to 0 and any index value greater than or equal to N to be set to N -1, select Clip . When you want to be warned when clipping occurs, select Clip and warn . When you want the simulation to stop and the block to display an error when the index values are out of range, select Error .

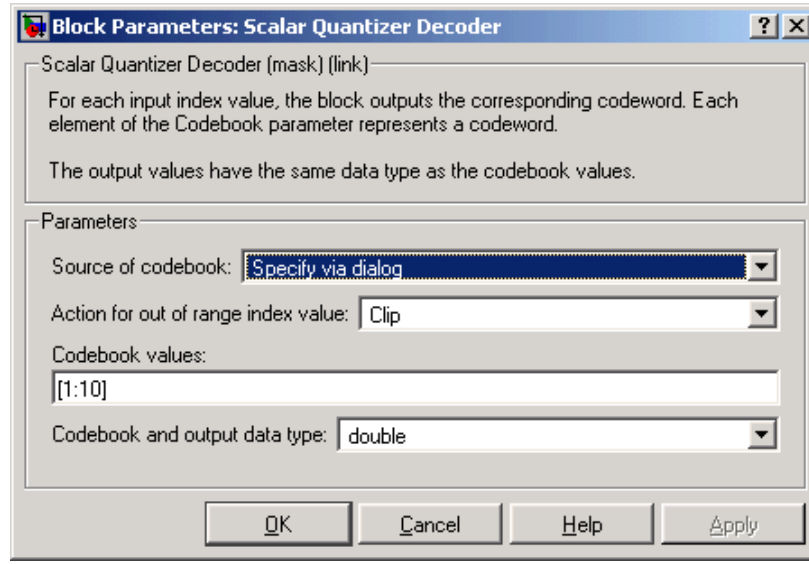
Data Type Support

The data type of the index values input at port I can be uint8 , uint16 , uint32 , int8 , int16 , or int32 . The data type of the codebook values input at port C can be double , single , or Fixed-point.

The output of the block is the quantized output values. If, for the Source of codebook parameter, you select Specify via dialog , the Codebook and output data type parameter appears. You can use this parameter to specify the data type of the codebook and quantized output values. In this case, the data type of the output values can be Same as input , double , single , Fixed-point , or User-defined .

If, for the Source of codebook parameter you select Input port, the quantized output values have the same data type as the codebook values input at port C.

Dialog Box



Source of codebook

Choose Specify via dialog to type the codebook values into the block parameters dialog box. Select Input port to specify the codebook using input port C.

Action for out of range index value

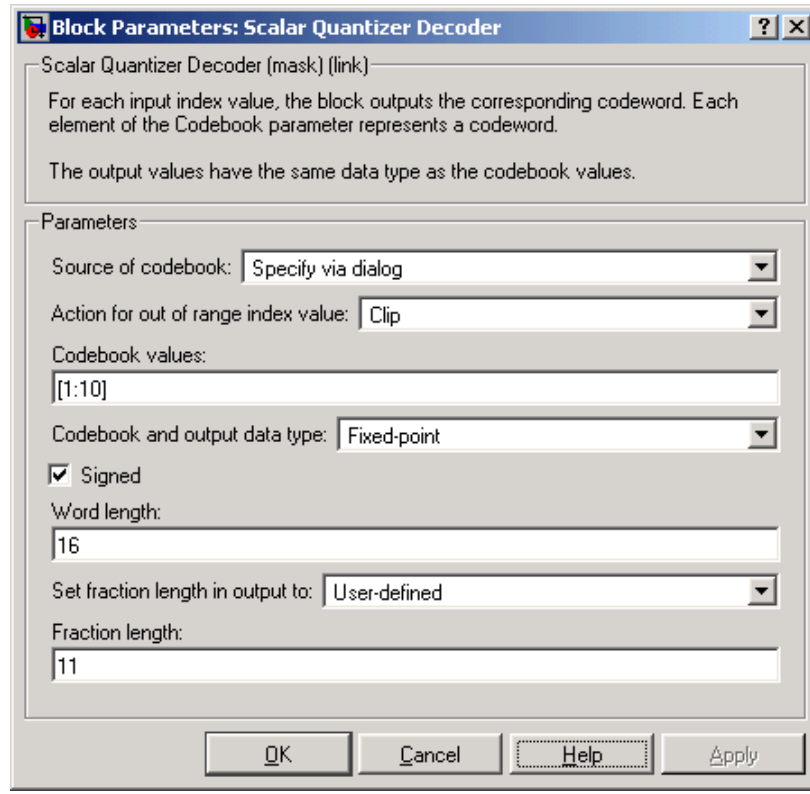
Use this parameter to determine the block's behavior when an input index value is out of range, where $0 \leq \text{index} < N$ and N is the length of the codebook vector. Select Clip, when you want any index values less than 0 to be set to 0 and any index values greater than or equal to N to be set to $N-1$. Select Clip and warn, when you want to be warned when clipping occurs. Select Error, when you want the simulation to stop and the block to display an error when the index values are outside the range.

Codebook values

Enter a vector of quantized output values that correspond to each index value. Tunable.

Codebook and output data type

Use this parameter to specify the data type of the codebook and quantized output values. The data type can be Same as input, double, single, Fixed-point, or User-defined. This parameter becomes visible when you select Specify via dialog for the Source of codebook parameter.



Signed

Select to output a signed fixed-point signal. Otherwise, the signal is unsigned. This parameter is only visible if, from the Codebook and output data type list, you select Fixed-point.

Word length

Specify the word length, in bits, of the fixed-point output data type. This parameter is only visible if, from the Codebook and output data type list, you select Fixed-point.

Set fraction length in output to

Specify the scaling of the fixed-point output by either of the following two methods:

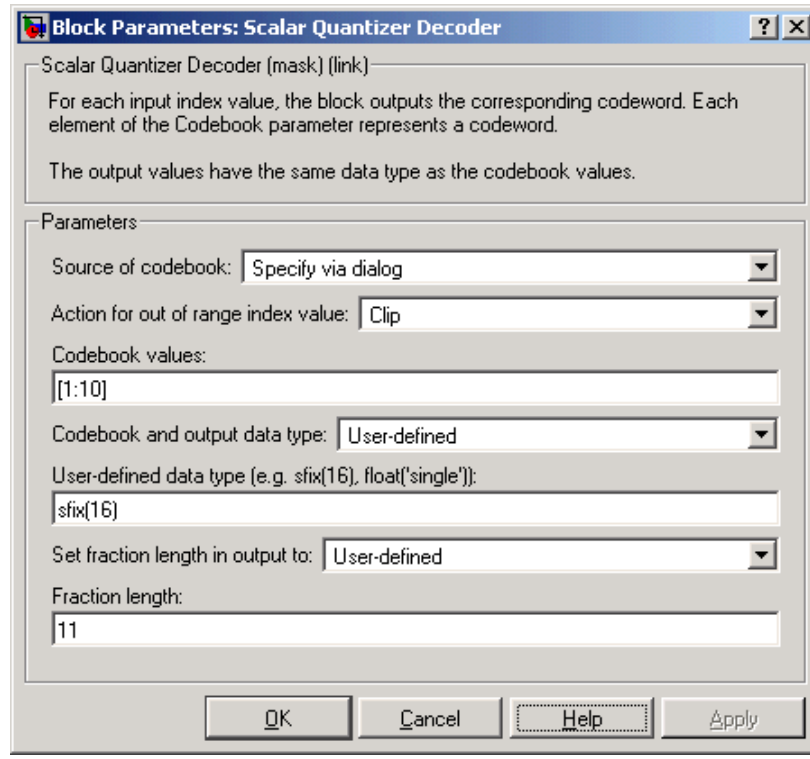
Choose Best precision to have the output scaling automatically set such that the output signal has the best possible precision.

Choose User-defined to specify the output scaling in the Fraction length parameter.

This parameter is only visible if, from the Codebook and output data type list, you select Fixed-point or when you select User-defined and the specified output data type is a fixed-point data type.

Fraction length

For fixed-point output data types, specify the number of fractional bits, or bits to the right of the binary point. This parameter is only visible when you select Fixed-point or User-defined for the Codebook and output data type parameter and User-defined for the Set fraction length in output to parameter.



User-defined data type

Specify any built-in or fixed-point data type. You can specify fixed-point data types using the following Simulink Fixed Point functions: sfix, ufix, sint, uint, sfrac, and ufrac. This parameter is only visible when you select User-defined for the Codebook and output data type parameter.

References

Gersho, A. and R. Gray. Vector Quantization and Signal Compression. Boston: Kluwer Academic Publishers, 1992.

Supported Data Types

Port	Supported Data Types
I	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
C	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers
Q(U)	• Double-precision ßoating point • Single-precision ßoating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

For more information on what data types are supported for each quantizer mode, see “Data Type Support” on page 2-1116.

See Also

Quantizer	Simulink
Scalar Quantizer Design	Signal Processing Blockset
Scalar Quantizer Encoder	Signal Processing Blockset
Uniform Encoder	Signal Processing Blockset
Uniform Decoder	Signal Processing Blockset

Scalar Quantizer Design

Purpose

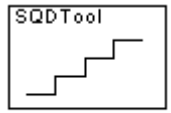
Start Scalar Quantizer Design Tool (SQDTool) to design scalar quantizer using Lloyd algorithm

Library

Quantizers

dspquant2

Description



Scalar Quantizer Design

Double-click on the Scalar Quantiz er Design block to start SQDTool, a GUI that allows you to design and implement a scalar quantizer. Based on your input values, SQDTool iteratively calculates the codebook values that minimize the mean squ ared error until the stopping criteria for the design process is satis Þed. The block uses the resulting quantizer codebook values and bo undary points to implement your scalar quantizer encoder and/or decoder.

For the Training Set parameter, enter a set of observations, or samples, of the signal you want to quantize. This data can be any variable de Þned in the MATLAB workspace in cluding a variable created using a MATLAB function, such as the default value randn(10000,1) .

You have two choices for the Source of initial codebook parameter. Select Auto-generate to have the block choose the values of the initial codebook vector. In this case, the minimum training set value becomes the Þrst codeword, and the maximum t raining set value becomes the last codeword. Then, the remaining initial codewords are equally spaced between these two values to form a codebook vector of length N, where N is the Number of levels parameter. When you select User defined , enter the initial codebook values in the Initial codebook Þeld. Then, set the Source of initial boundary points parameter. You can select Mid-points to locate the boundary points at the midpoint between the codewords. To calculate the mid-points, the block internally arranges the initial codebook values in asce nding order. You can also choose User defined and enter your own boundary points in the Initial boundary points (unbounded) Þeld. Only one boundary point can be located between two codewords. When you select User defined for the Source of initial boundary points parameter, the values you enter in the Initial codebook and Initial boundary points (unbounded) Þelds must be arranged in ascending order.

Note This block assumes that you are designing an unbounded quantizer. Therefore, the first and last boundary points are always -inf and inf regardless of any other boundary point values you might enter.

After you have specified the quantization parameters, the block performs an iterative process to design the optimal scalar quantizer. Each step of the design process involves using the Lloyd algorithm to calculate codebook values and quantizer boundary points. Then, the block calculates the squared quantization error and checks whether the stopping criteria has been satisfied.

The two possible options for the Stopping criteria parameter are Relative threshold and Maximum iteration. When you want the design process to stop when the fractional drop in the squared quantization error is below a certain value, select Relative threshold. Then, for Relative threshold, type the maximum acceptable fractional drop. When you want the design process to stop after a certain number of iterations, choose Maximum iteration. Then, enter the maximum number of iterations you want the block to perform in the Maximum iteration field. For Stopping criteria, you can also choose Whichever comes first and enter a Relative threshold and Maximum iteration value. The block stops iterating as soon as one of these conditions is satisfied.

With each iteration, the block quantizes the training set values based on the newly calculated codebook values and boundary points. When the training point lies on a boundary point, the algorithm uses the Tie-breaking rules parameter to determine which region the value is associated with. When you want the training point to be assigned to the lower indexed region, select Lower indexed codeword. To assign the training point with the higher indexed region, select Higher indexed codeword.

The Searching methods parameter determines how the block compares the training points to the boundary points. Select Linear search and SQDTool compares each training point to each quantization

region sequentially. This process continues until all the training points are associated with the appropriate regions.

Select Binary search for the Searching methods parameter and the block compares the training point to the middle value of the boundary points vector. When the training point is larger than this boundary point, the block discards the lower boundary points. The block then compares the training point to the middle boundary point of the new range, defined by the remaining boundary points. This process continues until the training point is associated with the appropriate region.

Click Design and Plot to design the quantizer with the parameter values specified on the left side of the GUI. The performance curve and the staircase character of the quantizer are updated and displayed in the figures on the right side of the GUI.

Note You must click Design and Plot to apply any changes you make to the parameter values in the SQDTool dialog box.

SQDTool can export parameter values that correspond to the figures displayed in the GUI. Click the Export Outputs button, or press Ctrl+E, to export the Final Codebook, Final Boundary Points, and Error values to the workspace, a text file, or a MAT-file. The Error values represent the mean squared error for each iteration.

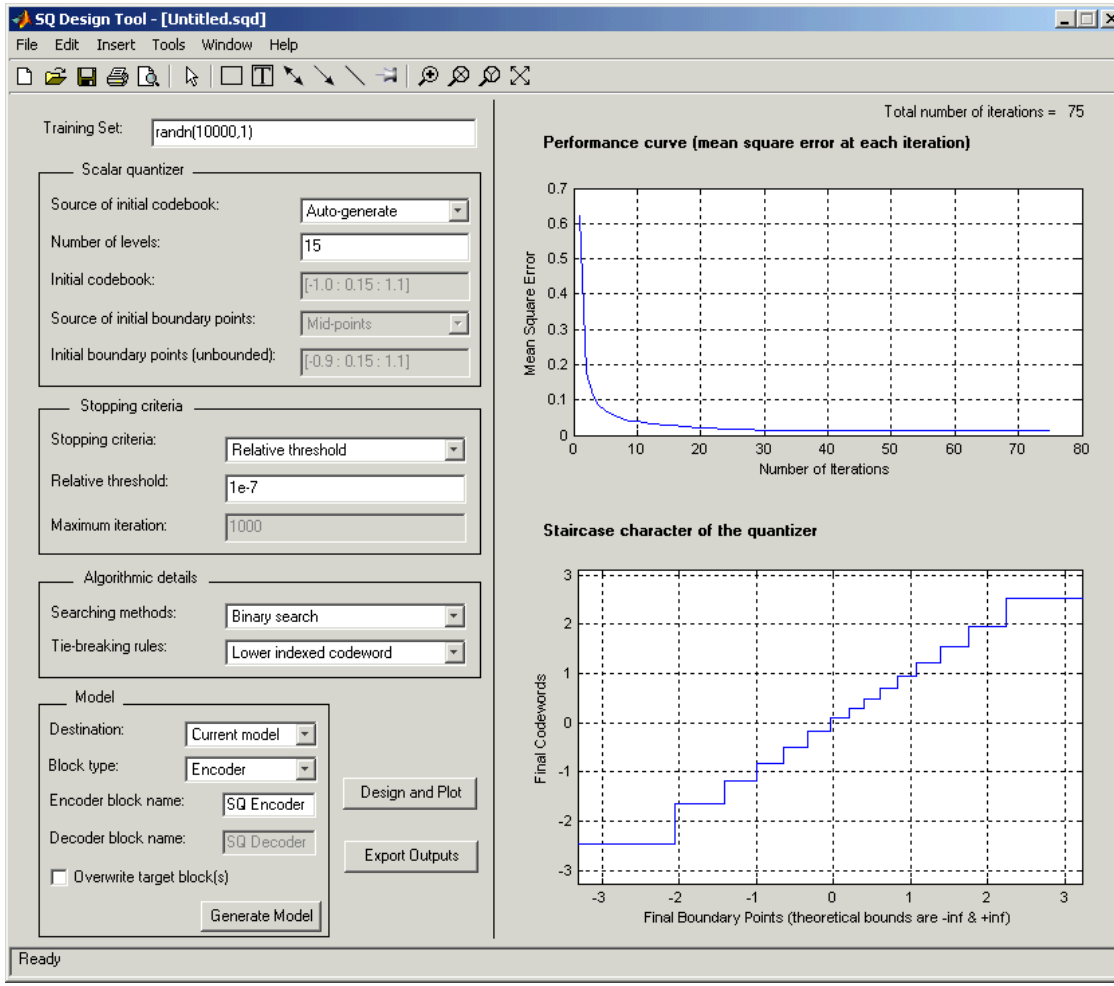
In the Model section of the GUI, specify the destination of the block that will contain the parameters of your quantizer. For Destination, select Current model to create a block with your parameters in the model you most recently selected. Type gcs in the MATLAB Command Window to display the name of your current model. Select New model to create a block in a new model file.

From the Block type list, select Encoder to design a Scalar Quantizer Encoder block. Select Decoder to design a Scalar Quantizer Decoder block. Select Both to design a Scalar Quantizer Encoder block and a Scalar Quantizer Decoder block.

In the Encoder block name field, enter a name for the Scalar Quantizer Encoder block. In the Decoder block name field, enter a name for the Scalar Quantizer Decoder block. When you have a Scalar Quantizer Encoder and/or Decoder block in your destination model with the same name, select the Overwrite target block(s) check box to replace the block's parameters with the current parameters. When you do not select this check box, a new Scalar Quantizer Encoder and/or Decoder block is created in your destination model.

Click Generate Model . SQDTool uses the parameters that correspond to the current plots to set the parameters of the Scalar Quantizer Encoder and/or Decoder blocks.

Dialog Box


SQ Design Tool - [Untitled.sqd]
File Edit Insert Tools Window Help
Training Set: randn(10000,1)
Scalar quantizer
Source of initial codebook: Auto-generate
Number of levels: 15
Initial codebook: [-1.0 : 0.15 : 1.1]
Source of initial boundary points: Mid-points
Initial boundary points (unbounded): [-0.9 : 0.15 : 1.1]
Stopping criteria
Stopping criteria: Relative threshold
Relative threshold: 1e-7
Maximum iteration: 1000
Algorithmic details
Searching methods: Binary search
Tie-breaking rules: Lower indexed codeword
Model
Destination: Current model
Block type: Encoder
Encoder block name: SQ Encoder
Decoder block name: SQ Decoder
Overwrite target block(s)
Generate Model
Design and Plot
Export Outputs
Total number of iterations = 75
Performance curve (mean square error at each iteration)
Mean Square Error
Number of Iterations
Staircase character of the quantizer
Final Codewords
Final Boundary Points (theoretical bounds are -inf & +inf)
Ready

Training Set

Enter the samples of the signal you would like to quantize. This data set can be a MATLAB function or a variable defined in the MATLAB workspace. The typical length of this data vector is `1e6`.

Source of initial codebook

Select Auto-generate to have the block choose the initial codebook values. Select User defined to enter your own initial codebook values.

Number of levels

Enter the length of the codebook vector. For a b-bit quantizer, the length should be $N = 2^b$.

Initial codebook

Enter your initial codebook values. From the Source of initial codebook list, select User defined in order to activate this parameter.

Source of initial boundary points

Select Mid-points to locate the boundary points at the midpoint between the codebook values. Choose User defined to enter your own boundary points. From the Source of initial codebook list, select User defined in order to activate this parameter.

Initial boundary points (unbounded)

Enter your initial boundary points. This block assumes that you are designing an unbounded quantizer. Therefore, the first and last boundary point are `-inf` and `inf`, regardless of any other boundary point values you might enter. From the Source of initial boundary points list, select User defined in order to activate this parameter.

Stopping criteria

Choose Relative threshold to enter the maximum acceptable fractional drop in the squared quantization error. Choose Maximum iteration to specify the number of iterations at which to stop. Choose Whichever comes first and the block stops the iteration process as soon as the relative threshold or maximum iteration value is attained.

Relative threshold

Type the value that is the maximum acceptable fractional drop in the squared quantization error.

Maximum iteration

Enter the maximum number of iterations you want the block to perform. From the Stopping criteria list, select Maximum iteration in order to activate this parameter.

Searching methods

Choose Linear search to use a linear search method when comparing the training points to the boundary points. Choose Binary search to use a binary search method when comparing the training points to the boundary points.

Tie-breaking rules

When a training point lies on a boundary point, choose Lower indexed codeword to assign the training point to the lower indexed quantization region. Choose Higher indexed codeword to assign the training point to the higher indexed region.

Design and Plot

Click this button to display the performance curve and the staircase character of the quantizer in the Þgures on the right side of the GUI. These plots are based on the current parameter settings.

You must click Design and Plot to apply any changes you make to the parameter values in the SQDTool GUI.

Export Outputs

Click this button, or press Ctrl+E, to export the Final Codebook, Final Boundary Points, and Error values to the workspace, a text Þle, or a MAT-Þle.

Destination

Choose Current model to create a Scalar Quantizer block in the model you most recently selected. Type gcs in the MATLAB Command Window to display the name of your current model. Choose New model to create a block in a new model Þle.

Block type
: Select Encoder to design a Scalar Quantizer Encoder block. Select Decoder to design a Scalar Quantizer Decoder block. Select Both to design a Scalar Quantizer Encoder block and a Scalar Quantizer Decoder block.

Encoder block name
: Enter a name for the Scalar Quantizer Encoder block.

Decoder block name
: Enter a name for the Scalar Quantizer Decoder block.

Overwrite target block(s)
: When you do not select this check box and a Scalar Quantizer Encoder and/or Decoder block with the same block name exists in the destination model, a new Scalar Quantizer Encoder and/or Decoder block is created in the destination model. When you select this check box and a Scalar Quantizer Encoder and/or Decoder block with the same block name exists in the destination model, the parameters of these blocks are overwritten by new parameters.

Generate Model
: Click this button and SQDTool uses the parameters that correspond to the current plots to set the parameters of the Scalar Quantizer Encoder and/or Decoder blocks.

References

Gersho, A. and R. Gray. Vector Quantization and Signal Compression. Boston: Kluwer Academic Publishers, 1992.

Supported Data Types

- Double-precision floating point

See Also

Quantizer

Scalar Quantizer Decoder

Simulink

Signal Processing Blockset

Scalar Quantizer Encoder	Signal Processing Blockset
Uniform Encoder	Signal Processing Blockset
Uniform Decoder	Signal Processing Blockset

Purpose

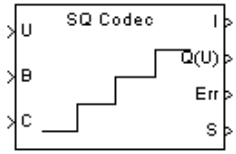
Encode each input value by associating it with index value of quantization region

Library

Quantizers

dspquant2

Description



Scalar Quantizer Encoder

The Scalar Quantizer Encoder block maps each input value to a quantization region by comparing the input value to the quantizer boundary points defined in the Boundary points parameter. The block outputs the zero-based index of the associated region.

You can select how you want to enter the Boundary points using the Source of quantizer parameters. When you select Specify via dialog, type the boundary points into the block parameters dialog box. When you select Input port, port B appears on the block. The block uses the input to port B as the Boundary points parameter.

Use the Boundary points parameter to specify the boundary points for your quantizer. These values are used to break up the set of input numbers into regions. Each region is specified by an index number.

Let N be the number of quantization regions. When the codebook is defined as [c1 c2 c3 ... cN], and the Boundary points parameter is defined as [p0 p1 p2 p3 ... pN], then p0<c1<p1<c2 ... p(N-1)<cN<pN for a regular quantizer. When your quantizer is bounded, from the Partitioning list, select Bounded. You need to specify N+1 boundary points, or [p0 p1 p2 p3 ... pN]. When your quantizer is unbounded, from the Partitioning list, select Unbounded. You need to specify N-1 boundary points, or [p1 p2 p3 ... p(N-1)]; the block sets p0 equal to -inf and pN equal to inf.

The block uses the Partitioning parameter to interpret the boundary points you enter. For instance, to create a bounded quantizer, from the Partitioning list, select Bounded and enter the following boundary points:

```
[0 0.5 3.7 5.8 6.0 11]
```

The block assigns any input values between 0 and 0.5 to index 0, input values between 0.5 and 3.7 to index 1, and so on. The block assigns any values that are less than 0 to index 0, the lowest index value. The block assigns any values that are greater than 11 to index 4, the highest index value.

To create an unbounded quantizer, from the Partitioning list, select Unbounded and enter the following boundary points:

```
[0  0.5  3.7  5.8  6.0  11]
```

The block assigns any input values between 0 and 0.5 to index 1, input values between 0.5 and 3.7 to index 2, and so on. The block assigns any input values less than 0 to index 0 and any values greater than 11 to index 6.

The Searching method parameter determines how the appropriate quantizer index is found. When you select Linear, the Scalar Quantizer Encoder block compares the input value to the first region defined by the first two boundary points. When the input value does not fall within this region, the block then compares the input value to the next region. This process continues until the input value is determined to be within a region and is associated with the appropriate index value. The computational cost of this process is of the order P, where P is the number of boundary points.

When you select Binary for the Searching method, the block compares the input value to the middle value of the boundary points vector. When the input value is larger than this boundary point, the block discards the boundary points that are lower than this middle value. The block then compares the input value to the middle boundary point of the new range, defined by the remaining boundary points. This process continues until the input value is associated with the appropriate index value. The computational cost of this process is of the order $\log_2 P$, where P is the number of boundary points. In most cases, the Binary option is faster than the Linear option.

When an input value is the same as a boundary point, the Tie-breaking rule parameter determines the region to which the value is assigned.

When you want the input value to be assigned to the lower indexed region, select Choose the lower index. To assign the input value with the higher indexed region, select Choose the higher index.

Select the Output codeword check box to output the codeword values that correspond to each index value at port Q(U).

Select the Output the quantization error check box to output the quantization error for each input value from the Err port on this block. The quantization error is the difference between the input value and the quantized output value.

When you select either the Output codeword check box or the Output quantization error check box, you must also enter your codebook values. If, from the Source of quantizer parameters list, you choose Specify via dialog, use the Codebook parameter to enter a vector of quantized output values that correspond to each region. If, from the Source of quantizer parameters list, you choose Input port, use input port C to specify your codebook values.

If, for the Partitioning parameter, you select Bounded, the Output clipping status check box and the Action for out of range input parameter appear. When you select the Output clipping status check box, port S appears on the block. Any time an input value is outside the range defined by the Boundary points parameter, the block outputs a 1 at the S port. When the value is inside the range, the blocks outputs a 0.

You can use the Action for out of range input parameter to determine the block's behavior when an input value is outside the range defined by the Boundary points parameter. Suppose the boundary points for a bounded quantizer are defined as [p0 p1 p2 p3 ... pN] and the possible index values are defined as [i0 i1 i2 ... i(N-1)], where i0=0 and i0<i1<i2<...<i(N-1). When you want any input value less than p0 to be assigned to index value i0 and any input values greater than pN to be assigned to index value i(N-1), select Clip. When you want to be warned when clipping occurs, select Clip and warn. When you want the simulation to stop and the block to display an error when the index values are out of range, select Error.

The Scalar Quantizer Encoder block accepts real floating-point and fixed-point inputs. For more information on the data types accepted by each port, see “Data Type Support” on page 2-1134 or “Supported Data Types” on page 2-1139.

Data Type Support

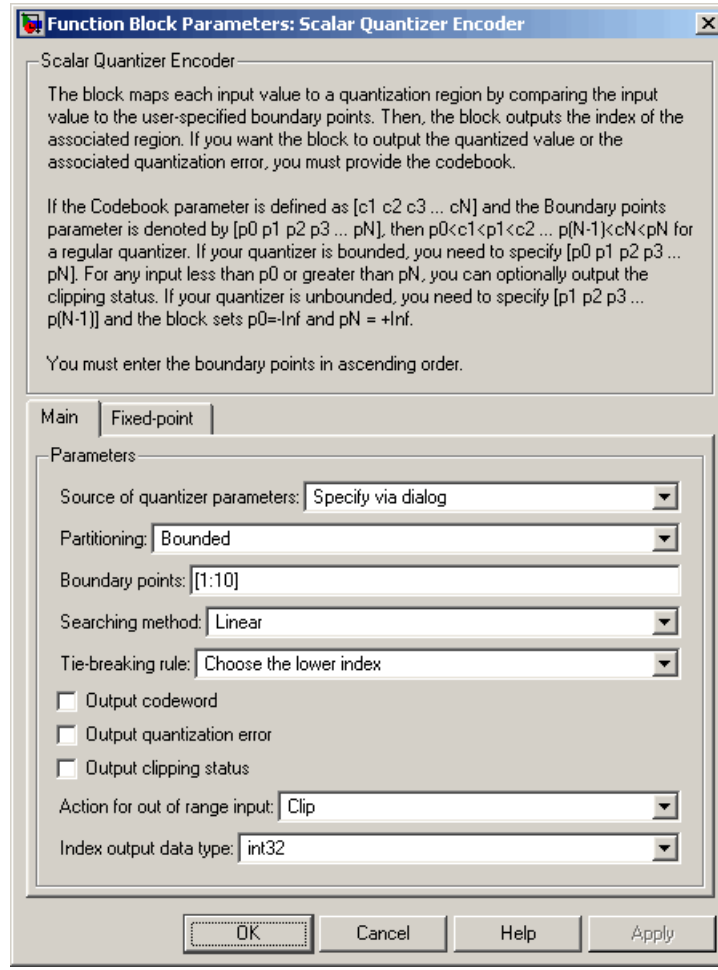
The input data values, boundary points, and codebook values can be input to the block at ports U, B, and C, respectively. The data type of the inputs can be double, single, or Fixed-point.

The outputs of the block can be the index values, the quantized output values, the quantization error, and the clipping status. Use the Index output data type parameter to specify the data type of the index output from the block at port I. You can choose int8, uint8, int16, uint16, int32, or uint32. The data type of the quantized output and the quantization error can be double, single, or Fixed-point. The clipping status values output at port S are Boolean values.

Note The input data, boundary points, codebook values, quantized output values, and the quantization error must have the same data type whenever they are present.

Dialog Box

The Main pane of the Scalar Quantizer Encoder block dialog appears as follows.



Source of quantizer parameters

Choose Specify via dialog to enter the boundary points and codebook values using the block parameters dialog box. Select

Input port to specify the parameters using the block's input ports. Input the boundary points and codebook values using ports B and C, respectively.

Partitioning

When your quantizer is bounded, select Bounded. When your quantizer is unbounded, select Unbounded.

Boundary points

Enter a vector of values that represent the boundary points of the quantizer regions. This parameter is visible when you select Specify via dialog from the Source of quantizer parameters list. Tunable.

Searching method

When you select Linear, the block finds the region in which the input value is located using a linear search. When you select Binary, the block finds the region in which the input value is located using a binary search.

Tie-breaking rule

Set this parameter to determine the behavior of the block when the input value is the same as the boundary point. When you select Choose the lower index, the input value is assigned to lower indexed region. When you select Choose the higher index, the value is assigned to the higher indexed region.

Output codeword

Select this check box to output the codeword values that correspond to each index value at port Q(U).

Output quantization error

Select this check box to output the quantization error for each input value at port Err.

Codebook

Enter a vector of quantized output values that correspond to each index value. If, for the Partitioning parameter, you select Bounded and your boundary points vector has length N, then you must specify a codebook of length N-1. If, for the Partitioning

parameter, you select Unbounded and your boundary points vector has length N, then you must specify a codebook of length N+1.

This parameter is visible when you select Specify via dialog from the Source of quantizer parameters list and you select either the Output codeword or Output quantization error check box. Tunable.

Output clipping status

When you select this check box, port S appears on the block. Any time an input value is outside the range de Þned by the Boundary points parameter, the block outputs a 1 at this port. When the value is inside the range, the block outputs a 0. This parameter is visible when you select Bounded from the Partitioning list.

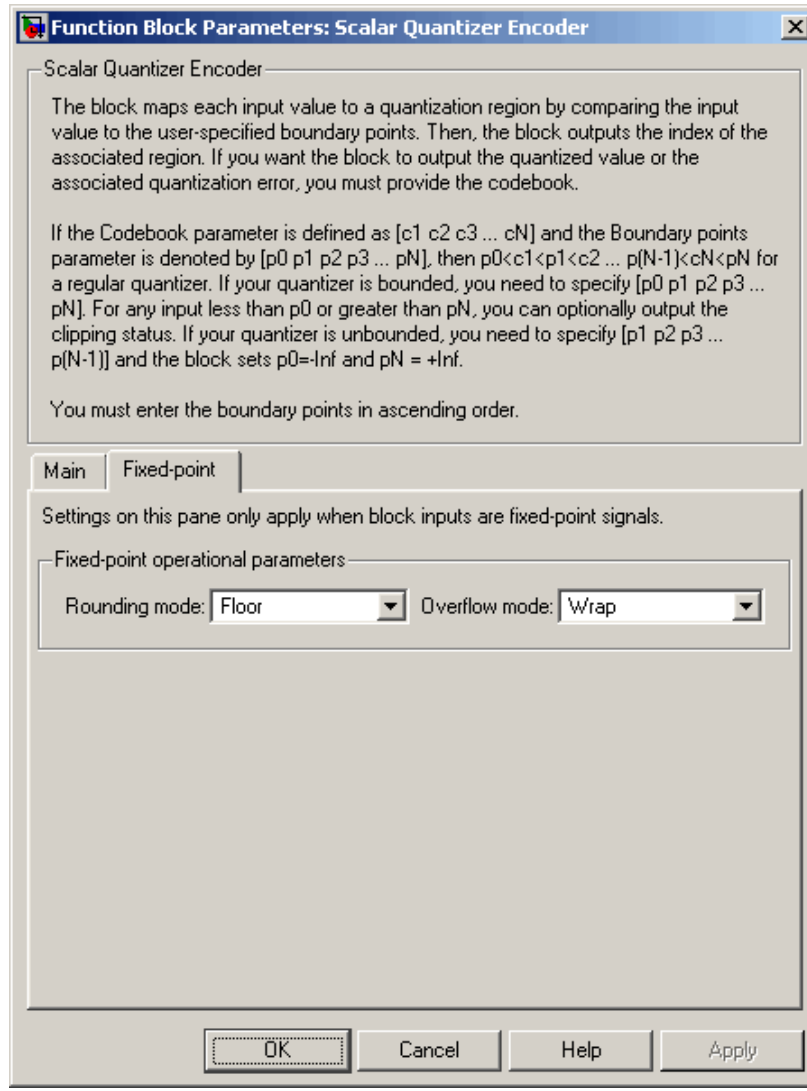
Action for out of range input

Use this parameter to determine the behavior of the block when an input value is outside the range de Þned by the Boundary points parameter. Suppose the boundary points are de Þned as [p0 p1 p2 p3 ... pN] and the index values are de Þned as [i0 i1 i2 ... i(N-1)] . When you want any input value less than p0 to be assigned to index value i0 and any input values greater than pN to be assigned to index value i(N-1) , select Clip . When you want to be warned when clipping occurs, select Clip and warn . When you want the simulation to stop and the block to display an error when the index v alues are out of range, select Error . This parameter is visible when you select Bounded from the Partitioning list.

Index output data type

Specify the data type of the index output from the block at port I. You can choose int8 , uint8 , int16 , uint16 , int32 , or uint32 .

The Fixed-point pane of the Scalar Quantizer Encoder block dialog appears as follows.



Rounding mode

Select the rounding mode for fixed-point operations.

Over flow mode

Select the overflow mode to be used when block inputs are fixed point.

References

Gersho, A. and R. Gray. Vector Quantization and Signal Compression. Boston: Kluwer Academic Publishers, 1992.

Supported Data Types

Port	Supported Data Types
U	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
B	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
C	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
I	• 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Q(U)	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

Port	Supported Data Types
Err	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
S	• Boolean

For more information on what data types are supported for each quantizer mode, see “Data Type Support” on page 2-1116.

See Also

Quantizer	Simulink
Scalar Quantizer Decoder	Signal Processing Blockset
Scalar Quantizer Design	Signal Processing Blockset
Uniform Encoder	Signal Processing Blockset
Uniform Decoder	Signal Processing Blockset

Purpose

Select input elements from vector, matrix, or multidimensional signal

Library

Signal Management / Indexing

dspindex

Description

The Selector block is an implementati on of the Simulink Selector block. See Selector for more information.

Short-Time FFT

Purpose

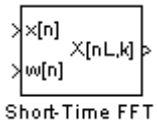
Compute nonparametric estimate of spectrum using short-time, fast Fourier transform (FFT) method

Library

Transforms

dspxfrm3

Description



The Short-Time FFT block computes a nonparametric estimate of the spectrum. The block buffers, applies a window, and zero pads the input signal. The block then takes the FFT of the signal, transforming it into the frequency domain.

Connect your sample-based or frame- based, single-channel analysis window to the w(n) port. For the Analysis window length parameter, enter the length of the analysis window, W. When your analysis window is a sample-based signal, the block buffers it into a frame-based signal with frame length W. When your analysis window is a frame-based signal and its frame length is not W, the block buffers the signal so that its frame length is W.

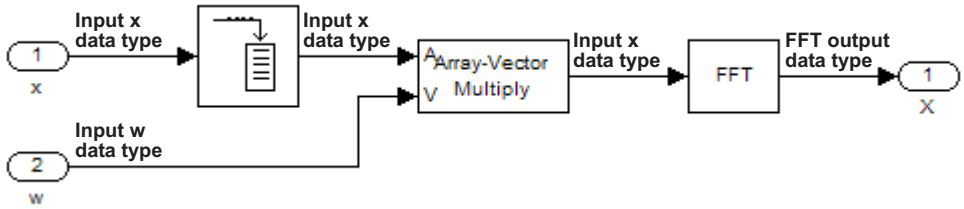
Connect your sample-based or frame-based, single-channel or multichannel input signal to the x(n) port. After the block buffers and windows this signal, it zero-pa ds the signal before computing the FFT. For the FFT length parameter, enter the length to which the block pads the input signal. For the Overlap between consecutive windows (in samples) parameter, enter the number of samples to overlap each frame of the input signal.

The complex-valued, sample-based, single-channel or multichannel short-time FFT is outp ut at port X(n,k).

The Short-Time FFT block supports real and complex ßoating-point and Þxed-point signals.

Fixed-Point Data Types

The following diagram shows the data types used within the Short-Time FFT subsystem block for Þxed-point signals.



The settings for the fixed-point parameters of the Array-Vector Multiply block in the diagram above are as follows:

- Rounding Mode — Floor
- Overflow Mode — Wrap
- Product output — Inherit via internal rule
- Accumulator — Inherit via internal rule
- Output — Same as first input

The settings for the fixed-point parameters of the FFT block in the diagram above are as follows:

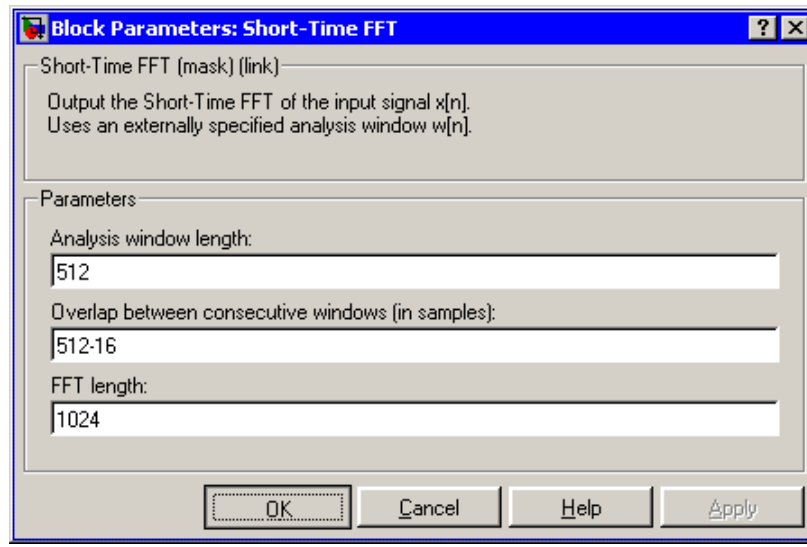
- Rounding Mode — Floor
- Overflow Mode — Wrap
- Sine table — Same word length as input
- Product output — Inherit via internal rule
- Accumulator — Inherit via internal rule
- Output — Inherit via internal rule

See the FFT and Array-Vector Multiply block reference pages for more information.

Examples

The dspstsa_win32 demo illustrates how to use the Short-Time FFT and Inverse Short-Time FFT block s to remove the background noise from a speech signal.

Dialog Box



Analysis window length
: Enter the frame length of the analysis window.

Overlap between consecutive windows (in samples)
: Enter the number of samples of overlap for each frame of the input signal.

FFT length
: Enter the length to which the block pads the input signal.

References

Quatieri, Thomas E. Discrete-Time Speech Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 2001.

Supported Data Types

Port	Supported Data Types
x(n)	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
w(n)	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers
X(n,k)	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed only) • 8-, 16-, and 32-bit signed integers

See Also

Burg Method	Signal Processing Blockset
Inverse Short-Time FFT	Signal Processing Blockset
Magnitude FFT	Signal Processing Blockset
Periodogram	Signal Processing Blockset
Spectrum Scope	Signal Processing Blockset
Window Function	Signal Processing Blockset
Yule-Walker Method	Signal Processing Blockset
pwelch	Signal Processing Toolbox

See “Power Spectrum Estimation” for related information.

Purpose

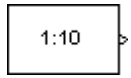
Import signal from MATLAB workspace

Library

Signal Processing Sources

dspsrcs4

Description



The Signal From Workspace block imports a signal from the MATLAB workspace into the Simulink model. The Signal parameter specifies the name of a MATLAB workspace variable containing the signal to import, or any valid MATLAB expression defining a matrix or 3-D array.

When the Signal parameter specifies an M-by-N matrix ($M \neq 1$), each of the N columns is treated as a distinct channel. You specify the frame size in the Samples per frame parameter, M_o, and the output is an M_o-by-N matrix containing M_o consecutive samples from each signal channel. You specify the output sample period in the Sample time parameter, T_s, and the output frame period is M_o*T_s. For $M_o=1$, the output is sample based; otherwise the output is frame based. For convenience, an imported row vector (M=1) is treated as a single channel, so the output dimension is M_o-by-1.

When the Signal parameter specifies an M-by-N-by-P array, each of the P pages (an M-by-N matrix) is output in sequence with period T_s. The Samples per frame parameter must be set to 1, and the output is always sample based.

Initial and Final Conditions

Unlike the Simulink From Workspace block, the Signal From Workspace block holds the output value constant between successive output frames (that is, no linear interpolation takes place). Additionally, the initial signal values are always produced immediately at t=0.

When the block has output all of the available signal samples, it can start again at the beginning of the signal, or simply repeat the final value or generate zeros until the end of the simulation. (The block does not extrapolate the imported signal beyond the last sample.) The Form output after final data value by parameter controls this behavior:

- When you specify Setting To Zero, the block generates zero-valued outputs for the duration of the simulation after generating the last frame of the signal.
- When you specify Holding Final Value, the block repeats the final sample for the duration of the simulation after generating the last frame of the signal.
- When you specify Cyclic Repetition, the block repeats the signal from the beginning after it reaches the last sample in the signal. If the frame size you specify in the Samples per frame parameter does not evenly divide the input length, a buffer block is inserted into the Signal From Workspace subsystem, and the model becomes multirate. If you do not want your model to become multirate, make sure the frame size evenly divides the input signal length.

  Select the Warn when frame size does not evenly divide input length parameter to be alerted when the input length is not an integer multiple of the frame size and your model will become multirate. Use the Model Explorer to turn these warnings on or off model-wide:

  1 Select Model Explorer from the View menu in your model.

  2 In the Search bar of the Model Explorer, search by Property Name for the ignoreOrWarnInputAndFrameLengths property. Each block with the Warn when frame size does not evenly divide input length check box appears in the list in the Contents pane.

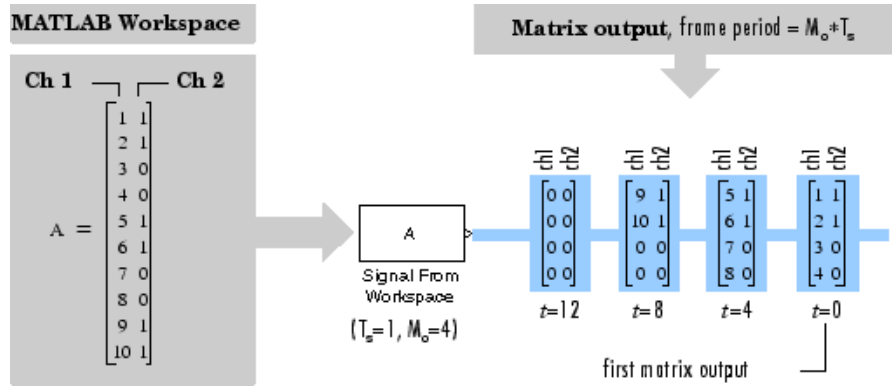
  3 Select each of the blocks for which you wish to toggle the warning parameter, and select or deselect the check box in the ignoreOrWarnInputAndFrameLengths column.

Examples

Example 1

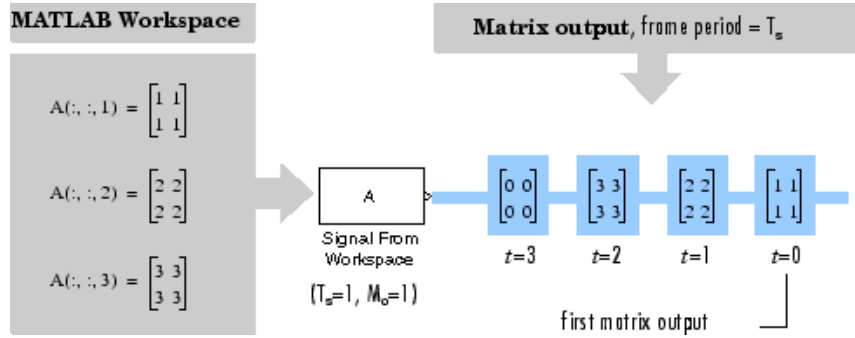
In the first model below, the Signal From Workspace imports a two-channel signal from the workspace matrix A. The Sample time is set to 1 and the Samples per frame is set to 4, so the output is frame based with a frame size of 4 and a frame period of 4 seconds. The Form

output after final data value by parameter specifies Setting To Zero, so all outputs after the third frame (at t=8) are zero.



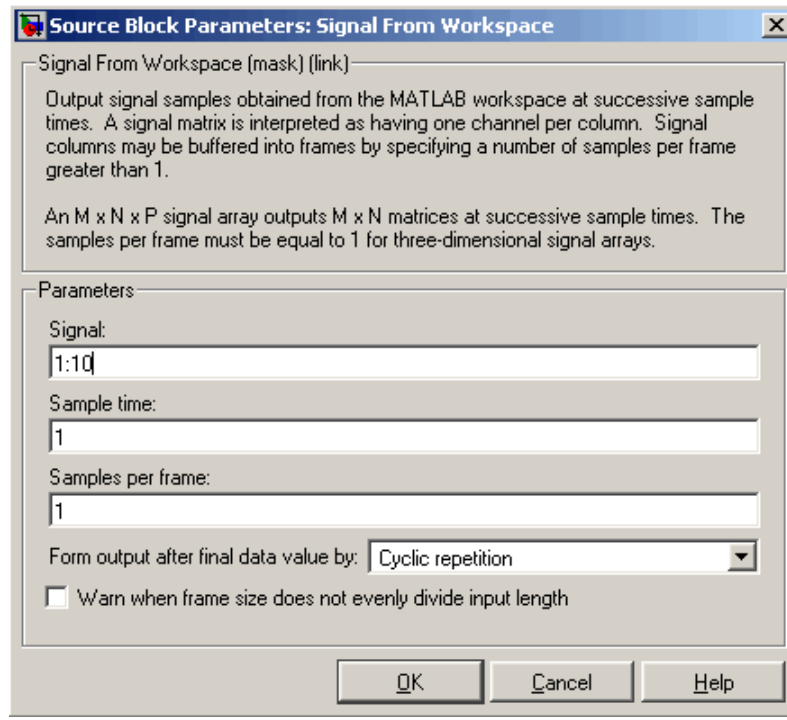
Example 2

In the second model below, the Signal From Workspace block imports a sample-based matrix signal from the 3-D workspace array A. Again, the Form output after final data value by parameter specifies Setting To Zero, so all outputs after the third (at t=2) are zero.



The Samples per frame parameter is set to 1 for 3-D input.

Dialog Box



Signal

The name of the MATLAB workspace variable from which to import the signal, or a valid MATLAB expression specifying the signal.

Sample time

The sample period, T_s, of the output. The output frame period is $M_o * T_s$.

Samples per frame

The number of samples, M_o, to buffer into each output frame. This value must be 1 when you specify a 3-D array in the Signal parameter.

Form output after final data value by
: Specifies the output after all of the specified signal samples have been generated. The block can output zeros for the duration of the simulation (Setting to zero), repeat the final data sample (Holding Final Value) or repeat the entire signal from the beginning (Cyclic Repetition).

Warn when frame size does not evenly divide input length
: Select this parameter to be alerted when the input length is not an integer multiple of the frame size and your model will become multirate. For more information, see “Initial and Final Conditions” on page 2-1147.

  This parameter is only visible when Cyclic Repetition is selected for the Form output after final data value by parameter.

Supported Data Types

- Double-precision floating point
- Single-precision floating point
- Fixed point (signed and unsigned)
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

See Also

From Audio Device	Signal Processing Blockset
From Wave File	Signal Processing Blockset
Signal From Workspace	Signal Processing Blockset
From Workspace	Simulink
To Workspace	Simulink
Triggered Signal From Workspace	Signal Processing Blockset

See the sections below for related information:

- “Creating Sample-Based Signals”
- “Creating Frame-Based Signals”
- “Importing and Exporting Sample-Based Signals”
- “Importing and Exporting Frame-Based Signals”

Purpose

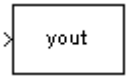
Write simulation data to array in MATLAB workspace

Library

Signal Processing Sinks

dspsnks4

Description



The Signal To Workspace block writes data from your simulation into an array in the MATLAB main workspace. The output array can be 2-D or 3-D, depending on whether t he data is 1-D, sample based, or frame based. The Signal To Workspace block and the Simulink To Workspace block can output the sa me arrays when their parameters are set appropriately.

For more information on the Signal To Workspace block, see the following sections of this reference page:

- “Parameter Descriptions” on page 2-1153
- “Output Dimension Summary” on page 2-1155
- “Matching the Outputs of Signal To Workspace and To Workspace Blocks” on page 2-1155
- “Examples” on page 2-1156

Parameter Descriptions

The Variable name parameter is the name of the array in the MATLAB workspace into which the block logs the simulation data. The array is created in the workspace only after the simulation stops running. When you enter the name of an existing workspace variable, the block overwrites the variable with an array of simulation data after the simulation stops running.

When the block input is sample based or 1-D, the Limit data points to last parameter indicates how many samples of data to save. When the block input is frame based, th is parameter indicates how many frames of data to save. When the simulation generates more than the speciÞed maximum number of samples or frames, the simulation saves only the most recently generated data. To capture all data, set Limit data points to last to inf .

The Decimation parameter is the decimation factor. It can be set to any positive integer d, and allows you to write data at every dth sample. The default decimation, 1, writes data at every time step.

The Frames parameter sets the dimension of the output array to 2-D or 3-D for frame-based inputs. The block ignores this parameter for 1-D and sample-based inputs. The Frames parameter has the following two settings:

- Log frames separately (3-D array) : Given an M-by-N frame-based input signal, the block outputs an M-by-N-by-K array, where K is the number of frames logged by the end of the simulation. (K is bounded above by the Limit data points to last parameter.) Each input frame is an element of the 3-D array. (See "Example 2: Frame-Based Inputs" on page 2-1157.)
- Concatenate frames (2-D array) : Given an M-by-N frame-based input signal with frame size f, the block outputs a (K* f)-by-N matrix, where K* f is the number of samples acquired by the end of the simulation. Each input frame is vertically concatenated to the previous frame to produce the 2-D array output. (See "Example 2: Frame-Based Inputs" on page 2-1157.)

Signal to Workspace always logs sample-based input data as 3-D arrays, regardless of the Frame parameter setting. Given an M-by-N sample-based signal, the block outputs an M-by-N-by-L array, where L is the number of samples logged by the end of the simulation (L is bounded above by the Limit data points to last parameter). Each sample-based matrix is an element of the 3-D array. (See "Example 1: Sample-Based Inputs" on page 2-1156.)

For 1-D vector inputs, the block outputs a 2-D matrix regardless of the setting of Frame . For a length-N 1-D vector input, the block outputs an L-by-N matrix. Each input vector is a row of the output matrix, vertically concatenated to the previous vector.

Output Dimension Summary

The following table summarizes the output array dimensions for various block inputs. In the table, f is the frame size of the input, K is the number of frames acquired by the end of the simulation, and L is the number of samples acquired by the end of the simulation (K and L are bounded above by the Limit data points to last parameter).

Input Signal Type	Signal To Workspace Output Dimension
Sample-based M-by-N matrix	M-by-N-by-L array
Length-N 1-D vector	L-by-N matrix
Frame-based M-by-N matrix; Frame set to Log frames separately (3-D array)	M-by-N-by-K array
Frame-based M-by-N matrix; Frame set to Concatenate frames (2-D array)	(K* f)-by-N matrix K*f is the number of samples acquired by the end of the simulation.

Matching the Outputs of Signal To Workspace and To Workspace Blocks

The To Workspace block in the Simulink Sinks Library and the Signal To Workspace block can output the same array when they are given the same inputs. To match the blocks' outputs, set their parameters as follows.

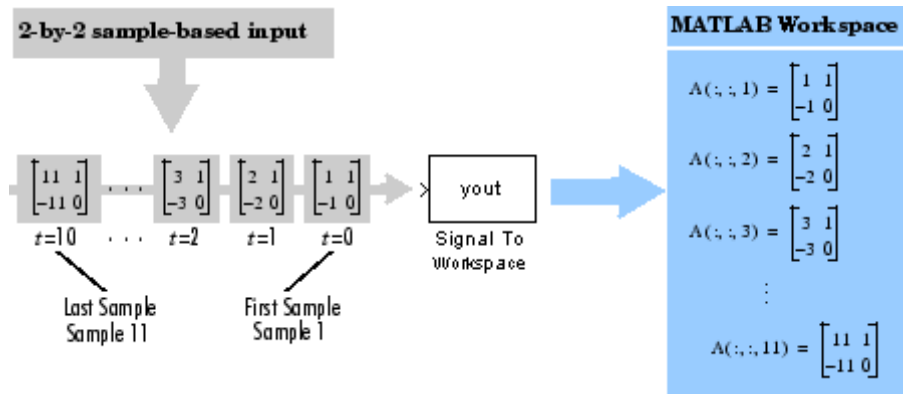
Block Parameters	Signal To Workspace	To Workspace
Limit data points to last	x (any positive integer or inf)	x
Decimation	y (any positive integer, not inf)	y

Block Parameters	Signal To Workspace	To Workspace
Sample Time	No such parameter	-1
Save format	No such parameter	Array
Frames	Concatenate frames (2-D array)	No such parameter

Examples

Example 1: Sample-Based Inputs

In the following model, the input to the Signal To Workspace block is a 2-by-2 sample-based matrix signal with a sample time of 1 (generated by a Signal From Workspace block). The Signal To Workspace block logs 11 samples by the end of the simulation, and creates a 2-by-2-by-11 array, A, in the MATLAB workspace.



The block settings are as follows.

Signal To Workspace Block Parameters	
Variable name	yout
Limit da ta points to last	inf
Decimation	1

Signal To Workspace Block Parameters	
Frames	Ignored since block input is not frame based
Con figuration Dialog Box Parameters	
Start time	0
Stop time	10
Signal From Workspace Parameters (provides Signal To Workspace input)	
Signal	input1 (deÞned below)
Sample time	1
Samples per frame	1
Form output after fi nal data value by	Setting to zero

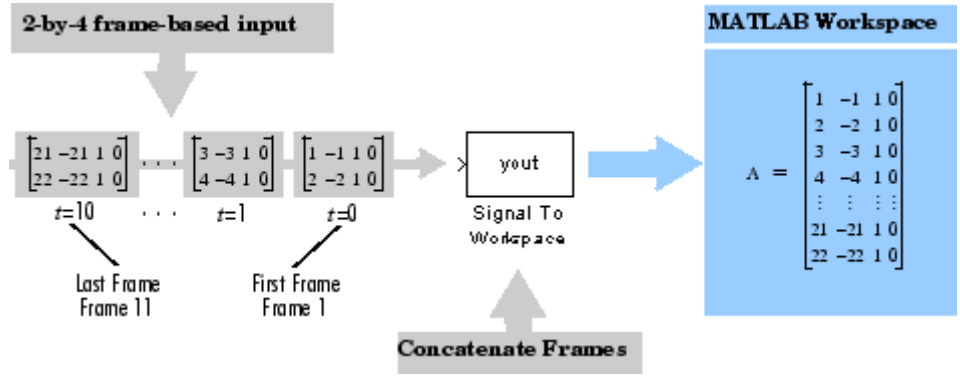
```
input1 = cat(3, [1 1; -1 0], [2 1; -2 0],...,[11 1; -11 0])
```

Example 2: Frame-Based Inputs

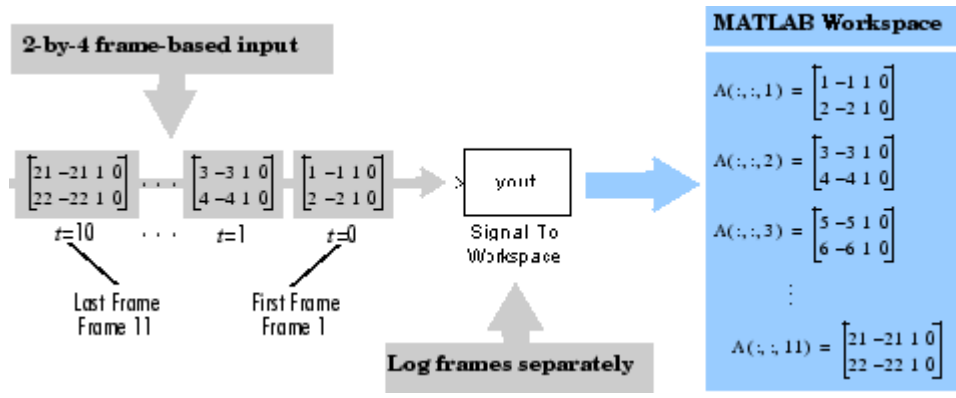
In the following model, the input to the Signal To Workspace block is a 2-by-4 frame-based matrix signal w ith a frame period of 1 (generated by a Signal From Workspace block). The block logs 11 frames (two samples per frame) by the end of the simulation. The frames are concatenated to create a 22-by-4 matrix, A, in the MATLAB workspace.

The block settings for the following model are similar to the settings used in Example 1, except Frames is set to Concatenate frames (2-D array) and the Signal From Workspace parameter, Signal , is set to input2 , where

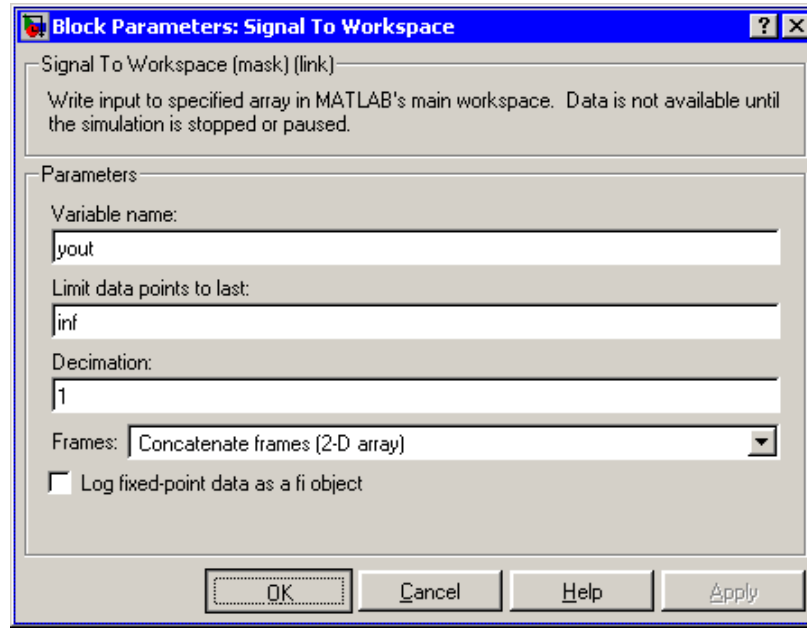
```
input2 = [1 -1 1 0; 2 -2 1 0; 3 -3 1 0;...; 22 -22 1 0]
```



In the 2-D output, there is no indication of where one frame ends and another begins. By setting Frames to Log frames separately (3-D array) in this model, you can easily see each frame in the MATLAB workspace, as illustrated in the following model. Each of the 11 frames is logged separately to create a 2-by -4-by-11 array, A, in the MATLAB workspace.



Dialog Box



Variable name

The name of the array that holds the input data.

Limit data points to last

The maximum number of input samples (for sample-based inputs) or input frames (for frame-based inputs) to be saved.

Decimation

The decimation factor, d. Data is written at every dth sample.

Frames

The output dimensionality for frame-based inputs. Frames can be set to Concatenate frames (2-D array) or Log frames separately (3-D array). This parameter is ignored when inputs are not frame based.

Log **fi**xed-point data as a **fi** object

Select to log Þxed-point data to the MATLAB workspace as a Fixed-Point Toolbox fi object. Otherwise, Þxed-point data is logged to the workspace as double .

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Fixed point (signed and unsigned)
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

See Also

Triggered To Workspace

To Workspace

Signal Processing Blockset

Simulink

Purpose

Generate continuous or discrete sine wave

Library

Signal Processing Sources

dspsrcs4

Description

The Sine Wave block generates a multichannel real or complex sinusoidal signal, with independen t amplitude, frequency, and phase in each output channel. A real sinusoidal signal is generated when the Output complexity parameter is set to Real , and is de Þned by an expression of the type

$$y = A\sin(2\pi ft + \phi)$$

where you specify A in the Amplitude parameter, f in hertz in the Frequency parameter, and φ in radians in the Phase offset parameter. A complex exponential signal is generated when the Output complexity parameter is set to Complex , and is de Þned by an expression of the type

$$y = Ae^{j(2\pi ft+\phi)} = A\{\cos(2\pi ft + \phi) + j\sin(2\pi ft + \phi)\}$$

Sections of This Reference Page

- “Generating Multichannel Outputs” on page 2-1162
- “Output Sample Time and Samples Per Frame” on page 2-1162
- “Sample Mode” on page 2-1162
- “Discrete Computational Methods” on page 2-1163
- “Examples” on page 2-1166
- “Dialog Box” on page 2-1167
- “Supported Data Types” on page 2-1172
- “See Also” on page 2-1172

Generating Multichannel Outputs

For both real and complex sinusoids, the Amplitude, Frequency, and Phase offset parameter values (A, f, and φ) can be scalars or length-N vectors, where N is the desired number of channels in the output. When you specify at least one of these parameters as a length-N vector, scalar values specified for the other parameters are applied to every channel.

For example, to generate the three-channel output containing the real sinusoids below, set Output complexity to Real and the other parameters as follows:

- Amplitude = [1 2 3]
- Frequency = [1000 500 250]
- Phase offset = [0 0 pi/2]

$$y = \begin{cases} \sin(2000\pi t) & (\text{channel } 1) \\ 2\sin(1000\pi t) & (\text{channel } 2) \\ 3\sin\left(500\pi t + \dfrac{\pi}{2}\right) & (\text{channel } 3) \end{cases}$$

Output Sample Time and Samples Per Frame

In all discrete modes, the block buffers the sampled sinusoids into frames of size M, where you specify M in the Samples per frame parameter. The output is a frame-based M-by-N matrix with frame period M*T_s, where you specify T_s in the Sample time parameter. For M=1, the output is sample based.

Sample Mode

The Sample mode parameter specifies the block's sampling property, which can be Continuous or Discrete:

- Continuous

In continuous mode, the sinusoid in the ith channel, y_i, is computed as a continuous function,

$$y_i = A_i \sin\left(2\pi f_i t + \phi_i\right) \quad \text{(real)}$$

or

$$y_i = A_i e^{j(2\pi f_i t + \phi_i)} \qquad \text{(complex)}$$

and the block's output is continuous. In this mode, the block's operation is the same as that of a Simulink Sine Wave block with Sample time set to 0. This mode offers high accuracy, but requires trigonometric function evaluations at each simulation step, which is computationally expensive. Additionally, because this method tracks absolute simulation time, a discontinuity will eventually occur when the time value reaches its maximum limit.

Note also that many Signal Processing Blockset blocks do not accept continuous-time inputs.

- Discrete

  In discrete mode, the block's discrete-time output can be generated by directly evaluating the trigonometric function, by table lookup, or by a differential method. The three options are explained below.

Discrete Computational Methods

When you select Discrete from the Sample mode parameter, the secondary Computation method parameter provides three options for generating the discrete sinusoid:

- Trigonometric Fcn

- Table Lookup
- Differential

Trigonometric Fcn

The trigonometric function method computes the sinusoid in the ith channel, y_i, by sampling the continuous function

$$y_i = A_i \sin(2\pi f_i t + \phi_i) \quad \text{(real)}$$

or

$$y_i = A_i e^{j(2\pi f_i t + \phi_i)} \quad \text{(complex)}$$

with a period of T_s, where you specify T_s in the Sample time parameter. This mode of operation shares the same benefits and liabilities as the Continuous sample mode described above.

At each sample time, the block evaluates the sine function at the appropriate time value within the first cycle of the sinusoid. By constraining trigonometric evaluations to the first cycle of each sinusoid, the block avoids the imprecision of computing the sine of very large numbers, and eliminates the possibility of discontinuity during extended operations (when an absolute time variable might overflow). This method therefore avoids the memory demands of the table lookup method at the expense of many more floating-point operations.

Table Lookup

The table lookup method precomputes the unique samples of every output sinusoid at the start of the simulation, and recalls the samples from memory as needed. Because a table of finite length can only be constructed when all output sequences repeat, the method requires that

the period of every sinusoid in the output be evenly divisible by the sample period. That is, $1/(f_i T_s) = k_i$ must be an integer value for every channel $i = 1, 2, ..., N$. When the Optimize table for parameter is set to Speed, the table constructed for each channel contains k_i elements. When the Optimize table for parameter is set to Memory, the table constructed for each channel contains $k_i/4$ elements.

For long output sequences, the table lookup method requires far fewer ßoating-point operations than any of the other methods, but can demand considerably more mem ory, especially for high sample rates (long tables). This is the recommended method for models that are intended to emulate or genera te code for DSP hardware, and that therefore need to be optimized for execution speed.

Differential

The differential method uses an incre mental algorithm. This algorithm computes the output samples based on the output values computed at the previous sample time (and pre computed update terms) by making use of the following identities.

$$\sin(t+T_s) = \sin(t)\cos(T_s) + \cos(t)\sin(T_s)$$
$$\cos(t+T_s) = \cos(t)\cos(T_s) - \sin(t)\sin(T_s)$$

The update equations for the sinusoid in the ith channel, y_i, can therefore be written in matrix form as

$$\begin{bmatrix} \sin\{2\pi f_i (t+T_s)+\phi_i\} \\ \cos\{2\pi f_i (t+T_s)+\phi_i\} \end{bmatrix} = \begin{bmatrix} \cos(2\pi f_i T_s) & \sin(2\pi f_i T_s) \\ -\sin(2\pi f_i T_s) & \cos(2\pi f_i T_s) \end{bmatrix} \begin{bmatrix} \sin(2\pi f_i t+\phi_i) \\ \cos(2\pi f_i t+\phi_i) \end{bmatrix}$$

where you specify T_s in the Sample time parameter. Since T_s is constant, the right-hand matrix is a constant and can be computed once at the start of the simulation. The value of $A_i \sin[2\pi f_i(t+T_s)+\varphi_i]$ is then computed from the values of $\sin(2\pi f_i t+\varphi_i)$ and $\cos(2\pi f_i t+\varphi_i)$ by a simple matrix multiplication at each time step.

This mode offers reduced computati onal load, but is subject to drift over time due to cumulative quant ization error. Because the method

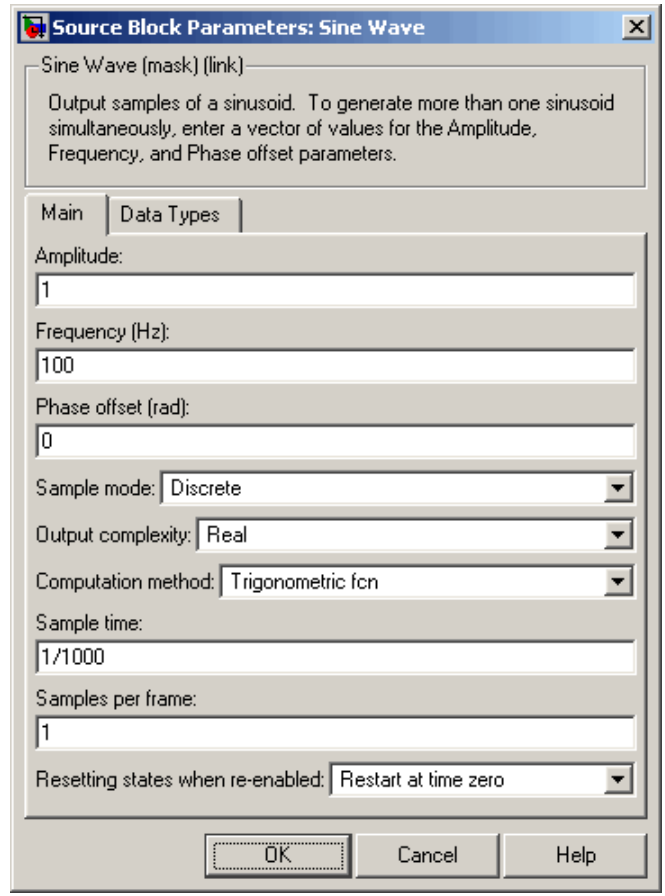
is not contingent on an absolute time value, there is no danger of discontinuity during extended operations (when an absolute time variable might over ßow).

Examples

The dspsinecomp demo provides a comparison of all the available sine generation methods.

Dialog Box

The Main pane of the Sine Wave block dialog appears as follows.



Opening this dialog box causes a running simulation to pause. See "Changing Source Block Parameters During Simulation" in the online Simulink documentation for details.

Amplitude

A length- N vector containing the amplitudes of the sine waves in each of N output channels, or a scalar to be applied to all N channels. The vector length must be the same as that speci Þed for the Frequency and Phase offset parameters. Tunable when Computation method is to Trigonometric fcn or Differential .

Frequency

A length- N vector containing frequencies, in rad/s, of the sine waves in each of N output channels, or a scalar to be applied to all N channels. The vector length must be the same as that speciÞed for the Amplitude and Phase offset parameters. You can specify positive, zero, or negative frequencies. Tunable when Sample mode is Continuous or Computation method is Trigonometric fcn .

Phase offset

A length- N vector containing the phase offsets, in radians, of the sine waves in each of N output channels, or a scalar to be applied to all N channels. The vector length must be the same as that speci Þed for the Amplitude and Frequency parameters. Tunable when Sample mode is Continuous or Computation method is Trigonometric fcn .

Sample mode

The block's sampling behavior, Continuous or Discrete . This parameter is not tunable.

Output complexity

The type of waveform to generate: Real speciÞes a real sine wave, Complex speciÞes a complex exponential. This parameter is not tunable.

Computation method

The method by which discrete-time sinusoids are generated: Trigonometric fcn , Table lookup , or Differential . This parameter is not tunable. This parameter is disabled when

you select Continuous from the Sample mode parameter. For details, see “Discrete Computational Methods” on page 2-1163.

Optimize table for

Optimizes the table of sine values for Speed or Memory (this parameter is only visible when the Computation method parameter is set to Table lookup). When optimized for speed, the table contains k elements, and when optimized for memory, the table contains k/4 elements, where k is the number of input samples in one full period of the sine wave.

Sample time

The period with which the sine wave is sampled, T_s. The block's output frame period is $M*T_s$, where you specify M in the Samples per frame parameter. This parameter is disabled when you select Continuous from the Sample mode parameter. This parameter is not tunable.

Samples per frame

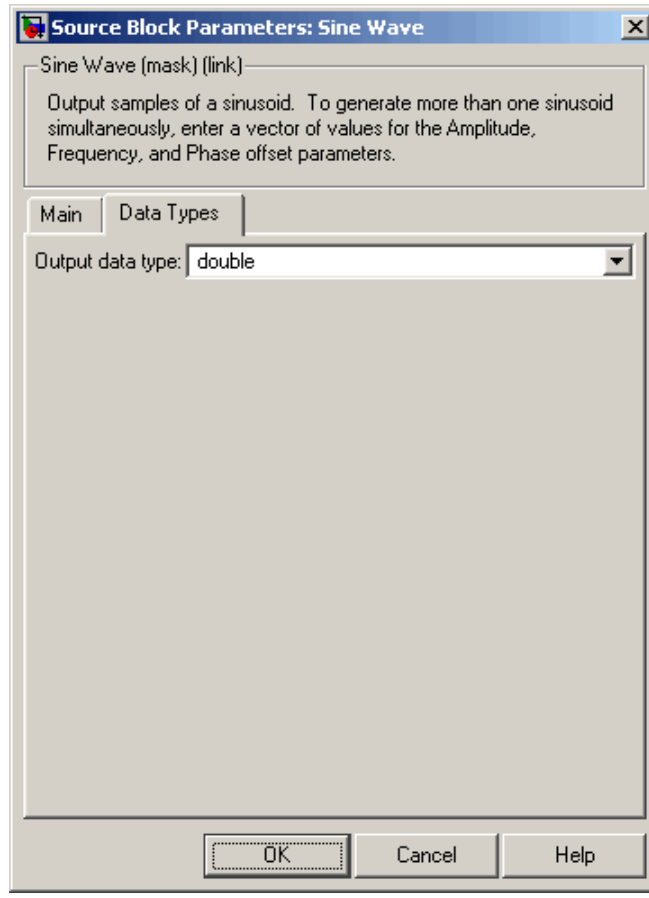
The number of consecutive samples from each sinusoid to buffer into the output frame, M. When the value of this parameter is 1, the block outputs a sample-based signal.

This parameter is disabled when you select Continuous from the Sample mode parameter.

Resetting states when re-enabled

This parameter only applies when the Sine Wave block is located inside an enabled subsystem and the States when enabling parameter of the Enable block is set to reset. This parameter determines the behavior of the Sine Wave block when the subsystem is re-enabled. The block can either reset itself to its starting state (Restart at time zero), or resume generating the sinusoid based on the current simulation time (Catch up to simulation time). This parameter is disabled when you select Continuous from the Sample mode parameter.

The Data types pane of the Sine Wave block dialog appears as follows.



Output data type

Specify the output data type in out of the following ways:

Choose one of the built-in data types from the list.

Choose Fixed-point to specify the output data type and scaling in the Word length , Set fraction length in output to , and Fraction length parameters.

Choose User-defined to specify the output data type and scaling in the User-defined data type, Set fraction length in output to, and Fraction length parameters.

Choose Inherit via back propagation to set the output data type and scaling to match the next block downstream.

Word length

Specify the word length, in bits, of the fixed-point output data type. This parameter is only visible when you select Fixed-point for the Output data type parameter.

User-defined data type

Specify any built-in or fixed-point data type. You can specify fixed-point data types using the following Simulink Fixed Point functions: sfix, ufix, sint, uint, sfrac, and ufrac. This parameter is only visible when you select User-defined for the Output data type parameter.

Set fraction length in output to

Specify the scaling of the fixed-point output by either of the following two methods:

Choose Best precision to have the output scaling automatically set such that the output signal has the best possible precision.

Choose User-defined to specify the output scaling in the Fraction length parameter.

This parameter is only visible when you select Fixed-point for the Output data type parameter, or when you select User-defined and the specified output data type is a fixed-point data type.

Fraction length

For fixed-point output data types, specify the number of fractional bits, or bits to the right of the binary point. This parameter is only visible when you select Fixed-point or User-defined for

the Output data type parameter and User-defined for the Set fraction length in output to parameter.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Fixed point (signed only)
- 8-, 16-, and 32-bit signed integers

See Also

Chirp	Signal Processing Blockset
Complex Exponential	Signal Processing Blockset
Signal From Workspace	Sig nal Processing Blockset
Signal Generator	Simulink
Sine Wave	Simulink
sin	MATLAB

Purpose

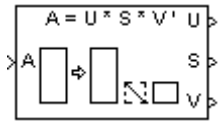
Factor matrix using singular value decomposition

Library

Math Functions / Matrices and Linea r Algebra / Matrix Factorizations

dspfactors

Description



The Singular Value Decomposition block factors the M-by-N input matrix A such that

$$A = U \cdot diag(S) \cdot V^*$$

where

- U is an M-by-P matrix
- V is an N-by-P matrix
- S is a length- P vector
- P is deÞned as min(M,N)

When

- M = N, U and V are both M-by-M unitary matrices
- M > N, V is an N-by-N unitary matrix, and U is an M-by-N matrix whose columns are the Þrst N columns of a unitary matrix
- N > M, U is an M-by-M unitary matrix, and V is an N-by-M matrix whose columns are the Þrst M columns of a unitary matrix

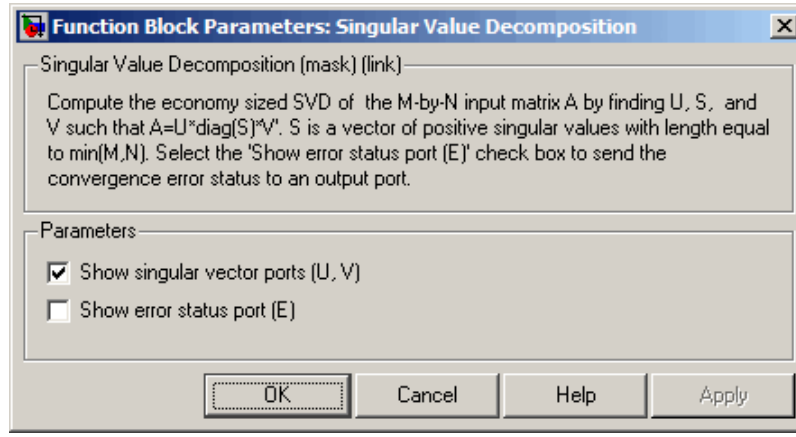
In all cases, S is a 1-D vector of positive singular values having length P.

Length- N row inputs are treated as length- N columns.

Note that the Þrst (maximum) element of output S is equal to the 2-norm of the matrix A.

The output is always sample based.

Dialog Box



Show singular vector ports

Select to enable the U and V output ports.

Show error status port

Select to enable the E output port, which reports a failure to converge. The possible values you can receive on the port are:

- 0 — The singular value decomposition calculation converges.
- 1 — The singular value decomposition calculation does not converge.

If the singular value decomposition calculation fails to converge, the output at ports U, S, and V are unde Þned matrices of the correct size.

References

Golub, G. H., and C. F. Van Loan. Matrix Computations . 3rd ed. Baltimore, MD: Johns Hopkin s University Press, 1996.

Supported Data Types

Port	Supported Data Types
A	• Double-precision ßoating point • Single-precision ßoating point
U	• Double-precision ßoating point • Single-precision ßoating point
S	• Double-precision ßoating point • Single-precision ßoating point
V	• Double-precision ßoating point • Single-precision ßoating point
E	• Boolean

See Also

Autocorrelation LPC	Signal Processing Blockset
Cholesky Factorization	Signal Processing Blockset
LDL Factorization	Signal Processing Blockset
LU Inverse	Signal Processing Blockset
Pseudoinverse	Signal Processing Blockset
QR Factorization	Signal Processing Blockset
SVD Solver	Signal Processing Blockset
svd	MATLAB

See “Matrix Factorizations” for related information.

Sort

Purpose

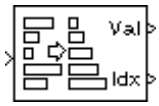
Sort input elements by value

Library

Statistics

dspstat3

Description



The Sort block ranks the values of the input elements using either a quick sort or an insertion sort algorithm. The quick sort algorithm uses a recursive sort method and is faster at sorting more than 32 elements. The insertion sort algorithm uses a non-recursive method and is faster at sorting less than 32 elements. You should also always use the insertion sort algorithm when you are generating code from the Sort block if you do not want recursive function calls in your code. To specify the sort method, use the Sort algorithm parameter.

The Mode parameter specifies the block's mode of operation, and can be set to Value, Index, or Value and index.

The Sort block supports real and complex floating-point and fixed-point inputs. Signed and unsigned fixed-point signals are supported. The block output has the same signedness as the input.

Value Mode

When Mode is set to Value, the block sorts the elements in each column of the M-by-N input matrix u in order of ascending or descending value, as specified by the Sort order parameter.

```
val = sort(u)
val = flipud(sort(u))
```

For convenience, length-M 1-D vector inputs and sample-based length-M row vector inputs are both treated as M-by-1 column vectors.

The output at each sample time, val, is an M-by-N matrix containing the sorted columns of u. The output has the same frame status as the input.

Complex inputs are sorted by magnitude squared. For complex value u = a + bi, the magnitude squared is $a^2 + b^2$.

Index Mode

When Mode is set to Index, the block sorts the elements in each column of the M-by-N input matrix u,

```
[val,idx] = sort(u)
[val,idx] = flipud(sort(u))
```

and outputs the sample-based M-by-N index matrix, idx. The j th column of idx is an index vector that permutes the j th column of u to the desired sorting order.

```
val(:,j) = u(idx(:,j),j)
```

The index value outputs are always 32-bit unsigned integer values.

As in Value mode, length-M 1-D vector inputs and sample-based length-M row vector inputs are both treated as M-by-1 column vectors.

Value and Index Mode

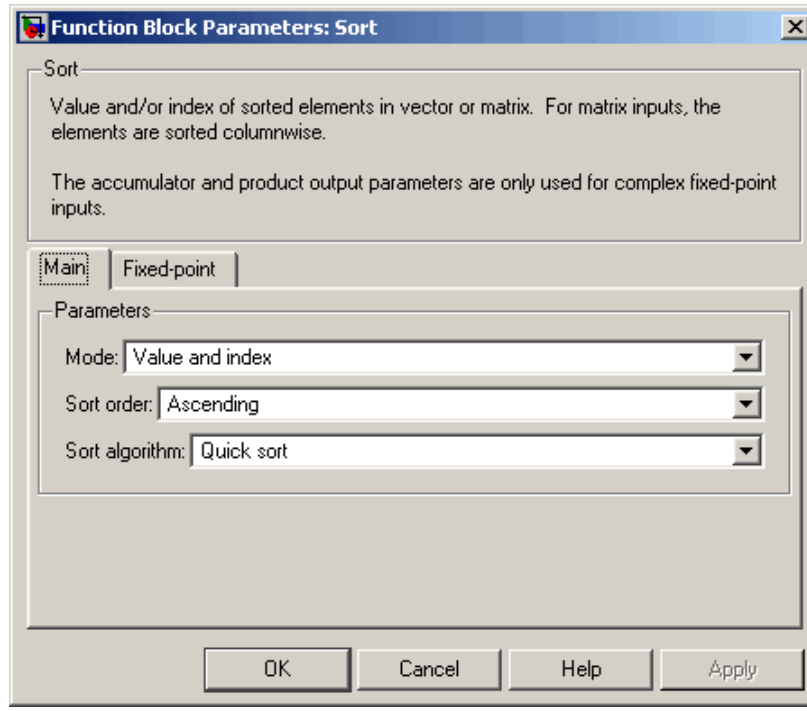
When Mode is set to Value and index, the block outputs both the sorted matrix, val, and the index matrix, idx.

Fixed-Point Data Types

The parameters on the Fixed-point pane are only used for complex Þxed-point inputs. Complex Þxed-point inputs are sorted by magnitude squared. The sum of the squares of the real and imaginary parts of such an input are formed before a com parison is made, as described in “Value Mode” on page 2-1176. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into the accumulator data type. These parameters are ign ored for other types of inputs.

Dialog Box

The Main pane of the Sort block dialog appears as follows.



Mode

Specify the block's mode of operation: Output the sorted matrix (`Value`), the index matrix (`Index`), or both (`Value and index`).

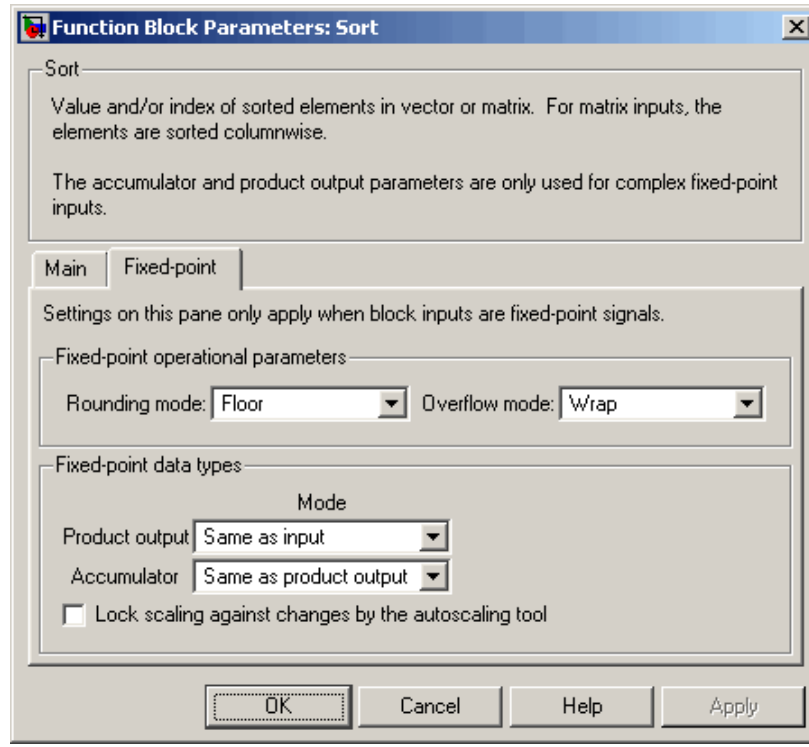
Sort order

Specify the order in which to sort the training points, `Descending` or `Ascending`.

Sort algorithm

Specify whether the elements of the input are sorted using a `Quick sort` or an `Insertion sort` algorithm.

The Fixed-point pane of the Sort block dialog appears as follows.



Note The parameters on the Fixed-point pane are only used for complex Þxed-point inputs. The sum of the squares of the real and imaginary parts of such an input a re formed before a comparison is made, as described in “Value Mode” on page 2-1176. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into the accumulator data type. These parameters are ignored for other types of inputs.

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths resulting from a complex-complex multiplication in the block. See “Multiplication Data Types” for more information:

When you select Same as input, these characteristics match those of the input to the block.

When you select Binary point scaling, you can enter the word length and the fraction length of the product output, in bits.

When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block. See “Multiplication Data Types” for more information:

When you select Same as product output, these characteristics match those of the product output

When you select Same as input, these characteristics match those of the input to the block.

When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.

When you select Slope and bias scaling , you can enter the word length, in bits, and the slop e of the accumulator. This block requires power-of-two slope and a bias of zero.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any Þxed-point scaling you specify in this block mask f rom being overridden by the autoscaling feature of the Fixed-Point Tool. See the fxptdlg reference page for more information.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, 32-, and 128-bit unsigned integers • 8-, 16-, 32-, and 128-bit signed integers
Val	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • 8-, 16-, 32-, and 128-bit unsigned integers • 8-, 16-, 32-, and 128-bit signed integers
Idx	• 32-bit unsigned integers

See Also

Histogram	Signal Processing Blockset
Median	Signal Processing Blockset
sort	MATLAB

Spectrum Scope

Purpose

Compute and display periodogram of each input signal

Library

Signal Processing Sinks

dspsnks4

Description

The Spectrum Scope block computes and displays the periodogram of the input. The input can be a sample-based or frame-based vector or a frame-based matrix.

Note When the Buffer input and Specify FFT length parameters are both cleared, the block input length must be a power of two.

Scope Properties Pane

The Buffer input check box must be selected for sample-based inputs. Buffering is optional for frame-based inputs. When the block input is buffered, you specify the number of input samples that the block buffers before computing and displaying the magnitude FFT in the Buffer size parameter. You also use the Buffer overlap parameter to specify the number of samples from the previous buffer to include in the current buffer. The number of new input samples the block acquires before computing and displaying the magni tude FFT is the difference between the buffer size and the buffer overlap.

The display update period is

$$(M_o - L) * T_s$$

where

- M_o = buffer size
- L = buffer overlap
- T_s = input sample period

For negative buffer overlap values, the block discards the appropriate number of input samples after the buffer Þlls, and updates the scope display at a slower rate than in the zero-overlap case.

The Window type and Window sampling parameters apply to the speciÞcation of the window function; see the Window Function block reference page for more details on these parameters.

The FFT length used by the block, N_{fft}, is determined in the following ways:

- If you clear the Specify FFT length check box and select Buffer input, the block uses the buffer size as the FFT size.
- If you clear the both the Specify FFT length and Buffer input check boxes, the block uses the input size as the FFT size.
- If you select the Specify FFT length check box, the FFT length parameter appears on the dialog box. Enter the number of samples on which you want the block to perform the FFT. This value must be a power of two.

The block zero pads or wraps every channel's buffer to the FFT length before computing the FFT.

The number of spectra to average is set by the Number of spectral averages parameter. Setting this parameter to 1 effectively disables averaging; see the Periodogram block reference page for more information.

Display Properties Pane

The Display Properties pane enables you to control how the block displays your data.

The Show grid parameter toggles the background grid on and off.

If you select the Persistence check box, the window maintains successive displays. That is, the scope does not erase the display after each frame (or collection of frames), but overlays successive input frames in the scope display.

If you select the Frame number check box, the block displays the number of the current frame in the input sequence on the scope window, and the block increments the count as each new input is received. Counting starts at 1 with the first input frame, and continues until the simulation stops.

If you select the Channel legend check box, a legend indicating the line color, style, and marker of each channel's data is added. When the input signal is labeled, that label appears in the channel legend. When the input signal is not labeled, but comes from a Concatenate block or a Mux block with labeled inputs, those labels appear in the channel legend. Otherwise, each channel in the legend is labeled with the channel number (CH 1, CH 2, etc.). Click and drag the legend to reposition it in the scope window; double-click on the line label to edit the text. If you rerun the simulation, the labels revert to the defaults.

If you select the Compact display check box, the scope completely fills the figure window. The scope does not display menus and axis titles, but it does show the numerical axis labels within the axes. If you clear the Compact display check box, the scope displays the axis labels and titles in a gray border surrounding the scope axes, and the window's menus and toolbar become visible.

If you select the Open scope at start of simulation check box, the scope opens at the start of the simulation. If you clear this parameter, the scope does not open automatically during the simulation. You can use this feature when you have several scope blocks in a model, and you do not want to view all the associated scopes during the simulation.

If you want to view a scope window that is not open during simulation, click Open scope immediately on the Display Properties pane of the desired Scope block.

The Scope position parameter specifies a four-element vector of the form

```
[left bottom width height]
```

specifying the position of the scope window on the screen, where (0,0) is the lower-left corner of the display. See the MATLAB figure function for more information.

Axis Properties Pane

The Frequency units parameter specifies whether the frequency axis values should be in units of Hertz or rad/sec. When the Frequency units parameter specifies Hertz, the spacing between frequency points is $1/(N_{fft}T_s)$. For Frequency units of rad/sec, the spacing between frequency points is $2\pi/(N_{fft}T_s)$.

The Frequency range parameter specifies the range of frequencies over which the magnitudes in the input should be plotted. The available options are [0..Fs/2], [-Fs/2..Fs/2], and [0..Fs], where F_s is the original time-domain signal's sample frequency.

Note that all of the Signal Processing Blockset FFT-based blocks, including those in the Power Spectrum Estimation library, compute the FFT at frequencies in the range [0, F_s). The Frequency range parameter controls only the displayed range of the signal.

The Frequency display offset parameter allows you to offset the range of values displayed on the frequency axis of the Spectrum Scope.

- When the Frequency display offset is 0, the DC frequency (0 Hz) is displayed at 0 Hz or 0 rad/sec, depending on the setting of the Frequency units parameter.
- When a non-zero value is specified for the Frequency display offset, the DC frequency (0 Hz) is displayed at the value specified in the Frequency display offset parameter. If the Frequency display limits parameter is set to User-defined, the frequency axis is not automatically relabeled. However, if the Frequency display limits parameter is set to Auto, the range of displayed values on the frequency axis shifts according to the value specified in the Frequency display offset parameter.

  For example, consider the following block settings:

  - Frequency units = Hertz

- Frequency range = [-Fs/2..Fs/2]
- Frequency display offset (Hz) = 0
- Frequency display limits = Auto
- Sampling frequency (Fs) = 1000 Hz

Based on these settings, the values on the frequency axis of the spectrum scope range from -500 Hz to 500 Hz, and the DC frequency (0 Hz) is centered at 0 Hz.

If you change the Frequency display offset (Hz) parameter to 100, the frequency axis is relabeled such that values range from -400 Hz to 600 Hz, and the DC frequency (0 Hz) is centered at 100 Hz.

If you select the Inherit sample increment from input check box, the block computes the frequency data from the sample period of the input to the block. This is valid when the following conditions hold:

- The input to the block is the original signal, with no samples added or deleted (by insertion of zeros, for example).
- The sample period of the time-domain signal in the simulation is equal to the period with which the physical signal was originally sampled.

In cases where not all of these conditions hold, specify the appropriate value for the Sample time of original time series parameter.

When Frequency display limits is set to Auto, the block scales the horizontal axis of frequency-domain signals automatically. To do this, the Spectrum Scope block needs to know the actual sample period of the time-domain input. Specify this in the Sample time of original time series parameter, T_s.

When Frequency display limits is set to User-defined, the Minimum frequency and Maximum frequency parameters set the range of the horizontal axis.

The Y–axis scaling parameter allows you to select Magnitude-squared or dB scaling along the y-axis.

Minimum Y-limit and Maximum Y-limit parameters allow you to set the range of the vertical axis. Setting these parameters is analogous to setting the ymin and ymax values of the MATLAB axis function.

The Y-axis title is the text displayed to the left of the y-axis.

Line Properties Pane

Use the parameters on the Line Properties pane to help you distinguish between two or more independent channels of data on the scope.

The Line visibilities parameter specifies which channel's data is displayed on the scope, and which is hidden. The syntax specifies the visibilities in list form, where the term on or off as a list entry specifies the visibility of the corresponding channel's data. The list entries are separated by the pipe symbol, |.

For example, a five-channel signal would ordinarily generate five distinct plots on the scope. To disable plotting of the third and fifth lines, enter the following visibility specification in the Line visibilities parameter.

```
 on | on | off | on | off
ch 1  ch 2  ch 3  ch 4  ch 5
```

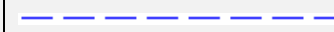
Note that the first (leftmost) list item corresponds to the first signal channel (leftmost column of the input matrix).

The Line styles parameter specifies the line style with which each channel's data is displayed on the scope. The syntax specifies the channel line styles in list form, with each list entry specifying a style for the corresponding channel's data. The list entries are separated by the pipe symbol, |.

For example, a five-channel signal would ordinarily generate all five plots with a solid line style. To plot each line with a different style, enter

```
   - | -- | : | -. | -
ch 1  ch 2  ch 3 ch 4  ch 5
```

These settings plot the signal channels with the following styles.

Line Style	Command to Type in Line Style Parameter	Appearance
Solid	-	
Dashed	--	
Dotted	:	
Dash-dot	-.	
No line	none	No line appears

Note that the first (leftmost) list item, '-' , corresponds to the first signal channel (leftmost column of the input matrix). See the LineStyle property of the MATLAB line function for more information about the style syntax.

The Line markers parameter specifies the marker style with which each channel's samples are represented on the scope. The syntax specifies the channels' marker styles in list form, with each list entry specifying a marker for the corresponding channel's data. The list entries are separated by the pipe symbol, |.

For example, a five-channel signal would ordinarily generate all five plots with no marker symbol (that is, the individual sample points are not marked on the scope). To instead plot each line with a different marker style, you could enter

```
  * | . | x | s | d
ch 1    ch 2    ch 3    ch 4    ch 5
```

These settings plot the signal channels with the following styles.

Marker Style	Command to Type in Marker Style Parameter	Appearance
Asterisk	*	
Point	.	
Cross	x	
Square	s	
Diamond	d	

Note that the leftmost list item, `'*'`, corresponds to the first signal channel or leftmost column of the input matrix. See the Marker property of the MATLAB line function for more information about the available markers.

To produce a stem plot for the data in a particular channel, type the word stem instead of one of the basic marker shapes.

The Line colors parameter specifies the color in which each channel's data is displayed on the scope. The syntax specifies the channel colors in list form, with each list entry specifying a color (in one of the MATLAB ColorSpec formats) for the corresponding channel's data. The list entries are separated by the pipe symbol, |.

For example, a five-channel signal would ordinarily generate all five plots in the color black. To instead plot the lines with the color order below, enter

```
[0 0 0] | [0 0 1] | [1 0 0 ] | [0 1 0] | [.7529 0 .7529]
  ch 1       ch 2       ch 3       ch 4          ch 5
```

or

```
 'k' | 'b' | 'r' | 'g' | [.7529 0 .7529]
ch 1   ch 2   ch 3   ch 4         ch 5
```

These settings plot the signal channels in the following colors (8-bit RGB equivalents shown in the center column).

Color	RGB Equivalent	Appearance
Black	(0,0,0)	
Blue	(0,0,255)	
Red	(255,0,0)	
Green	(0,255,0)	
Dark purple	(192,0,192)	

Note that the leftmost list item, 'k', corresponds to the first signal channel or leftmost column of the input matrix. See the MATLAB function ColorSpec for more information about the color syntax.

Spectrum Scope Window

The title that appears in the title bar of the scope window is the same as the block title. In addition to the standard MATLAB figure window menus such as File, Window, and Help, the Spectrum Scope window contains View, Axes, and Channels menus.

The options in the View menu allow you to zoom in and out of the scope window:

- To zoom in on the scope window, you must first select View > Zoom In or click the corresponding Zoom In toolbar button ( ). You can then zoom in by clicking in the center of your area of interest, or by clicking and dragging your cursor to draw a rectangular area of interest inside of the scope window.
- To zoom in on the x-axis of the scope window, you must first select View > Zoom X or click the corresponding Zoom X-Axis toolbar button ( ) on the scope window. You can then zoom in on the x-axis with a single click inside the scope window, or by clicking and dragging the cursor along the x-axis over your area of interest.
- To zoom in on the y-axis of the scope window, you must first select View > Zoom Y or click the corresponding Zoom Y-Axis toolbar button ( ). You can then zoom in on the y-axis with a single click inside the scope window, or by clicking and dragging the cursor along the y-axis over your area of interest.
- To return to the original view of the scope window, you have the following options:
  - Select Full View from the View menu.
  - Click the Restore default view toolbar button ( ) on the Spectrum Scope window.
  - Right-click inside the scope window and select Reset to Original View.

Note To zoom out in smaller increments, you can right-click inside of the scope window and select Zoom Out. You can also zoom out by holding down the Shift key and clicking the left mouse button inside the scope window.

The parameters that you set using the Axes menu apply to all channels. Many of the parameters in this menu are also accessible through the block parameters dialog box. For descriptions of these parameters, see "Display Properties Pane" on page 2-1432. Below are descriptions of other parameters in the Axes menu:

- Refresh erases all data on the scope display, except for the most recent trace. This command is useful in conjunction with the Persistence setting.
- Autoscale resizes the y-axis to best fit the vertical range of the data.

Note The Minimum Y-limit and Maximum Y-limit parameters on the Axis properties pane of the block dialog are not updated to display the numerical limits selected by the autoscale feature. See "Zoom Capability for Spectrum Scope and Vector Scope Blocks" in the Signal Processing Blockset Release Notes for more information.

- Save Axes Settings allows you to save the current axes settings. When you select this option, the Minimum Y-limit and Maximum Y-limit parameters of the Axes Properties pane update with the current y-axes limits. The Frequency display limits parameter is set to User-defined, and the current x-axes limits are saved in the Minimum Frequency and Maximum Frequency parameters. To save these axes settings for your next MATLAB session, you need to resave your model.
- Save Scope Position automatically updates the Scope position parameter in the Axis Properties pane of the block dialog to

reßect the scope window's current position and size. To make the scope window open at a particular location on the screen when the simulation runs, drag the window to the desired location, resize it, and select Save Scope Position from the Axes menu.

The properties listed in the Channels menu apply to a particular channel. All of the parameters in this menu are also accessible through the block parameters dialog box. For descriptions of these parameters, see "Line Properties Pane" on page 2-1437.

Many of these options can also be accessed by right-clicking with the mouse anywhere on the scope display. The menu that is displayed contains a combination of the options available in the View , Axes and Channels menus.

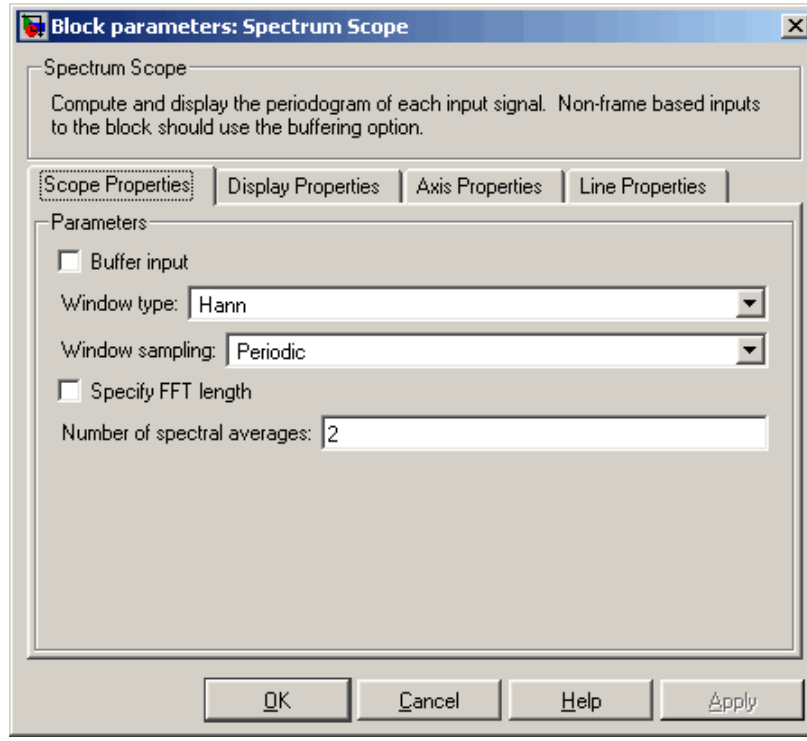
Note When you select Compact Display from the Axes menu, the scope window menus are no longer visible. Right-click in the Spectrum Scope window and click Compact Display in order to make the menus reappear.

Examples

See "Displaying Frequency-Domain Data" in the Signal Processing Blockset User's Guide.

Dialog Box

Scope Properties Pane



Buffer input

Select this check box to rebuffer the input data. This check box must be selected for sample-based inputs, but is optional for frame-based inputs.

This functionality is not supported for use with external mode. Instead, clear this check box and use a Buffer block prior to the Spectrum Scope in your model.

Buffer size

Specify the number of input samples that the block buffers before computing and displaying the magnitude FFT. When the Specify FFT length parameter is not selected, this value must be a power of two.

This parameter is only visible when the Buffer input check box is selected.

Buffer overlap

Specify the number of samples from the previous buffer to include in the current buffer. The number of new input samples the block acquires before computing and displaying the magnitude FFT is the difference between the buffer size and the buffer overlap.

This parameter is only visible when the Buffer input check box is selected.

Window type

Enter the type of window to apply. See the Window Function block reference page for more details. Tunable.

Stopband attenuation in dB

Enter the level, in dB, of stopband attenuation, R_s, for the Chebyshev window.. Tunable.

This parameter is only visible when Chebyshev is selected for the Window type parameter.

Beta

Enter the β parameter for the Kaiser window. Increasing Beta widens the mainlobe and decreases the amplitude of the window sidelobes in the window's frequency magnitude response. Tunable.

This parameter is only visible if Kaiser is selected for the Window type parameter.

Window sampling

Choose Symmetric or Periodic . Tunable.

This parameter is only visible if Blackman , Hamming, Hann, or Hanning is selected for the Window type parameter.

Specify FFT length

Select this check box to specify the FFT length yourself in the FFT length parameter.

FFT length

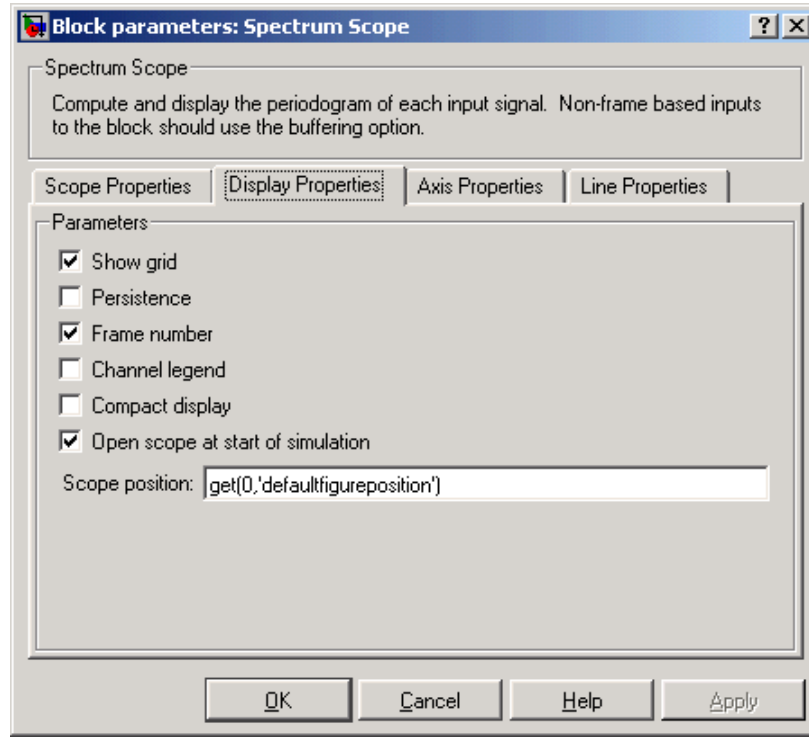
Enter the number of samples on which you want the block to perform the FFT. This value must be a power of two.

This parameter is only visible when then Specify FFT length check box is selected.

Number of spectral averages

The number of spectra to average. Setting this parameter to 1 effectively disables averaging. See the Periodogram block reference page for more information.

Display Properties Pane



Show grid

Toggle the scope grid on and off. Tunable.

Persistence

Select this check box to maintain successive displays. That is, the scope does not erase the display after each frame (or collection of frames), but overlays successive input frames in the scope display. Tunable.

Frame number

If you select this check box, the number of the current frame in the input sequence appears in the Vector Scope window. Tunable.

Channel legend
: Toggles the legend on and off. Tunable.

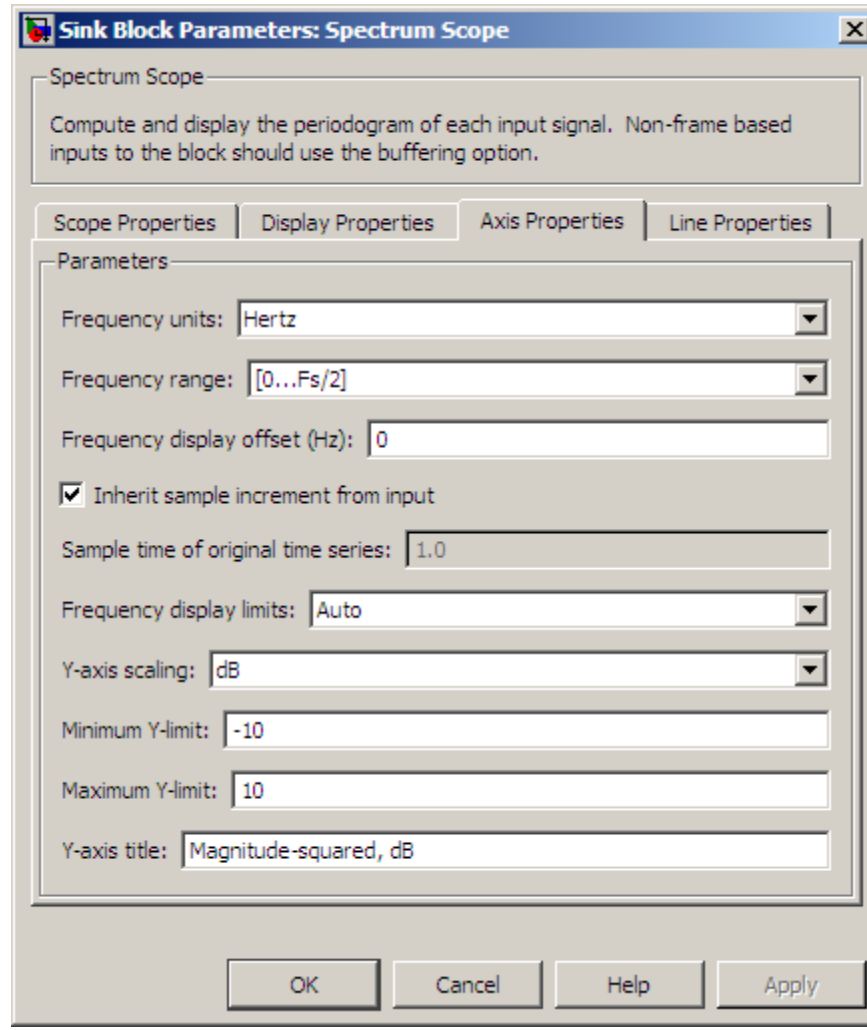
Compact display
: Resizes the scope to fill the window. Tunable.

Open scope at start of simulation
: Select this check box to open the scope at the start of the simulation. When this parameter is cleared, the scope not open automatically during the simulation. Tunable.

Open scope immediately
: If the scope is not open during simulation, select this check box to open it. This parameter is visible only while the simulation is running.

Scope position
: A four-element vector of the form `[left bottom width height]` specifying the position of the scope window. `(0,0)` is the lower-left corner of the display. Tunable.

Axis Properties Pane



Frequency units

Choose either Hertz or rad/sec as the frequency units for the horizontal axis. Tunable.

Frequency range

Specify the frequency range over which to plot the data. Tunable.

Frequency display offset

The Frequency display offset parameter allows you to offset the range of values displayed on the frequency axis of the Spectrum Scope. The value specified in this field becomes the new label for the DC frequency (0 Hz). See the example in the Description section for more information.

Inherit sample increment from input

If you select this check box, the block computes the time-domain sample period from the frame period and frame size of the frequency-domain input. Use this parameter only when the length of each frame of frequency-domain data is the same as the length of the frame of time-domain data from which it was generated. Tunable.

Sample time of original time series

Enter the sample period of the original time-domain signal. Tunable.

Frequency display limits

Select Auto to have the limits of the x-axis set for you automatically, or User-defined to set the limits yourself in the Minimum frequency and Maximum frequency parameters.

Minimum frequency

Specify the minimum frequency value of the x-axis in Hertz or rad/sec. This parameter is only visible if the Frequency display limits parameter is set to User-defined. Tunable.

Maximum frequency

Specify the maximum frequency value of the x-axis in Hertz or rad/sec. This parameter is only visible if the Frequency display limits parameter is set to User-defined. Tunable.

Y-axis scaling

Choose either dB (decibel) or Magnitude-squared scaling for the y-axis. Tunable.

Minimum Y-limit

Specify the minimum value of the y-axis. Setting this parameter is analogous to setting the ymin value of the MATLAB axis function.Tunable.

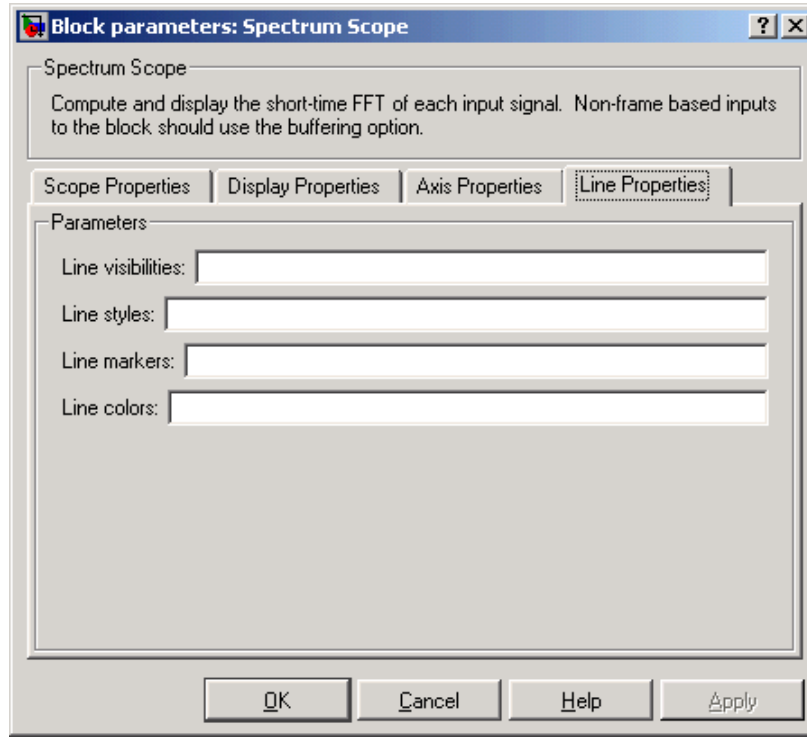
Maximum Y-limit

Specify the maximum value of the y-axis. Setting this parameter is analogous to setting the ymax value of the MATLAB axis function.Tunable.

Y-axis title

Specify text to be displayed to the left of the y-axis. Tunable.

Line Properties Pane



For more information about these parameters, see “Line Properties Pane” on page 2-1437 in the Vector Scope block reference page.

Line visibilities

Enter `on` or `off` to specify the visibility of the various channels' scope traces. Separate your choices for each channel with by a pipe (|) symbol. Tunable.

Line styles

Enter the line styles of the various channels' scope traces using the MATLAB `line` function `LineStyle` formats. Separate your choices for each channel with by a pipe (|) symbol. Tunable.

Line markers

Enter the line markers of the various channels' scope traces using the MATLAB line function Marker formats. Separate your choices for each channel with by a pipe (|) symbol. Tunable.

Line colors

Enter the colors of the various channels' scope traces using the MATLAB ColorSpec formats. Separate yo ur choices for each channel with by a pipe (|) symbol. Tunable.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

FFT	Signal Processing Blockset
Periodogram	Signal Processing Blockset
Short-Time FFT	Signal Processing Blockset
Vector Scope	Signal Processing Blockset
Window Function	Signal Processing Blockset

Stack

Purpose

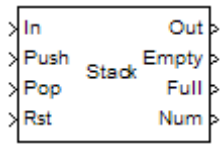
Store inputs into LIFO register

Library

Signal Management / Buffers

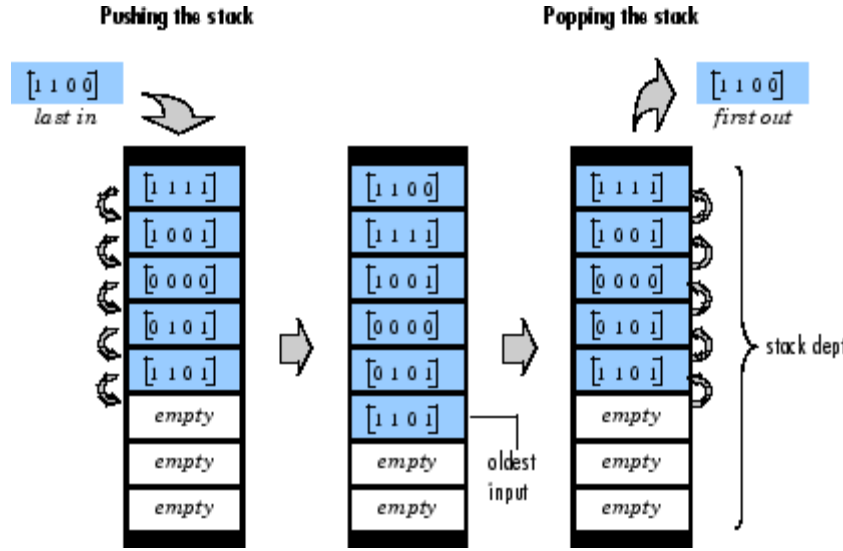
dspbuff3

Description



The Stack block stores a sequence of input samples in a last in, first out (LIFO) register. The register capacity is set by the Stack depth parameter, and inputs can be scalars, vectors, or matrices.

The block pushes the input at the In port onto the top of the stack when a trigger event is received at the Push port. When a trigger event is received at the Pop port, the block pops the top element off the stack and holds the Out port at that value. The last input to be pushed onto the stack is always the first to be popped off.



A trigger event at the optional Rst port empties the stack contents. When you select Clear output port on reset, then a trigger event at the Rst port empties the stack and sets the value at the Out port to zero. This setting also applies when a disabled subsystem containing

the Stack block is reenabled; the Out port value is only reset to zero in this case when you select Clear output port on reset .

When two or more of the control input ports are triggered at the same time step, the operations are executed in the following order:

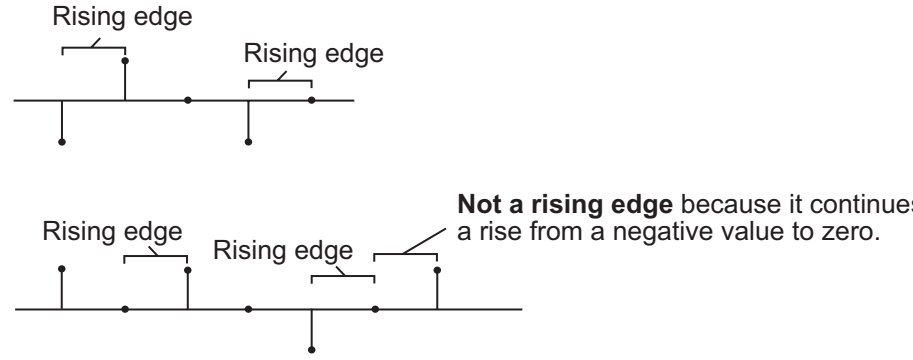
1 Rst

2 Push

3 Pop

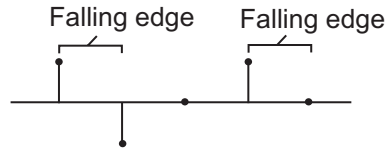
The rate of the trigger signal must be the same as the rate of the data signal input. You specify the triggering event for the Push, Pop, and Rst ports in the Trigger type pop-up menu:

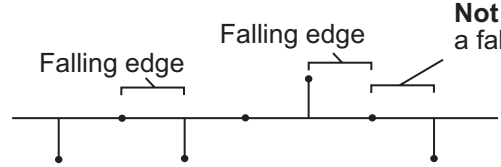
- Rising edge — Triggers execution of the block when the trigger input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following Þgure)



- Falling edge — Triggers execution of the block when the trigger input does one of the following:

- Falls from a positive value to a negative value or zero
- Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)





Not a falling edge because it continues a fall from a positive value to zero.

- Either edge — Triggers execution of the block when the trigger input is a Rising edge or Falling edge (as described above).
- Non-zero sample — Triggers execution of the block at each sample time that the trigger input is not zero.

Note When running simulations in the Simulink MultiTasking mode, sample-based trigger signals have a one-sample latency, and frame-based trigger signals have one frame of latency. Thus, there is a one-sample or one-frame delay between the time the block detects a trigger event, and when it applies the trigger. For more information on latency and the Simulink tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Mo dels with Multiple Sample Rates” in the Real-Time Workshop User’s Guide .

Note If your model contains any referenced models that use the Stack block, you cannot simulate the top-level model in Simulink Accelerator mode.

The Push full stack parameter specifies the block's behavior when a trigger is received at the Push port but the register is full. The Pop empty stack parameter specifies the block's behavior when a trigger is received at the Pop port but the register is empty. The following options are available for both cases:

- Ignore — Ignore the trigger event, and continue the simulation.
- Warning — Ignore the trigger event, but display a warning message in the MATLAB command window.
- Error — Display an error dialog box and terminate the simulation.

Note The Push full stack and Pop empty stack parameters are diagnostic parameters. Like all diagnostic parameters on the Configuration Parameters dialog box, they are set to Ignore in the code generated for this block by Real-Time Workshop code generation software.

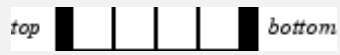
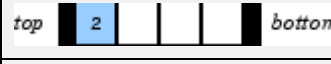
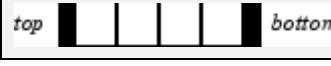
The Push full stack parameter additionally offers the Dynamic reallocation option, which dynamically resizes the register to accept as many additional inputs as memory permits. To find out how many elements are on the stack at a given time, enable the Num output port by selecting the Show number of stack entries port parameter.

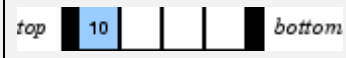
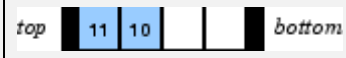
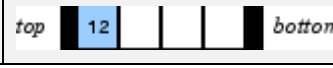
Note When Dynamic reallocation is selected, the System target file parameter on the Real-Time Workshop pane of the Configuration Parameters dialog box must be set to grt_malloc.tlc Generic Real-Time Target with dynamic memory allocation .

Examples

Example 1

The table below illustrates the Stack block's operation for a Stack depth of 4, Trigger type of Either edge , and Clear output port on reset enabled. Because the block triggers on both rising and falling edges in this example, each transition from 1 to 0 or 0 to 1 in the Push, Pop, and Rst columns below represents a distinct trigger event. A 1 in the Empty column indicates an empty buffer, while a 1 in the Full column indicates a full buffer.

In	Push	Pop	Rst	Stack	Out	Empty	Full	Num
1	0	0	0		0	1	0	0
2	1	0	0		0	0	0	1
3	0	0	0		0	0	0	2
4	1	0	0		0	0	0	3
5	0	0	0		0	0	1	4
6	0	1	0		5	0	0	3
7	0	0	0		4	0	0	2
8	0	1	0		3	0	0	1
9	0	0	0		2	1	0	0

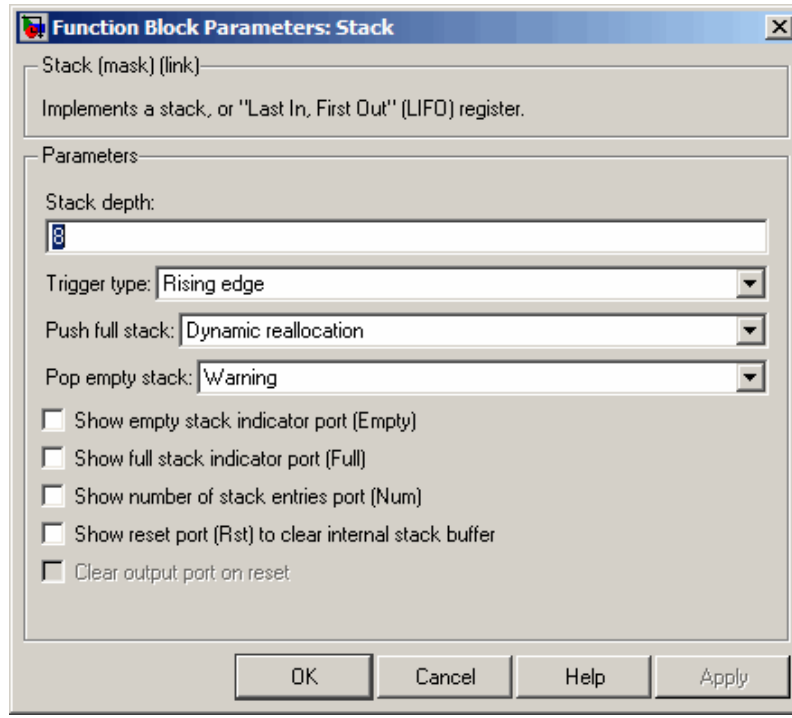
In	Push	Pop	Rst	Stack	Out	Empty	Full	Num
10	1	0	0		2	0	0	1
11	0	0	0		2	0	0	2
12	1	0	1		0	0	0	1

Note that at the last step shown, the Push and Rst ports are triggered simultaneously. The Rst trigger takes precedence, and the stack is Þrst cleared and then pushed.

Example 2

The dspqdemo demo provides an example of the related Queue block.

Dialog Box



Stack depth

The number of entries that the LIFO register can hold.

Trigger type

The type of event that triggers the block's execution. The rate of the trigger signal must be the same as the rate of the data signal input.

Push full stack

Response to a trigger received at the Push port when the register is full. Inputs to this port must have the same built-in data type as inputs to the Pop and Rst input ports.

When Dynamic reallocation is selected, the System target file parameter on the Real-Time Workshop pane of the Configuration Parameters dialog box must be set to grt_malloc.tlc Generic Real-Time Target with dynamic memory allocation .

Pop empty stack

Response to a trigger received at the Pop port when the register is empty. Inputs to this port must have the same built-in data type as inputs to the Push and Rst input ports.

Show empty stack indicator port

Enable the Empty output port, which is high (1) when the stack is empty, and low (0) otherwise.

Show full stack indicator port

Enable the Full output port, which is high (1) when the stack is full, and low (0) otherwise. The Full port remains low when you select Dynamic reallocation from the Push full stack parameter.

Show number of stack entries port

Enable the Numoutput port, which tracks the number of entries currently on the stack. When inputs to the In port are double-precision values, the outputs from the Numport are double-precision values. Otherwise, the outputs from the Numport are 32-bit unsigned integer values.

Show reset port to clear internal stack buffer

Enable the Rst input port, which empties the stack when the trigger specified by the Trigger type is received. Inputs to this port must have the same built-in data type as inputs to the Push and Pop input ports.

Clear output port on reset

Reset the Out port to zero (in addition to clearing the stack) when a trigger is received at the Rst input port.

Supported Data Types

Port	Supported Data Types
In	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Push	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers Inputs to this port must have the same built-in data type as inputs to the Pop and Rst input ports
Pop	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers Inputs to this port must have the same built-in data type as inputs to the Push and Rst input ports.

Port	Supported Data Types
Rst	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers Inputs to this port must have the same built-in data type as inputs to the Push and Pop input ports.
Out	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Empty	• Double-precision ßoating point • Boolean
Full	• Double-precision ßoating point • Boolean
Num	• Double-precision ßoating point The block outputs a double-precision ßoating-point value at this port when the data type of the In port is double-precision ßoating-point. • 32-bit unsigned integers The block outputs a 32-bit unsigned integer value at this port when the data type of the In port is anything other than double-precision ßoating-point.

See Also

Buffer	Signal Processing Blockset
Delay Line	Signal Processing Blockset
Queue	Signal Processing Blockset

Purpose

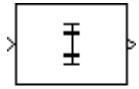
Find standard deviation of input or sequence of inputs

Library

Statistics

dspstat3

Description



The Standard Deviation block compu tes the standard deviation of each row or column of the input, along vectors of a speci fied dimension of the input, or of the entire input. The Standard Deviation block can also track the standard deviation of a sequence of inputs over a period of time. The Running standard deviation parameter selects between basic operation and running operation.

Basic Operation

When you do not select the Running standard deviation check box, the block computes the standard d eviation of each row or column of the input, along vectors of a speci fied dimension of the input, or of the entire input at each individual sample time, and outputs the array y. Each element in y contains the standard devia tion of the corresponding column, row, vector, or entire input. The output y depends on the setting of the Find the standard deviation value over parameter. For example, consider a 3-dimensional input signal of size M-by-N-by-P:

- Entire input — The output at each sample time is a scalar that contains the standard deviation of the entire input. In this mode, the output is always sample based.

  ```
  y = std(u(:))        % Equivalent MATLAB code
  ```

- Each Row — The output at each sample time consists of an M-by-1-by-P array, where each element contains the standard deviation of each vector over the second dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is an M-by-1 column vector. In this mode, the frame status of the output is the same as that of the input.

  ```
  y = std(u,0,2)       % Equivalent MATLAB code
  ```

- Each Column — The output at each sample time consists of a 1-by-N-by-P array, where each element contains the standard deviation of each vector over the first dimension of the input. For an input that is an M-by-N matrix, the output at each sample time is a 1-by-N row vector. In this mode, the frame status of the output is the same as that of the input.

  ```
  y = std(u,0,1)        % Equivalent MATLAB code
  ```

  For convenience, length-M 1-D vector inputs are treated as M-by-1 column vectors when the block is in this mode. Sample-based length-M row vector inputs are also treated as M-by-1 column vectors when the Treat sample-based row input as a column check box is selected.

- Specified Dimension — The output at each sample time depends on Dimension. If Dimension is set to 1, the output is the same as when you select Each column. If Dimension is set to 2, the output is the same as when you select Each row. If Dimension is set to 3, the output at each sample time is an M-by-N matrix containing the standard deviation of each vector over the third dimension of the input. In this mode, the frame status of the output is the same as that of the input.

  ```
  y = std(u,0,Dimension)        % Equivalent MATLAB code
  ```

For purely real or purely imaginary inputs, the standard deviation of the jth column of an M-by-N input matrix is the square root of its variance:

$$y_j = \sigma_j = \sqrt{\frac{\sum_{i=1}^{M} \left| u_{ij} - \mu_j \right|^2}{M-1}} \qquad 1 \le j \le N$$

For complex inputs, the output is the total standard deviation, which equals the square root of the total variance, or the square root of the

sum of the variances of the real an d imaginary parts. The standard deviation of each column in an M-by-N input matrix is given by:

$$\sigma_j = \sqrt{\sigma_{j,\mathrm{Re}}^2 + \sigma_{j,\mathrm{Im}}^2}$$

Note The total standard deviation does not equal the sum of the real and imaginary standard deviations.

Running Operation

When you select the Running standard deviation check box, the block tracks the standard deviatio n of successive inputs to the block. For sample-based M-by-N inputs, the output is a sample-based M-by-N matrix with each element y_{ij} containing the standard deviation of element u_{ij} over all inputs since the last reset. For frame-based M-by-N inputs, the output is a frame-based M-by-N matrix with each element y_{ij} containing the standard deviation of the jth column over all inputs since the last reset, up to and including element u_{ij} of the current input.

N-D signals cannot be frame based. When the block is set to Running mode, each element of the N-D signal is treated as a separate channel.

There are $\prod d_i$ channels, where d_i is the size of the ith dimension.

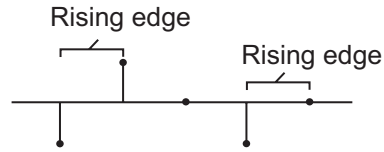
Resetting the Running Standard Deviation

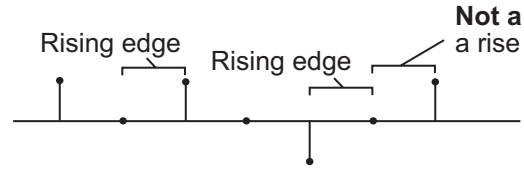
The block resets the running standard deviation whenever a reset event is detected at the optional Rst port. The reset sample time must be a positive integer multipl e of the input sample time.

You specify the reset event in the Reset port parameter:

- None disables the Rst port.
- Rising edge — Triggers a reset operation when the Rst input does one of the following:
  - Rises from a negative value to a positive value or zero

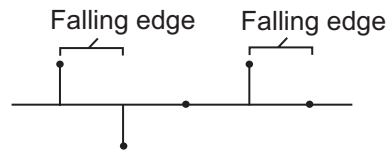
- Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following Þgure)

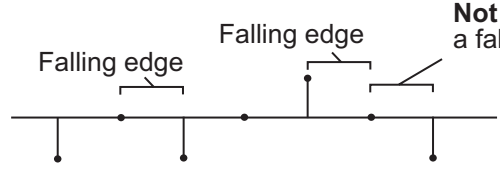




Not a rising edge because it continues a rise from a negative value to zero.

- Falling edge — Triggers a reset operation when the Rst input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following Þgure)





Not a falling edge because it continues a fall from a positive value to zero.

- Either edge — Triggers a reset operation when the Rst input is a Rising edge or Falling edge (as described earlier)
- Non-zero sample — Triggers a reset operation at each sample time that the Rst input is not zero

---

Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see "Excess Algorithmic Delay (Tasking Latency)" and "Models with Multiple Sample Rates" in the Real-Time Workshop User's Guide.

ROI Processing

To calculate the statistical value within a particular region of interest (ROI) of the input, select the Enable ROI processing check box. This option is only available when the Find the standard deviation value over parameter is set to Entire input and the Running standard deviation check box is not selected. ROI processing is only supported for 2-D inputs.

Note Full ROI processing is only available to users who have a Video and Image Processing Blockset license. If you only have a Signal Processing Blockset license, you can still use ROI processing, but are limited to the ROI type Rectangles.

Use the ROI type parameter to specify whether the ROI is a rectangle, line, label matrix, or binary mask. A binary mask is a binary image that enables you to specify which pixels to highlight, or select. In a label matrix, pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on. When the ROI type parameter is set to Label matrix, the Label and Label Numbers ports appear on the block. Use the Label

Numbers port to specify the objects in the label matrix for which the block calculates statistics. The input to this port must be a vector of scalar values that correspond to the labeled regions in the label matrix. For more information about the format of the input to the ROI port when the ROI is a rectangle or a line, see the Draw Shapes block reference page.

For rectangular ROIs, use the ROI portion to process parameter to specify whether to calculate the statistical value for the entire ROI or just the ROI perimeter.

Use the Output parameter to specify the block output. The block can output separate statistical values for each ROI or the statistical value for all specified ROIs. This parameter is not available if, for the ROI type parameter, you select Binary mask.

If, for the ROI type parameter, you select Rectangles or Lines, the Output flag indicating if ROI is within image bounds check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

Output = Individual statistics for each ROI

Flag Port Output	Description
0	ROI is completely outside the input image.
1	ROI is completely or partially inside the input image.

Output = Single statistic for all ROIs

Flag Port Output	Description
0	All ROIs are completely outside the input image.
1	At least one ROI is completely or partially inside the input image.

If the ROI is partially outside the image, the block only computes the statistical values for the portion of the ROI that is within the image.

If, for the ROI type parameter, you select Label matrix, the Output flag indicating if input label numbers are valid check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

Output = Individual statistics for each ROI

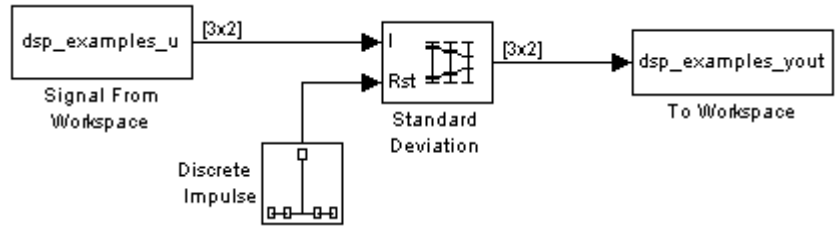
Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

Output = Single statistic for all ROIs

Flag Port Output	Description
0	None of the label numbers are in the label matrix.
1	At least one of the label numbers is in the label matrix.

Examples

The Standard Deviation block in the next model calculates the running standard deviation of a frame-based 3-by-2 (two-channel) matrix input, u. The running standard deviation is reset at t=2 by an impulse to the block's Rst port.



The Standard Deviation block has the following settings:

- Running standard deviation = ☑
- Reset port = Non-zero sample

The Signal From Workspace block has the following settings:

- Signal = dsp_examples_u
- Sample time = 1/3
- Samples per frame = 3

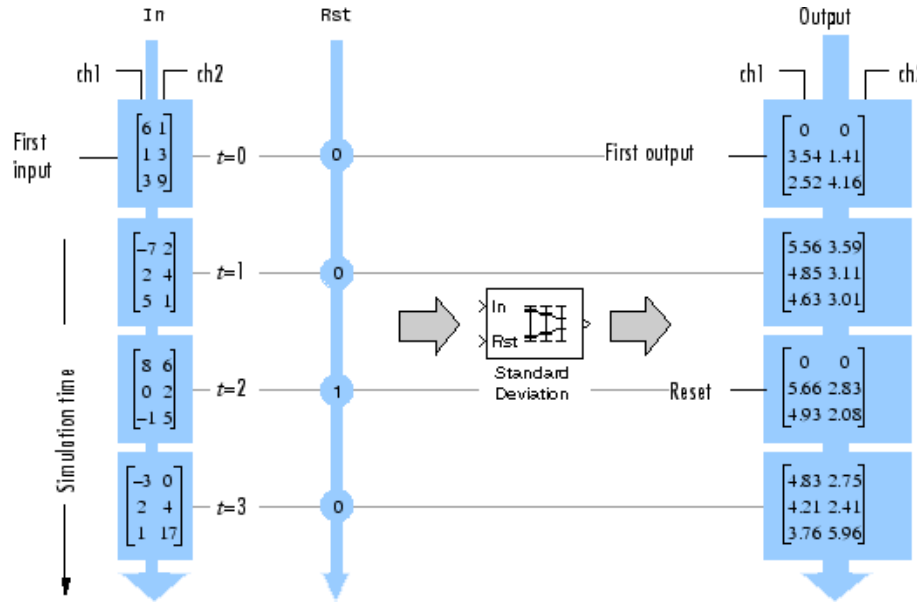
where

```
dsp_examples_u = [6 1 3 -7 2 5 8 0 -1 -3 2 1;1 3 9 2 4 1 6 2 5 0 4 17]'
```

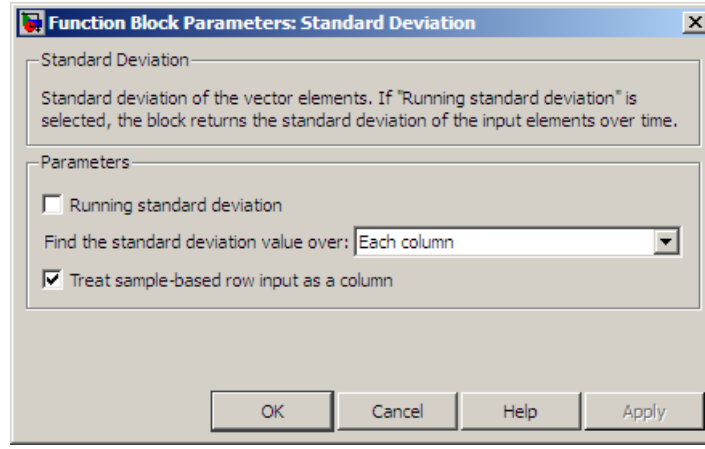
The Discrete Impulse block has the following settings:

- Delay (samples) = 2
- Sample time = 1
- Samples per frame = 1

The block's operation is shown in the next figure.



Dialog Box



Running standard deviation

Enables running operation when selected.

Reset port

Determines the reset event that causes the block to reset the running standard deviation. The reset signal rate must be a positive integer multiple of the rate of the data signal input. This parameter is enabled only when you select Running standard deviation. For more information, see “Resetting the Running Standard Deviation” on page 2-1217.

Find the standard deviation value over

Specify whether to find the standard deviation value along rows, columns, entire input, or the dimension specified in the Dimension parameter. For more information, see “Basic Operation” on page 2-1215.

Treat sample-based row input as a column

Select to treat sample-based length- M row vector inputs as M-by-1 column vectors. This parameter is only visible when the Find the standard deviation value over parameter is set to Each column.

Dimension

Specify the dimension (one-based value) of the input signal, over which the standard deviation is computed. The value of this parameter cannot exceed the number of dimensions in the input signal. This parameter is only visible when the Find the standard deviation value over parameter is set to Specified dimension.

Enable ROI Processing

Select this check box to calculate the statistical value within a particular region of each image. This parameter is only available when the Find the standard deviation value over parameter is set to Entire input, and the block is not in running mode.

Note Full ROI processing is only available to users who have a Video and Image Processing Blockset license. If you only have a Signal Processing Blockset license, you can still use ROI processing, but are limited to the ROI type Rectangles .

ROI type

Specify the type of ROI you want to use. Your choices are Rectangles , Lines , Label matrix , or Binary mask .

ROI portion to process

Specify whether you want to calcu late the statistical value for the entire ROI or just the ROI perimeter. This parameter is only visible if, for the ROI type parameter, you specify Rectangles .

Output

Specify the block output. The blo ck can output a vector of separate statistical values for each ROI or a scalar value that represents the statistical value for all the speci ed ROIs. This parameter is not available if, for the ROI type parameter, you select Binary mask.

Output flag indicating if ROI is within image bounds

If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in “ROI Processing” on page 2-1219. This parameter is visible if, for the ROI type parameter, you select Rectangles or Lines .

Output flag indicating if label numbers are valid

If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in “ROI Processing” on page 2-1219. This parameter is visible if, for the ROI type parameter, you select Label matrix .

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point
Reset	• Double-precision ßoating point • Single-precision ßoating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
ROI	Rectangles and lines: • Double-precision ßoating point • Single-precision ßoating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers Binary Mask: • Boolean
Label	• 8-, 16-, and 32-bit unsigned integers
Label Numbers	• 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point
Flag	• Boolean

See Also

Mean	Signal Processing Blockset
RMS	Signal Processing Blockset
Variance	Signal Processing Blockset
std	MATLAB

Submatrix

Purpose

Select subset of elements (submatrix) from matrix input

Library

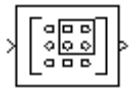
- Math Functions / Matrices and Lin ear Algebra / Matrix Operations

  dspmtrx3

- Signal Management / Indexing

  dspindex

Description



The Submatrix block extracts a cont iguous submatrix from the M-by-N input matrix u. A length-M 1-D vector input is treated as an M-by-1 matrix. The Row span parameter provides three options for specifying the range of rows in u to be retained in submatrix output y:

- All rows

  SpeciÞes that y contains all M rows of u.

- One row

  SpeciÞes that y contains only one row from u. The Starting row parameter (described below) is enabled to allow selection of the desired row.

- Range of rows

  SpeciÞes that y contains one or more rows from u. The Row and Ending row parameters (described bel ow) are enabled to allow selection of the desired range of rows.

The Column span parameter contains a corresponding set of three options for specifying the range of columns in u to be retained in submatrix y: All columns , One column , or Range of columns . The One column option enables the Column parameter, and Range of columns options enable the Starting column and Ending column parameters.

The output has the same frame status as the input.

Range Specification Options

When you select One row or Range of rows from the Row span parameter, you specify the desired row or range of rows in the Row parameter, or the Starting row and Ending row parameters. Similarly, when you select One column or Range of columns from the Column span parameter, you specify th e desired column or range of columns in the Column parameter, or the Starting column and Ending column parameters.

The Row , Column , Starting row or Starting column can be speciÞed in six ways:

- First

  For rows, this speci Þes that the Þrst row of u should be used as the Þrst row of y. When all columns are to be in cluded, this is equivalent to y(1,:) = u(1,:) .

  For columns, this speci Þes that the Þrst column of u should be used as the Þrst column of y. When all rows are to be included, this is equivalent to y(:,1) = u(:,1) .

- Index

  For rows, this speci Þes that the row of u, firstrow , forward-indexed by the Row index parameter or the Starting row index parameter, should be used as the Þrst row of y. When all columns are to be included, this is equivalent to y(1,:) = u(firstrow,:) .

  For columns, this speci Þes that the column of u, forward-indexed by the Column index parameter or the Starting column index parameter, firstcol , should be used as the Þrst column of y. When all rows are to be included, this is equivalent to y(:,1) = u(:,firstcol) .

- Offset from last

  For rows, this speci Þes that the row of u offset from row M by the Row offset or Starting row offset parameter, firstrow , should be

used as the first row of y. When all columns are to be included, this is equivalent to y(1,:) = u(M-firstrow,:) .

For columns, this specifies that the column of u offset from column N by the Column offset or Starting column offset parameter, firstcol , should be used as the first column of y. When all rows are to be included, this is equivalent to y(:,1) = u(:,N-firstcol) .

- Last

  For rows, this specifies that the last row of u should be used as the only row of y. When all columns are to be included, this is equivalent to y = u(M,:) .

  For columns, this specifies that the last column of u should be used as the only column of y. When all rows are to be included, this is equivalent to y = u(:,N) .

- Offset from middle

  For rows, this specifies that the row of u offset from row M/2 by the Starting row offset parameter, firstrow , should be used as the first row of y. When all columns are to be included, this is equivalent to y(1,:) = u(M/2-firstrow,:) .

  For columns, this specifies that the column of u offset from column N/2 by the Starting column offset parameter, firstcol , should be used as the first column of y. When all rows are to be included, this is equivalent to y(:,1) = u(:,N/2-firstcol) .

- Middle

  For rows, this specifies that the middle row of u should be used as the only row of y. When all columns are to be included, this is equivalent to y = u(M/2,:) .

  For columns, this specifies that the middle column of u should be used as the only column of y. When all rows are to be included, this is equivalent to y = u(:,N/2) .

The Ending row or Ending column can similarly be specified in five ways:

- Index

  For rows, this specifies that the row of u forward-indexed by the Ending row index parameter, lastrow, should be used as the last row of y. When all columns are to be included, this is equivalent to y(end,:) = u(lastrow,:).

  For columns, this specifies that the column of u forward-indexed by the Ending column index parameter, lastcol, should be used as the last column of y. When all rows are to be included, this is equivalent to y(:,end) = u(:,lastcol).

- Offset from last

  For rows, this specifies that the row of u offset from row M by the Ending row offset parameter, lastrow, should be used as the last row of y. When all columns are to be included, this is equivalent to y(end,:) = u(M-lastrow,:).

  For columns, this specifies that the column of u offset from column N by the Ending column offset parameter, lastcol, should be used as the last column of y. When all rows are to be included, this is equivalent to y(:,end) = u(:,N-lastcol).

- Last

  For rows, this specifies that the last row of u should be used as the last row of y. When all columns are to be included, this is equivalent to y(end,:) = u(M,:).

  For columns, this specifies that the last column of u should be used as the last column of y. When all rows are to be included, this is equivalent to y(:,end) = u(:,N).

- Offset from middle

  For rows, this specifies that the row of u offset from row M/2 by the Ending row offset parameter, lastrow, should be used as the last row of y. When all columns are to be included, this is equivalent to y(end,:) = u(M/2-lastrow,:).

For columns, this specifies that the column of u offset from column N/2 by the Ending column offset parameter, lastcol, should be used as the last column of y. When all rows are to be included, this is equivalent to y(:,end) = u(:,N/2-lastcol).

- Middle

  For rows, this specifies that the middle row of u should be used as the last row of y. When all columns are to be included, this is equivalent to y(end,:) = u(M/2,:).

  For columns, this specifies that the middle column of u should be used as the last column of y. When all rows are to be included, this is equivalent to y(:,end) = u(:,N/2).

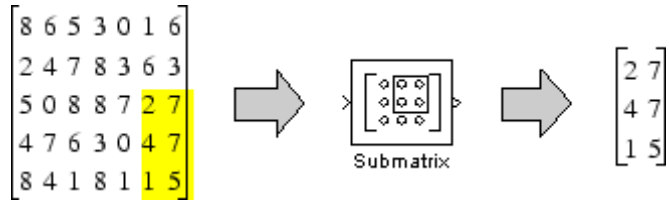
This block supports Simulink virtual buses.

Examples

To extract the lower-right 3-by-2 submatrix from a 5-by-7 input matrix, enter the following set of parameters:

- Row span = Range of rows
- Starting row = Index
- Starting row index = 3
- Ending row = Last
- Column span = Range of columns
- Starting column = Offset from last
- Starting column offset = 1
- Ending column = Last

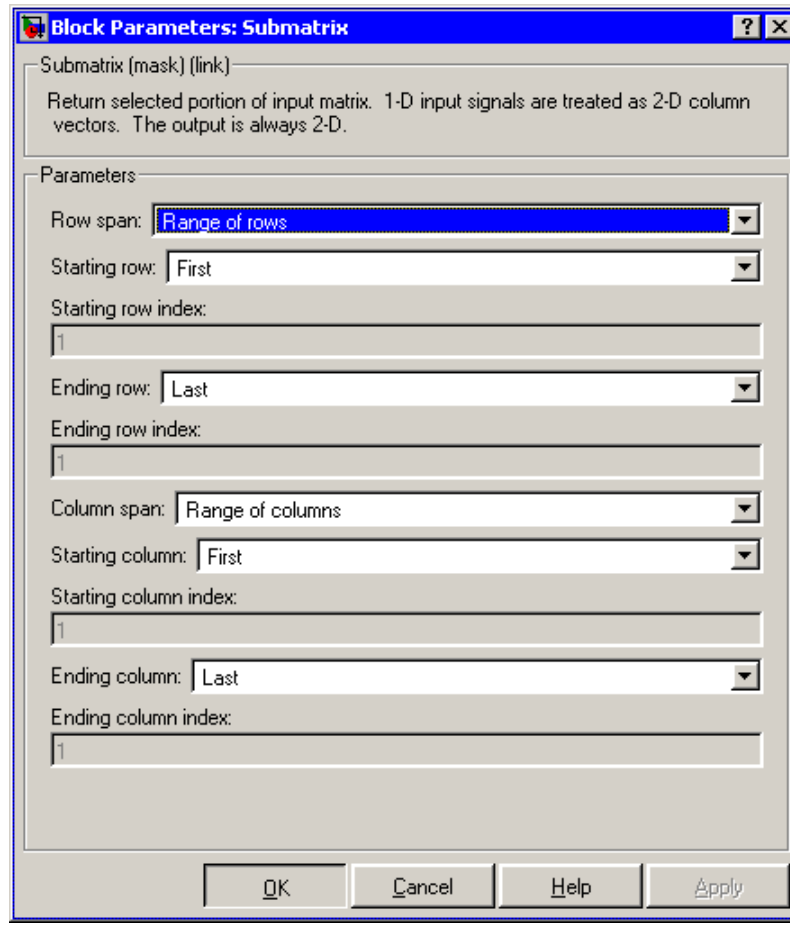
The figure below shows the operation for a 5-by-7 matrix with random integer elements, randint(5,7,10).



There are often several possible param eter combinations that select the same submatrix from the input. For example, instead of specifying Last for Ending column , you could select the same submatrix by specifying

- Ending column = Index
- Ending column index = 7

Dialog Box



The parameters displayed in the dialog box vary for different menu combinations. Only some of the para meters listed below are visible in the dialog box at any one time.

Row span

The range of input rows to be retained in the output. Options are All rows , One row, or Range of rows .

Row/Starting row

The input row to be used as the first row of the output. Row is enabled when you select One row from Row span, and Starting row when you select Range of rows from Row span.

Row index/Starting row index

The index of the input row to be used as the first row of the output. Row index is enabled when you select Index from Row, and Starting row index when you select Index from Starting row.

Row offset/Starting row offset

The offset of the input row to be used as the first row of the output. Row offset is enabled when you select Offset from middle or Offset from last from Row, and Starting row offset is enabled when you select Offset from middle or Offset from last from Starting row.

Ending row

The input row to be used as the last row of the output. This parameter is enabled when you select Range of rows from Row span and you select any option but Last from Starting row.

Ending row index

The index of the input row to be used as the last row of the output. This parameter is enabled when you select Index from Ending row.

Ending row offset

The offset of the input row to be used as the last row of the output. This parameter is enabled when you select Offset from middle or Offset from last from Ending row.

Column span

The range of input columns to be retained in the output. Options are All columns, One column, or Range of columns.

Column/Starting column

The input column to be used as the first column of the output. Column is enabled when you select One column from Column

span , and Starting column is enabled when you select Range of columns from Column span .

Column index/Starting column index
: The index of the input column to be used as the first column of the output. Column index is enabled when you select Index from Column, and Starting column index is enabled when you select Index from Starting column .

Column offset/Starting column offset
: The offset of the input column to be used as the first column of the output. Column offset is enabled when you select Offset from middle or Offset from last from Column. Starting column offset is enabled when you select Offset from middle or Offset from last from Starting column .

Ending column
: The input column to be used as the last column of the output. This parameter is enabled when you select Range of columns from Column span and you select any option but Last from Starting column .

Ending column index
: The index of the input column to be used as the last column of the output. This parameter is enabled when you select Index from Ending column .

Ending column offset
: The offset of the input column to be used as the last column of the output. This parameter is enabled when you select Offset from middle or Offset from last from Ending column .

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Reshape	Simulink
Selector	Simulink
Variable Selector	Signal Processing Blockset
reshape	MATLAB

See “Splitting Multichannel Sample-Based Signals into Several Multichannel Signals” for related information.

SVD Solver

Purpose

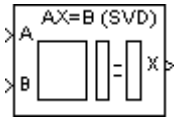
Solve AX=B using singular value decomposition

Library

Math Functions / Matrices and Linea r Algebra / Linear System Solvers

dspsolvers

Description

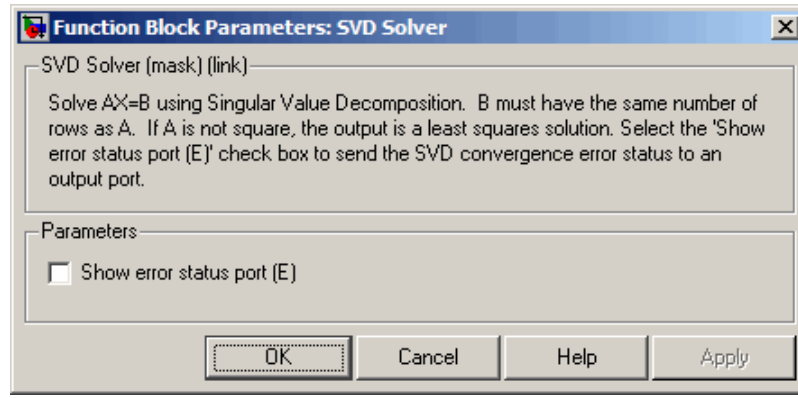


The SVD Solver block solves the linear system AX=B, which can be overdetermined, underdetermined , or exactly determined. The system is solved by applying singular value decomposition (SVD) factorization to the M-by-N matrix A, at the A port. The input to the B port is the right side M-by-L matrix, B. A length- M 1-D vector input at either port is treated as an M-by-1 matrix.

The output at the X port is the N-by-L matrix, X. X is always sample based, and is chosen to minimize the sum of the squares of the elements of B-AX (the residual). When B is a vector, this solution minimizes the vector 2-norm of the residual. When B is a matrix, this solution minimizes the matrix Frobenius nor m of the residual. In this case, the columns of X are the solutions to the L corresponding systems $AX_k=B_k$, where B_k is the kth column of B, and X_k is the kth column of X.

X is known as the minimum-norm-residual solution to AX=B. The minimum-norm-residual solutio n is unique for overdetermined and exactly determined linear systems, but it is not unique for underdetermined linear systems. Thus when the SVD Solver block is applied to an underdetermined system, the output X is chosen such that the number of nonzero entries in X is minimized.

Dialog Box



Show error status port

Select to enable the E output port, which reports a failure to converge. The possible values you can receive on the port are:

- 0 — The singular value decomposition calculation converges.
- 1 — The singular value decomposition calculation does not converge.

If the singular value decomposition calculation fails to converge, the output at port X is an unde Þned matrix of the correct size.

Supported Data Types

Port	Supported Data Types
A	• Double-precision ßoating point • Single-precision ßoating point
B	• Double-precision ßoating point • Single-precision ßoating point
X	• Double-precision ßoating point • Single-precision ßoating point
E	• Boolean

See Also

Autocorrelation LPC	Signal Processing Blockset
Cholesky Solver	Signal Processing Blockset
LDL Solver	Signal Processing Blockset
Levinson-Durbin	Signal Processing Blockset
LU Inverse	Signal Processing Blockset
Pseudoinverse	Signal Processing Blockset
QR Solver	Signal Processing Blockset
Singular Value Decomposition	Signal Processing Blockset

See “Linear System Solvers” for related information.

Purpose

Display signals generated during simulation

Library

Signal Processing Sinks

dspsnks4

Description

The Time Scope block is an implementation of the Simulink Scope block. See Scope for more information.

Time-Varying Direct-Form II Transpose Filter (Obsolete)

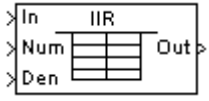
Purpose

Apply variable IIR filter to input

Library

dspobslib

Description



Note This block is now just an implementation of the Digital Filter block.

Purpose

Apply variable lattice Þlter to input

Library

dspobslib

Description

Note This block is now just an implementation of the Digital Filter block.

Toeplitz

Purpose

Generate matrix with Toeplitz symmetry

Library

Math Functions / Matrices and Linear Algebra / Matrix Operations

dspmtrx3

Description

The Toeplitz block generates a Toeplitz matrix from inputs defining the first column and first row. The top input (Col) is a vector containing the values to be placed in the first column of the matrix, and the bottom input (Row) is a vector containing the values to be placed in the first row of the matrix.

```
y = toeplitz(Col,Row)          % Equivalent MATLAB code
```

The other elements of the matrix obey the relationship

```
y(i,j) = y(i-1,j-1)
```

and the output has dimension [length(Col) length(Row)]. The y(1,1) element is inherited from the Col input. For example, the following inputs

```
Col = [1 2 3 4 5]
Row = [7 7 3 3 2 1 3]
```

produce the Toeplitz matrix

$$\begin{bmatrix} 1 & 7 & 3 & 3 & 2 & 1 & 3 \\ 2 & 1 & 7 & 3 & 3 & 2 & 1 \\ 3 & 2 & 1 & 7 & 3 & 3 & 2 \\ 4 & 3 & 2 & 1 & 7 & 3 & 3 \\ 5 & 4 & 3 & 2 & 1 & 7 & 3 \end{bmatrix}$$

When both of the inputs are sample based, the output is sample based. Otherwise, the output is frame based.

When you select the Symmetric check box, the block generates a symmetric (Hermitian) Toeplitz matrix from a single input, u, deÞning both the Þrst row and Þrst column of the matrix.

```
y = toeplitz(u)          % Equivalent MATLAB code
```

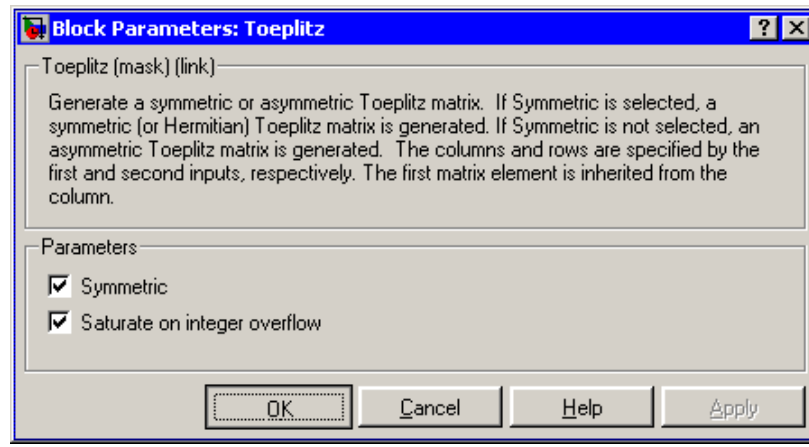
The output has dimension [length(u) length(u)] . For example, the Toeplitz matrix generated from the input vector [1 2 3 4] is

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

The output has the same frame status as the input.

The Toeplitz block supports real and complex ßoating-point and Þxed-point inputs.

Dialog Box



Symmetric

When selected, enables the single-input conÞguration for symmetric Toeplitz matrix output.

Saturate on integer overflow

When you generate a symmetric Toeplitz matrix with this block, if the input vector is complex, the output is a symmetric Hermitian matrix whose elements satisfy the relationship

$$y(i, j) = \mathrm{conj}(y(j, i))$$

For fixed-point signals the conjugate operation could result in an overflow. When you select this parameter, overflows saturate. This parameter is only visible with the Symmetric parameter is selected. This parameter is ignored for floating-point signals.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers (real signals only)
Toep Col	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Port	Supported Data Types
Toep Row	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Constant Diagonal Matrix — Signal Processing Blockset

toeplitz — MATLAB

To Audio Device

Purpose

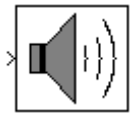
Write audio data to computer's audio device

Library

Signal Processing Sinks

dspsnks4

Description



The To Audio Device block sends audio data to your computer's audio device. This block has the following limitations:

- Not supported for use with the Simulink Model block.
- Not currently supported on Solaris platforms.

Use the Device parameter to specify the device to which you want to send the audio data. This parameter is automatically populated based on the audio devices installed on your system. If you plug or unplug an audio device from your system, type clear mex at the MATLAB command prompt to update the list.

Select the Inherit sample rate from input check box if you want the block to inherit the sample rate of the audio signal from the input to the block. If you clear this check box, the Sample rate (Hz) parameter appears on the block. Use this parameter to specify the number of samples per second in the signal.

Use the Device data type to specify the data type of the audio data that is sent to the device. You can choose:

- 8-bit integer
- 16-bit integer
- 24-bit integer
- 32-bit float
- Determine from input data type

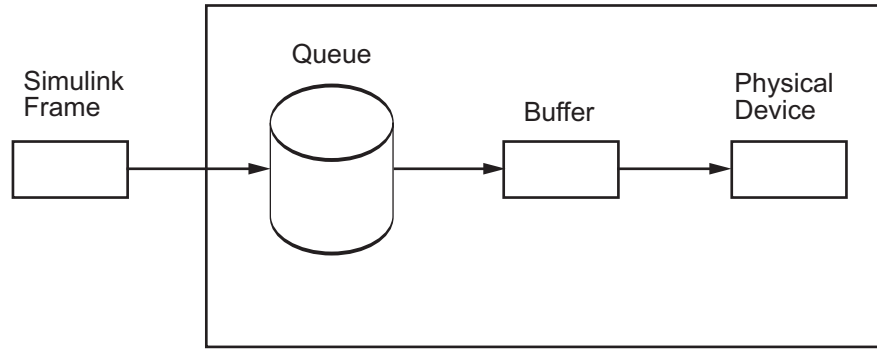
If you choose Determine from input data type , the following table summarizes the block's behavior.

Input Data Type	Device Data Type
Double-precision ßoating point or single-precision ßoating point	32-bit ßoating point
32-bit integer	24-bit integer
16-bit integer	16-bit integer
8-bit integer	8-bit integer

If you choose Determine from input data type and the device does not support the input data type, the block uses the next lowest-precision data type supported by the device.

Buffering

The To Audio Device block buffers the data from a Simulink signal using the process illustrated by the following Þgure.



To Audio Device Block

1 At the start of the simulation, the queue is Þlled with silence. Specify the size of this queue using the Queue duration (seconds) parameter. As Simulink runs, the block appends Simulink frames to the bottom of the queue.

2 At each time step, the blocks sends a buffer of samples from the top of the queue to the audio device. Select the Automatically determine

buffer size check box to allow the block to use a conservative buffer size. See the From Audio Device block reference page for the equation the block uses to calculate this buffer size. If you clear this check box, the Buffer size (samples) parameter appears on the block. Use this parameter to specify the size of the buffer in samples.

3 The block writes the buffer of audio data to the device. If the queue did not contain enough data to completely Þll the buffer, the block Þlls the remaining portion of the buffer with zeros. This data has a the data type speci Þed by the Device data type parameter.

When the simulation throughput rate is lower than the hardware throughput rate, the queue, which is initially full, becomes empty. If the queue is empty, the block sends zeros (silence) to the audio device. When the simulation throughput rate is higher than the hardware throughput rate, the To Audio Device block waits to write data to the queue.

To minimize the chance of dropouts, the block checks to make sure the queue duration is at least as large as the maximum of the buffer size and the frame size. If it is not, the q ueue duration is automatically set to this maximum value.

Troubleshooting

When Simulink cannot keep up with an audio device that is operating in real time, the queue becomes empty and gaps occur in the audio data that the block sends to the devi ce. Here are several ways to deal with this situation:

- Increase the queue duration .

  The Queue duration (seconds) parameter speci Þes the duration of the signal, in seconds, that can be buffered during the simulation. This is the maximum length of time that the block's data supply can lag the hardware's data demand.

- Increase the buffer size.

  The size of the buffer processed in each interrupt from the audio device affects the performance of your model. If the buffer is too

small, a large portion of hardware resources are used to write data to the device. If the buffer is too big, Simulink must wait for the device to empty the buffer before it can write the data to the queue, which introduces latency.

- Increase the simulation throughput rate .

  Two useful methods for improving simulation throughput rates are increasing the signal frame size and compiling the simulation into native code:

  - Increase frame sizes and conv ert sample-based signals to frame-based signals througho ut the model to reduce the amount of block-to-block communicati on overhead. This can increase throughput rates in many cases. However, larger frame sizes generally result in greater model l atency due to initial buffering operations.
  - Generate executable code wit h Real-Time Workshop code generation software. Native code runs much faster than Simulink and should provide rates adequate for real-time audio processing.

Other ways to improve throughput ra tes include simplifying the model and running the simulation on a faster PC processor. For other ideas on improving simulation performance, see “Delay and Latency” and “Improving Simulation Performance and Accuracy” in the Simulink documentation.

Channel-to-Speaker Mapping on Windows Operating Systems

The To Audio Device and From Audio Device blocks can support multiple channels. On Windows operating s ystems, the channel-to-speaker mapping is de Þned as listed below. This mapping only applies when your sound card is properly con Þgured and capable of receiving the audio data you send. If the number of channels on the card does not match the number of channels on the b lock, or if you specify a data type for the Device data type parameter that is not supported by your device, the Windows mixer interven es to translate from one format to

another. If the Windows mixer does intervene, the channel-to-speaker mapping might differ from what is speci Þed here.

- Single channel input — Front center speaker

  On systems with two speakers, the front center channel is split between the right and left speakers.

- Multichannel input — Channels are assigned to speakers as follows:
  - One channel — Front center
  - Two channels — Front left, front right
  - Four channels — Front left, fro nt right, rear left, rear right
  - Six channels — Front left, front righ t, front center, low frequency, rear left, rear right
  - Eight channels — Front left, front right, front center, low frequency, rear left, rear right, front left center, front right center
  - For all other channel combinations, the channel assignment is dictated by the audio card.

Audio Hardware API

The To Audio Device and From Audio Device blocks use the open-source PortAudio library in order to comm unicate with the audio hardware on a given computer. The PortAudio library supports a range of API's designed to communicate with the aud io hardware on a given platform. The following API choices were made when building the PortAudio library for the Signal Processing Blockset product:

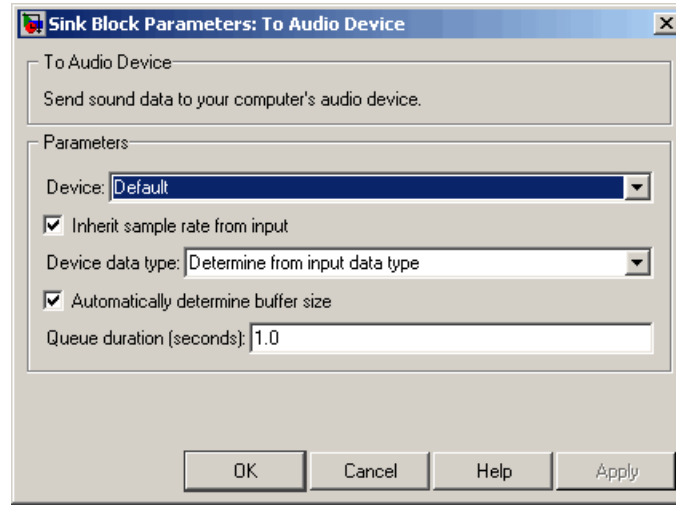
- Windows: DirectSound
- Linux: OSS
- Mac: CoreAudio

If you are interested in using a different audio API, such as ASIO (Windows) or ALSA (Linux) please se arch for PortAudio on the Matlab Central website.

Example

See the Positional Audio demo for an example of how to use this block. You can open this demo by typing `dspAudioPos` at the MATLAB command line.

Dialog Box



Device
: Specify which device to send the audio data to.

Inherit sample rate from input
: Select this check box if you want the block to inherit the sample rate of the audio signal fr om the input to the block.

Sample rate (Hz)
: Specify the number of samples per second in the signal. This parameter is visible when the Inherit sample rate from input check box is cleared.

Device data type
: Specify the data type of the audio data sent to the device.

Automatically determine buffer size
: Select this check box to allow the bl ock to calculate a conservative buffer size.

Buffer size (samples)

Specify the size of the buffer. This parameter is visible when the Automatically determine buffer size check box is cleared.

Queue duration (seconds)

Specify the size of the queue in seconds.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • 32-bit signed integers • 16-bit signed integers • 8-bit unsigned integers

See Also

From Audio Device	Signa l Processing Blockset
To Wave File	Signal Processing Blockset
audioplayer	MATLAB
sound	MATLAB

Purpose

Write video frames and/or audio samples to multimedia file

Library

Signal Processing Sinks

dspsnks4

Description

The To Multimedia File block writes video frames and/or audio samples to a multimedia (.avi) file. Video processing requires the Video and Image Processing Blockset product.

You can also compress the video frames or audio samples by selecting a compression algorithm. You can connect as many of the input ports as you want. Therefore, you can control what type of video and/or audio is sent to the multimedia file.

Note This block supports code generation for platforms that have file I/O available. This excludes RTWin (Real-Time Windows Target software, which does not support file I/O). This block performs best on platforms with Version 9.0 or later of DirectX software and Version 9.0 or later of Windows Media software. On UNIX and Linux platforms, this block supports only uncompressed RGB24 AVI files whose size is less than 2 GB.

Port	Input	Supported Data Types	Supports Complex Values?
Image	M-by-N-by-3 matrix RGB signal. To record M-by-N intensity video, use the Matrix Concatenate block to create an RGB signal.	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16- 32-bit signed integers	No

Port	Input	Supported Data Types	Supports Complex Values?
		• 8-, 16- 32-bit unsigned integers	
R, G, B	Matrix that represents one plane of the RGB video stream. Inputs to the R, G, or B port must have the same dimensions and data type.	Same as Image port	No
Audio	Vector of audio data	• Double-precision ßoating point • Single-precision ßoating point • 16-bit signed integers • 8-bit unsigned integers	No

For the block to display video data properly, double- and single-precision ßoating-point pixel values must be between 0 and 1. For any other data type, the pixel values must be between the minimum and maximum values supported by their data type.

Use the Output file name parameter to specify the name of the multimedia Þle to which to write. This Þle is saved in your current directory. To specify a different Þle, click the Browse button, and then navigate to the new Þle.

Use the Write parameter to specify whether the block writes video frames and/or audio samples to the multimedia Þle. The choices are Video and audio , Video only , or Audio only .

Use the Audio compressor parameter to specify the type of compression algorithm to use to compress the audio data. This compression reduces the size of the multimedia Þle. Choose None

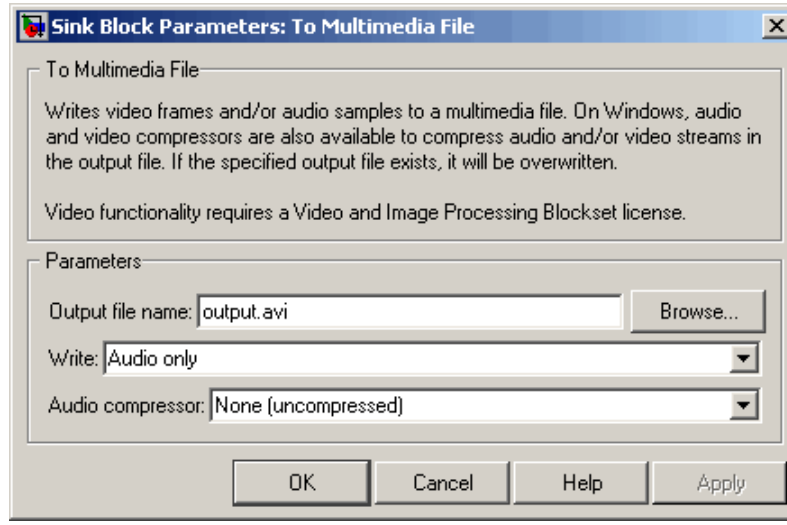
(uncompressed) to save uncompressed audio data to the multimedia Þle. The other items available in this parameter list are the audio compression algorithms installed on your system. For information about a speciÞc audio compressor, see its documentation.

Use the Video compressor parameter to specify the type of compression algorithm to use to compress the video data. This compression reduces the size of the multimedia Þle. Choose None (uncompressed) to save uncompressed video data to the multimedia Þle. The other items available in this parameter list are the video compression algorithms installed on your system. For information about a speciÞc video compressor, see its documentation.

Use the Image signal parameter to specify how the block accepts a color video signal. If you select One multidimensional signal , the block accepts an M-by-N-by-3 color video signal. To record M-by-N intensity video, use the Matrix Concatenate block to create an RGB signal. If you select Separate color signals , additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

Note All the To Multimedia File block input signals must have the same frame period. You might need to adjust the frame size of the audio signal so that the frame period of the video signal is the same as the frame period of the audio signal. To calculate the frame size, divide the frequency of the audio signal (samples per second) by the frame rate of the video signal (frames per second).

Dialog Box



Output file name

Specify the name of the multimedia file to which to write. This file is saved in your current directory. To specify a different file, click the Browse button, and then navigate to the new file.

Write

Specify whether the block writes video frames and/or audio samples to the multimedia file. The choices are Video and audio, Video only, or Audio only.

Audio compressor

Select the type of compression algorithm to use to compress the audio data.

Video compressor

Select the type of compression algorithm to use to compress the video data.

Image signal

Specify how the block accepts a color video signal. If you select One multidimensional signal, the block accepts an M-by-N-by-P

color video signal, where P is the number of color planes, at one port. If you select Separate color signals , additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

See Also

From Multimedia File	Signal Processing Blockset
To Wave File	Signal Processing Blockset
Frame Rate Display	Video and Image Processing Blockset
To Video Display	Video and Image Processing Blockset
Video To Workspace	Video and Image Processing Blockset
Video Viewer	Video and Image Processing Blockset

To Wave Device (Obsolete)

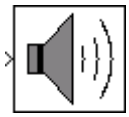
Purpose

Send audio data to standard Windows audio device in real time

Library

dspwin32

Description



Note The To Wave Device block is still supported but is likely to be obsoleted in a future release. We st rongly recommend replacing this block with the To Audio Device block.

The To Wave Device block sends audio data to a standard Windows audio device in real time. It is compatible with most popular Windows hardware, including Sound Blaster cards. The data is sent to the hardware in uncompressed pulse code modulation (PCM) format, and should typically be sampled at one of the standard Windows audio device rates: 8000, 11025, 22050, or 44100 Hz. Some hardware might support other rates in addition to these.

Note Models that contain both the To Wave Device block and the From Wave Device block require a d uplex-capable sound card.

The Use default audio device check box allows the To Wave Device block to detect and use the system's default audio hardware. You should select this option for systems that ha ve a single sound device installed, or when the default sound device on a multiple-device system is your desired target. When the default sound device is not your desired output device, clear Use default audio device , and set the desired hardware in the Audio device parameter. This parameter lists the names of the installed audio devices.

The block input can contain audio data from a mono or stereo signal. A mono signal is represented as either a sample-based scalar or a frame-based length- M vector, where M is frame size. A stereo signal is represented as a sample-based length-2 vector or a frame-based M -by-2 matrix.

When the input data type is uint8, the block conveys the signal samples to the audio device using 8 bits. When the input data type is double, single, int16, or fixed point with a word length of 16 and a fraction length of 15, the block conveys the signal samples to the audio device using 16 bits by default. For inputs of data type double and single, you can also set the block to convey the signal samples using 24 bits by selecting the Enable 24-bit output for double- and single-precision input signals check box. The 24-bit sample width requires more memory but in general yields better fidelity.

The amplitude of the input must be in a valid range that depends on the input data type, as shown in the following table. Amplitudes outside the valid range are clipped to the nearest allowable value.

Input Data Type	Valid Input Amplitude Range
double	$-1 \le amplitude < 1$
single	$-1 \le amplitude < 1$
int16	$-32768 \le amplitude \le 32767$
uint8	$0 \le amplitude \le 255$
Fixed point with a word length of 16 and a fraction length of 15	$-1 \le amplitude \le 1 - 2^{-15}$

Buffering

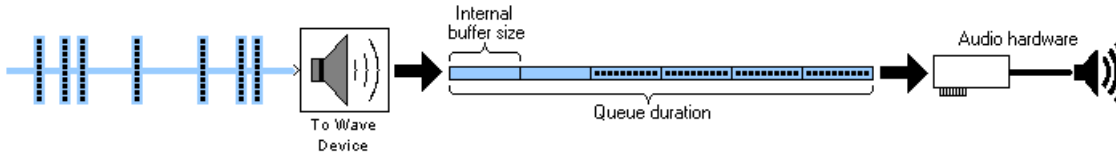
Because audio devices generate real-time audio output, the Simulink environment must maintain a continuous flow of data to a device throughout simulation. Delays in passing data to the audio hardware can result in hardware errors or distortion of the output. This means that the To Wave Device block must in principle supply data to the

audio hardware as quickly as the hardware reads the data. However, the To Wave Device block often cannot match the throughput rate of the audio hardware, especially when the simulation is running within Simulink rather than as generated code. Simulink execution speed can vary during the simulation as the host operating system services other processes. The block must therefore rely on a buffering strategy to ensure that signal data is available to the hardware on demand.

Note This block requires real-time execution of the parent model for best performance.

The following block parameters control the memory management for this block:

- Queue duration
- Automatically determine internal buffer size or User-defined internal buffer size
- Initial output delay



The Queue duration parameter defines the overall size of the block's buffer. The block reads in chunks of data in the size of the input dimensions and stores them in the buffer. The internal buffer size defines the dimensions of the block output to the hardware. You can define the internal buffer size yourself in the User-defined internal buffer size parameter. If you select Automatically determine internal buffer size instead, the internal buffer size is calculated for you according to the following rules:

- If the input to the block has a frame size of 32 samples or larger, the internal buffer size be the same as the input frame size.
- If the input to the block has a frame size smaller than 32 samples, the internal buffer size is based on the input sample rate according to the following table, where

$$F_s = sampling\ frequency = \frac{1}{sample\ time}$$

F_s (Hz)	Internal Buffer Size (samples)
$F_s < 8000$	$\min(64, 2 * F_s)$
$8000 \le F_s < 22,050$	128
$22,050 \le F_s < 44,100$	256
$44,100 \le F_s < 96,000$	512
$F_s \ge 96,000$	1024

To minimize the chance of dropouts, the block checks to make sure that the queue duration is at least as big as twice the internal buffer size. If it is not, the queue duration is automatically set to twice the internal buffer size.

The Initial output delay parameter enables you to preload the buffer before the block starts to output data to the audio device, which can be helpful for models that do not run in real time. However, for real-time applications, it is best to set the initial output delay to zero (one frame of delay), or as close to zero as possible.

Troubleshooting

If you are getting undesirable audio output using the To Wave Device block, Þrst determine whether your model can run in real time. Replace the To Wave Device block with a To Wave File block, run the model, and compare the model's simulation stop time to the elapsed time on your watch. If the model simulation stop time is less than the elapsed time on your watch, your model can probably run in real time. Then,

- If your model can run in real time,
  1. Select Automatically determine internal buffer size . This alone might solve the problem. If not,
  2. Try increasing the Queue duration parameter to a relatively large value, such as 0.5 s.

If one or both of these options res tores desirable audio output, you can try reducing the internal buffer size and/or queue duration until the quality of the audio output again degrades.

- If your model is not running in real time, try to make it run in real time by

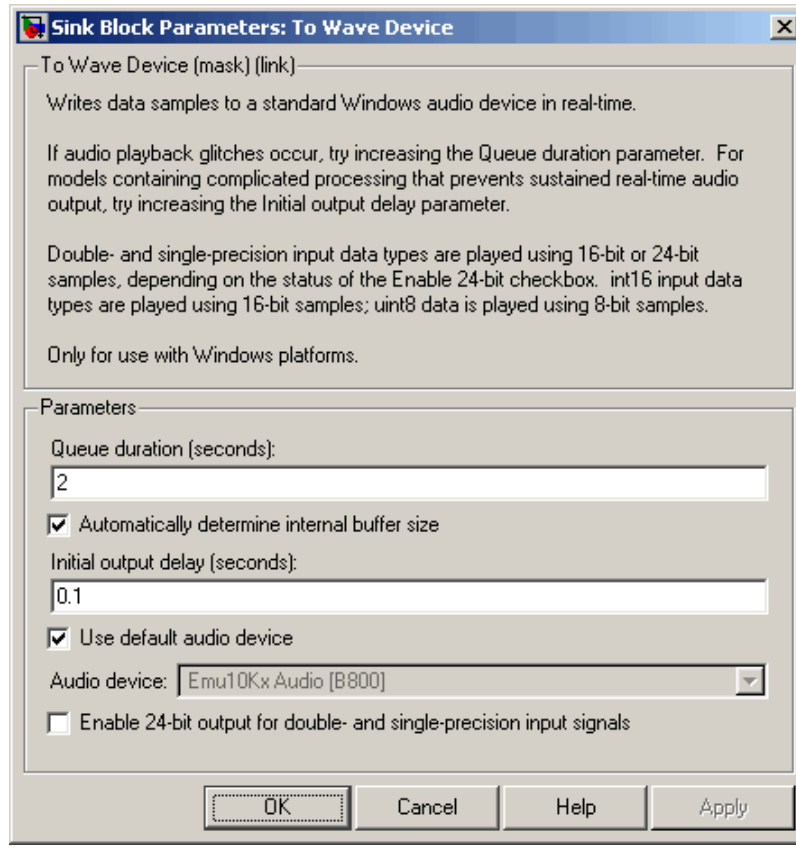
  1 Optimizing the model (using a more ef Þcient implementation), or

  2 Using the Simulink Accelerator mode, or

  3 Generating stand-alone code

  If none of these are possible, but the model only runs for a short period of time, set the Queue duration parameter to a size equal to a signi Þcant fraction of the model stop time and use a similarly large initial delay. This is not an optimal solution, but might work in some cases.

Dialog Box



Queue duration (seconds)

Specify the overall buffer size. To minimize the chance of dropouts, the block checks to make sure that the queue duration is as least as large as twice the internal buffer size. If it is not, the queue duration is automatically set to twice the internal buffer size.

Automatically determine internal buffer size

Select to have the block automati cally select the internal buffer size for you. For details, see “Buffering” on page 2-1261.

User-defined internal buffer size (samples)

Define the internal buffer size, or the size of the chunks of data sent by the block to the audio hardware device.

This parameter is only visible when Automatically determine internal buffer size is not selected.

Initial output delay (seconds)

Specify the amount of time by which to delay the initial output to the audio device. During this time data accumulates in the block's buffer. Any value less than or equal to the queue duration specifies the smallest possible initial delay, which is a single frame.

Use default audio device

Select to direct audio output to the system's default audio device.

Audio device

This parameter lists the names of the installed audio devices. Specify the name of the audio device to receive the audio output. Select Use default audio device when the system has only a single audio card installed.

This parameter is only enabled when the Use default audio device check box is not selected.

Enable 24-bit output for double and single precision input signals

Select to output 24-bit data when inputs are double- or single-precision. Otherwise, the block outputs 16-bit data for double- and single-precision inputs.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Signed Þxed point with a word length of 16 and a fraction length of 15 • 16-bit signed integers • 8-bit unsigned integers

See Also

From Wave Device (Obsolete)	Signal Processing Blockset
To Wave File	Signal Processing Blockset
audioplayer	MATLAB
sound	MATLAB

Purpose

Write audio data to Þle in Microsoft Wave (.wav) format

Library

Signal Processing Sinks

dspsnks4

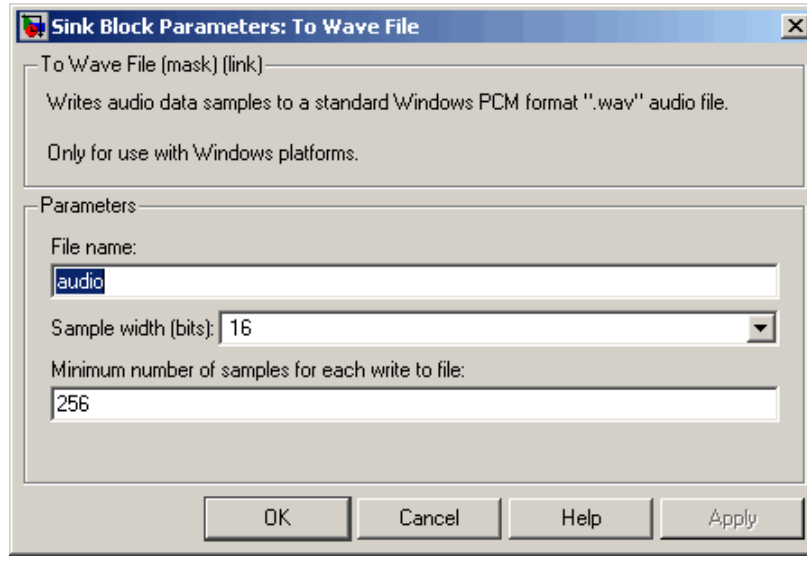
Description

The To Wave File block streams audio data to a Microsoft Wave (.wav) Þle in the uncompressed pulse code modulation (PCM) format. For compatibility reasons, the sample rate of the discrete-time input signal should typically be one of th e standard Windows audio device rates (8000, 11025, 22050, or 44100 Hz), although the block supports arbitrary rates.

The input to the block, u, can contain audio data with one or more channels. A signal with C channels is represented as a sample-based length- C vector or a frame-based M-by-C matrix. The amplitude of the input should be in the range ±1. Values outside this range are clipped to the nearest allowable value.

```
wavwrite(u,Fs,bits,'filename')      % Equivalent MATLAB code
```

Dialog Box



File name

Specify the path and name of the Þle to write. Paths can be relative or absolute. You do not need to specify the .wav extension.

Sample width (bits)

Specify the number of bits used to represent the signal samples in the Þle. The higher sample width settings require more memory but yield better Þdelity for double- and single-precision inputs:

- 8 — Allocates 8 bits to each sample, allowing a resolution of 256 levels
- 16 — Allocates 16 bits to each sample, allowing a resolution of 65536 levels
- 24 — Allocates 24 bits to each sample, allowing a resolution of 16777216 levels
- 32 — Allocates 32 bits to each sample, allowing a resolution of 2^{32} levels ranging from -1 to 1

The 8-, 16-, and 24-bit modes output integer data, while the 32-bit mode outputs single-precision ßoating-point data.

Minimum number of samples for each write to ßle

Specify the number of consecutive samples, L, to write with each Þle access. To reduce the required number of Þle accesses, the block writes L consecutive samples to the Þle during each access for $L \geq M$. For $L < M$, the block instead writes M consecutive samples during each access. Larger values of L result in fewer Þle accesses, which reduces run-time overhead.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Signed Þxed point with a word length of 16 and a fraction length of 15 • 16-bit signed integers • 8-bit unsigned integers

See Also

From Wave File	Signal Processing Blockset
To Audio Device	Signal Processing Blockset
To Workspace	Simulink
wavwrite	MATLAB

Transpose

Purpose

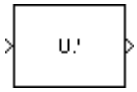
Compute matrix transpose

Library

Math Functions / Matrices and Lin ear Algebra / Matrix Operations

dspmtrx3

Description



The Transpose block transposes the M-by-N input matrix to size N-by-M. When you select the Hermitian check box, the block performs the Hermitian (complex conjugate) transpose.

```
y = u'      % Equivalent MATLAB code
```

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \end{bmatrix} \xrightarrow{u'} \begin{bmatrix} u_{11}^* & u_{21}^* \\ u_{12}^* & u_{22}^* \\ u_{13}^* & u_{23}^* \end{bmatrix}$$

When you do not select the Hermitian check box, the block performs the nonconjugate transpose.

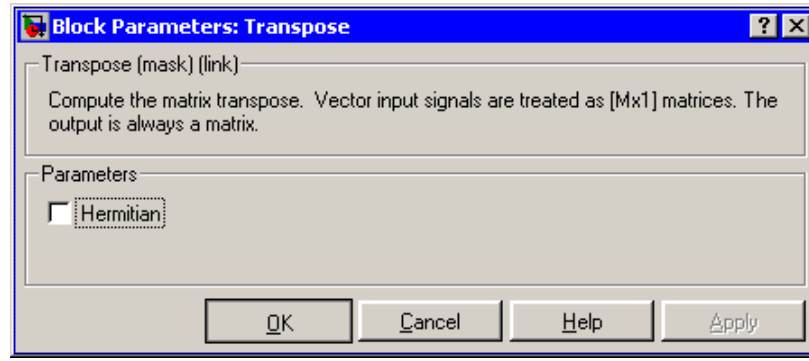
```
y = u.'     % Equivalent MATLAB code
```

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \end{bmatrix} \xrightarrow{u.'} \begin{bmatrix} u_{11} & u_{21} \\ u_{12} & u_{22} \\ u_{13} & u_{23} \end{bmatrix}$$

A length-M 1-D vector input is treated as an M-by-1 matrix. The output is always sample based.

The Transpose block supports real and complex ßoating-point and Þxed-point data types. When Hermitian is selected, the block input must be a signed data type.

Dialog Box



Hermitian

When selected, specifies the complex conjugate transpose.

Saturate on integer overflow

This parameter is only visible when the Hermitian parameter is selected because overflows can occur when computing the complex conjugate of complex fixed-point signals. When you select this parameter, such overflows saturate. This parameter is ignored for floating-point signals and for real-valued fixed-point signals.

Supported Data Types

When Hermitian is selected, the block input must be a signed data type.

Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers

Port	Supported Data Types
	• 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Math Function	Simulink
Permute Matrix	Signal Processing Blockset
Reshape	Simulink
Submatrix	Signal Processing Blockset

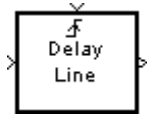
Purpose

Buffer sequence of inputs into frame-based output

Library

dspobslib

Description



Note The Triggered Delay Line block is still supported but is likely to be obsoleted in a future release. We strongly recommend replacing this block with the Delay Line block.

The Triggered Delay Line block acquires a collection of M_o input samples into a frame, where you specify M_o in the Delay line size parameter. The block buffers a single sample from input 1 whenever it is triggered by the control signal at input 2 ( ). When the next triggering event occurs, the newly acquired input sample is appended to the output frame so that the new output overlaps the previous output by M_o-1 samples. Between triggering events the block ignores input 1 and holds the output at its last value.

You specify the triggering event at input 2 in the Trigger type pop-up menu:

- Rising edge triggers execution of the block when the trigger input rises from a negative value to zero or a positive value, or from zero to a positive value.
- Falling edge triggers execution of the block when the trigger input falls from a positive value to zero or a negative value, or from zero to a negative value.
- Either edge triggers execution of the block when either a rising or falling edge (as described above) occurs.

The Triggered Delay Line block has zero latency, so the new input appears at the output in the same simulation time step. The output frame period is the same as the input sample period, $T_{fo}=T_{si}$.

Sample-Based Operation

In sample-based operation, the Triggered Delay Line block buffers a sequence of sample-based length-N vector inputs (1-D, row, or column) into a sequence of overlapping sample-based M_o-by-N matrix outputs, where you specify M_o in the Delay line size parameter (M_o>1). That is, each input vector becomes a row in the sample-based output matrix. When M_o=1, the input is simply passed through to the output, and retains the same dimension. Sample-based full-dimension matrix inputs are not accepted.

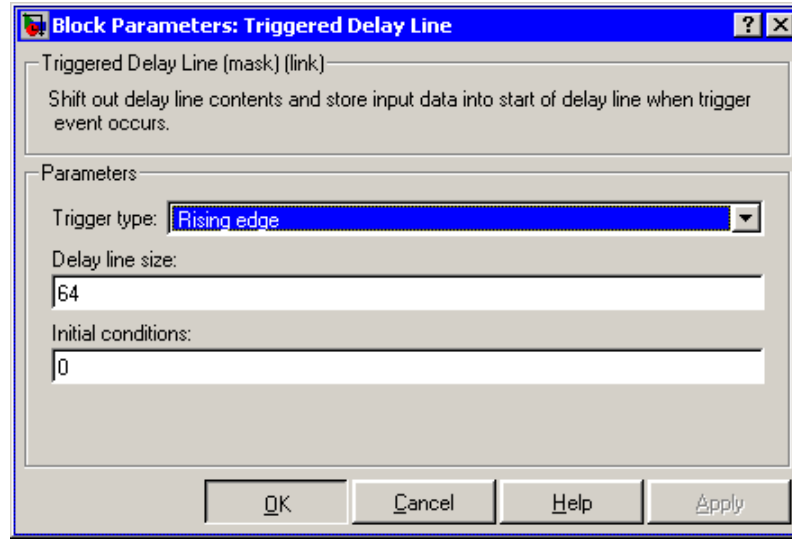
Frame-Based Operation

In frame-based operation, the Triggered Delay Line block rebuffers a sequence of frame-based M_i-by-N matrix inputs into an sequence of overlapping frame-based M_o-by-N matrix outputs, where M_o is the output frame size specified by the Delay line size parameter (that is, the number of consecutive samples from the input frame to rebuffer into the output frame). M_o can be greater or less than the input frame size, M_i. Each of the N input channels is rebuffered independently.

Initial Conditions

The Triggered Delay Line block's buffer is initialized to the value specified by the Initial condition parameter. The block always outputs this buffer at the first simulation step (t=0). When the block's output is a vector, the Initial condition can be a vector of the same size or a scalar value to be repeated across all elements of the initial output. When the block's output is a matrix, the Initial condition can be a matrix of the same size or a scalar to be repeated across all elements of the initial output.

Dialog Box



Trigger type
: The type of event that triggers the block's execution.

Delay line size
: The length of the output frame (number of rows in output matrix), M_o.

Initial condition
: The value of the block's initial output, a scalar, vector, or matrix.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Trigger	• Any data type supported by the Trigger block
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Buffer	Signal Processing Blockset
Delay Line	Signal Processing Blockset
Unbuffer	Signal Processing Blockset

Purpose

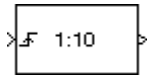
Import signal samples from MATLAB workspace when triggered

Library

Signal Operations

dspsigops

Description



The Triggered Signal From Workspace block imports signal samples from the MATLAB workspace into the Simulink model when triggered by the control signal at the input port ( ). The Signal parameter specifies the name of a MATLAB workspace variable containing the signal to import, or any valid MATLAB expression defining a matrix or 3-D array.

When the Signal parameter specifies an M-by-N matrix (M ≠1), each of the N columns is treated as a distinct channel. You specify the frame size in the Samples per frame parameter, M_o, and the output when triggered is an M_o-by-N matrix containing M_o consecutive samples from each signal channel. For M_o=1, the output is sample based; otherwise the output is frame based. For convenience, an imported row vector (M=1) is treated as a single channel, so the output dimension is M_o-by-1.

When the Signal parameter specifies an M-by-N-by-P array, the block generates a single page of the array (an M-by-N matrix) at each trigger time. The Samples per frame parameter must be set to 1, and the output is always sample based.

Trigger Event

You specify the triggering event at the input port in the Trigger type pop-up menu:

- Rising edge triggers execution of the block when the trigger input rises from a negative value to zero or a positive value, or from zero to a positive value.
- Falling edge triggers execution of the block when the trigger input falls from a positive value to zero or a negative value, or from zero to a negative value.

- Either edge triggers execution of the block when either a rising or falling edge (as described above) occurs.

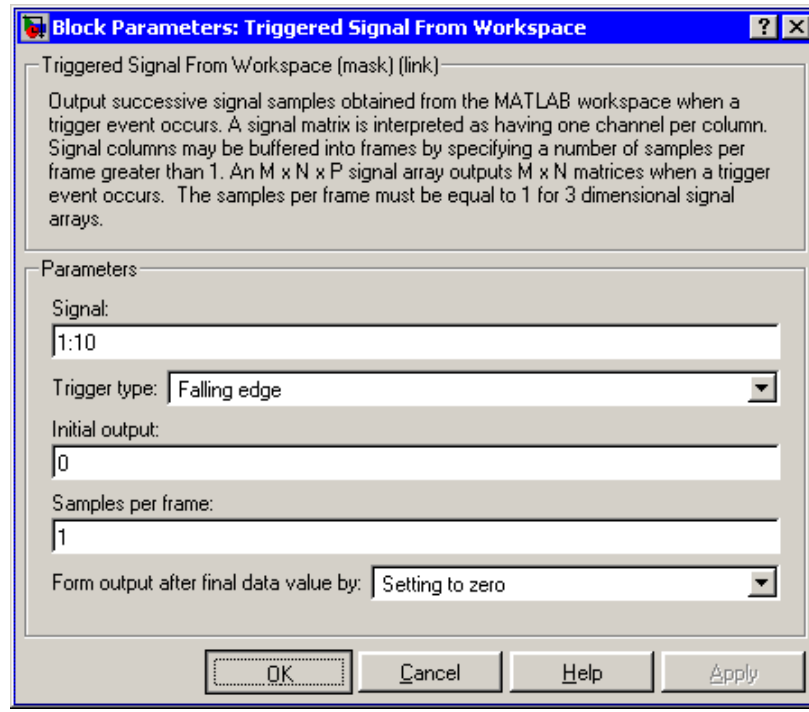
Initial and Final Conditions

The Initial output parameter specifies the output of the block from the start of the simulation until the first trigger event arrives. Between trigger events, the block holds the output value constant at its most recent value (that is, no linear interpolation takes place). For single-channel signals, the Initial output parameter value can be a vector of length M_o or a scalar to repeat across the M_o elements of the initial output frames. For matrix outputs (M_o-by-N or M-by-N), the Initial output parameter value can be a matrix of the same size or a scalar to be repeated across all elements of the initial output.

When the block has output all of the available signal samples, it can start again at the beginning of the signal, or simply repeat the final value or generate zeros until the end of the simulation. (The block does not extrapolate the imported signal beyond the last sample.) The Form output after final data value by parameter controls this behavior:

- When you specify Setting To Zero, the block generates zero-valued outputs for the duration of the simulation after generating the last frame of the signal.
- When you specify Holding Final Value, the block repeats the final sample for the duration of the simulation after generating the last frame of the signal.
- When you specify Cyclic Repetition, the block repeats the signal from the beginning after generating the last frame. When there are not enough samples at the end of the signal to fill the final frame, the block zero-pads the final frame as necessary to ensure that the output for each cycle is identical (for example, the ith frame of one cycle contains the same samples as the ith frame of any other cycle).

Dialog Box



Signal

The name of the MATLAB workspace variable from which to import the signal, or a valid MATLAB expression specifying the signal.

Trigger type

The type of event that triggers the block's execution.

Initial output

The value to output until the first trigger event is received.

Samples per frame

The number of samples, M_o, to buffer into each output frame. This value must be 1 when you specify a 3-D array in the Signal parameter.

Form output after final data value by
: Specifies the output after all of the specified signal samples have been generated. The block can output zeros for the duration of the simulation (`Setting to zero`), repeat the final data sample (`Holding Final Value`) or repeat the entire signal from the beginning (`Cyclic Repetition`).

Supported Data Types

- Double-precision floating point
- Single-precision floating point
- Fixed point (signed and unsigned)
- 8-, 16-, and 32-bit signed integers
- 8-, 16-, and 32-bit unsigned integers

See Also

Signal From Workspace	Signal Processing Blockset
Signal To Workspace	Signal Processing Blockset
Triggered To Workspace	Signal Processing Blockset

Purpose

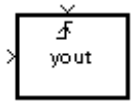
Write input sample to MATLAB workspace when triggered

Library

Signal Processing Sinks

dspsnks4

Description



The Triggered To Workspace block creates a matrix or array variable in the MATLAB workspace, where it stores the acquired inputs at the end of a simulation. The block overwrites an existing variable with the same name.

For an M-by-N frame-based input, the block creates an N-column workspace matrix in which each group of M rows represents a single input frame from each of N channels (the most recent frame occupying the last M rows). The maximum size of this workspace variable is limited to P-by-N, where P is the Maximum number of rows parameter. (When the simulation progresses long enough for the block to acquire more than P samples, it stores only the most recent P samples.) The Decimation factor, D, allows you to store only every Dth input frame.

For an M-by-N sample-based input, the block creates a three-dimensional array in which each M-by-N page represents a single sample from each of M*N channels (the most recent input matrix occupying the last page). The maximum size of this variable is limited to M-by-N-by-P, where P is the Maximum number of rows parameter. (When the simulation progresses long enough for the block to acquire more than P inputs, it stores only the last P inputs.) The Decimation factor, D, allows you to store only every Dth input matrix.

The block acquires and buffers a single frame from input 1 whenever it is triggered by the control signal at input 2 ( ). At all other times, the block ignores input 1. You specify the triggering event at input 2 in the Trigger type pop-up menu:

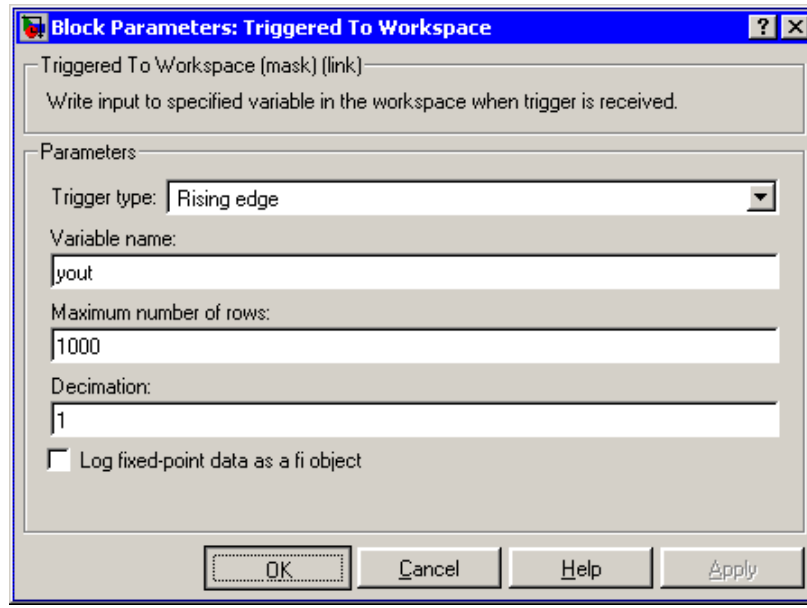
- Rising edge triggers execution of the block when the trigger input rises from a negative value to zero or a positive value, or from zero to a positive value.

- Falling edge triggers execution of the block when the trigger input falls from a positive value to zero or a negative value, or from zero to a negative value.
- Either edge triggers execution of the block when either a rising or falling edge (as described above) occurs.

To save a record of the sample time co rresponding to each sample value, open the ConÞguration Parameters dialog box. In the Select pane, click Data Import/Export . In the Save to workspace section, select the Time check box.

The nontriggered version of this block is the Simulink To Workspace block.

Dialog Box



Trigger type

The type of event that triggers the block's execution.

Variable name
: The name of the workspace matrix in which to store the data.

Maximum number of rows
: The maximum number of rows (one row per time step) to be saved, P.

Decimation
: The decimation factor, D.

Log fixed-point data as a fi object
: Select to log fixed-point data to the MATLAB workspace as a Fixed-Point Toolbox fi object. Otherwise, fixed-point data is logged to the workspace as double.

Supported Data Types

Port	Supported Data Types
Input	• Any data type supported by the To Workspace block
Trigger	• Any data type supported by the Trigger block

See Also

Signal From Workspace — Signal Processing Blockset

Signal To Workspace — Signal Processing Blockset

Triggered Signal From Workspace — Signal Processing Blockset

Two-Channel Analysis Subband Filter

Purpose

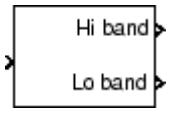
Decompose signal into high-frequency subband and low-frequency subband

Library

Filtering / Multirate Filters

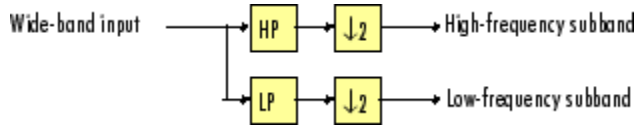
dspmlti4

Description



The Two-Channel Analysis Subband F ilter block decomposes the input into a high-frequency subband and a low-frequency subband, each with half the bandwidth and half the sample rate of the input.

The block Þlters the input with a pair of highpass and lowpass FIR Þlters, and then downsamples the results by 2, as illustrated in the following Þgure.



Note that the block implements the FIR Þltering and downsampling steps together using a polyphase Þlter structure, which is more ef Þcient than the straightforward Þlter-then-decimate algorithm illustrated above. Each subband is the Þrst phase of the respective polyphase Þlter.

You must provide the vector of Þlter coef Þcients for the two Þlters. Each Þlter should be a half-band Þlter that passes the frequency band that the other Þlter stops. For frame-based inputs, you also need to specify whether the change in the sample rate of the output gets re ßected by a change in the frame size, or the frame rate.

Note By connecting many copies of t his block, you can implement a multilevel dyadic analysis Þlter bank. In some cases, it is more efÞcient to use the Dyadic Analysis Filter Bank block instead. For more information, see “Creating Multilevel Dyadic Analysis Filter Banks” on page 2-1290.

Sections of This Reference Page

- “Specifying the FIR Filters” on page 2-1287
- “Sample-Based Operation” on page 2-1288
- “Frame-Based Operation” on page 2-1288
- “Latency” on page 2-1289
- “Creating Multilevel Dyadic Analysis Filter Banks” on page 2-1290
- “Fixed-Point Data Types” on page 2-1292
- “Examples” on page 2-1292
- “Dialog Box” on page 2-1293
- “References” on page 2-1299
- “Supported Data Types” on page 2-1300
- “See Also” on page 2-1300

Specifying the FIR Filters

You must provide the vector of numerator coefficients for the lowpass and highpass filters in the Lowpass FIR filter coefficients and Highpass FIR filter coefficients parameters.

For example, to specify a filter with the following transfer function, enter the vector [b(1) b(2) ... b(m)] .

$$H(z) = B(z) = b_1 + b_2 z^{-1} + \ldots + b_m z^{-(m-1)}$$

Each filter should be a half-band filter that passes the frequency band that the other filter stops. When you plan to use the Two-Channel Synthesis Subband Filter block to reconstruct the input to this block, you need to design perfect reconstruction filters to use in the synthesis subband filter.

The best way to design perfect reconstruction filters is to use the Wavelet Toolbox wfilters function in to design both the filters both in this block and in the Two-Channel Synthesis Subband Filter

block. You can also use Filter Design Toolbox functions and Signal Processing Toolbox functions. To learn how to design your own perfect reconstruction filters, see "References" on page 2-1299.

The block initializes all filter states to zero.

Sample-Based Operation

- "Valid Sample-Based Inputs" on page 2-1288
- "Sample-Based Outputs" on page 2-1288

Valid Sample-Based Inputs

The block accepts all M-by-N sample-based matrix inputs. The block treats such inputs as $M \cdot N$ independent channels, and decomposes each channel over time.

Sample-Based Outputs

Given a sample-based M-by-N input, the block outputs two M-by-N sample-based matrices whose sample rates are half the input sample rate. Each output matrix element is the high- or low-frequency subband output of the corresponding input matrix element. Depending on the Simulink configuration parameters, some sample-based outputs can have one sample of latency, as described in "Latency" on page 2-1289.

Frame-Based Operation

- "Valid Frame-Based Inputs" on page 2-1288
- "Frame-Based Outputs" on page 2-1289

Valid Frame-Based Inputs

The block accepts M-by-N frame-based matrix inputs where M is a multiple of two. The block treats such inputs as N independent channels, and decomposes each channel over time.

Frame-Based Outputs

Given a valid frame-based input, the block outputs two frame-based matrices. Each output column is the high- or low-frequency subband of the corresponding input column.

The sample rate of the outputs are half that of the input. The Framing parameter sets whether the block halves the sample rate by halving the output frame size, or halving the output frame rate:

- Maintain input frame size — The input and output frame sizes are the same, but the frame rate of the outputs are half that of the input. So, the overall sample rate of the output is half that of the input. This setting causes the block to have one frame of latency, as described in “Latency” on page 2-1289.
- Maintain input frame rate — The input and output frame rates are the same, but the frame size of the outputs are half that of the input (the input frame size must be a multiple of two). So, the overall sample rate of the output is half that of the input.

Latency

In some cases, the block has nonzero tasking latency, which means that there is a constant delay between the time that the block receives an input, and produces the corresponding output, as summarized below and in the following table:

- For sample-based inputs, there are cases where the block exhibits one-sample latency. In such cases, when the block receives the nth input sample, it produces the outputs corresponding to the $n-1$th input sample. When the block receives the first input sample, the block outputs an initial value of zero in each output channel.
- For frame-based inputs, there are cases where the block exhibits one-frame latency. In such cases, when the block receives the nth input frame, it produces the outputs corresponding to the $n-1$th input frame. When the block receives the first input frame, the block outputs a frame of zeros.

Note For more information on latency and the Simulink tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Models with Multiple Sample Rates” in the Real-Time Workshop User’s Guide .

Amount of Block Latency for All Possible Block Settings

Input	Latency	No Latency
Sample based	One sample of latency when the Tasking mode for periodic sample times parameter is set to MultiTasking or Auto in the Solver pane of the Con Þguration Parameters di alog box. The Þrst output sample of each channel is always 0.	The Tasking mode for periodic sample times parameter is set to SingleTasking in the Solver pane of the Con Þguration Parameters dialog box.
Frame based	One frame of latency when the Framing parameter is set to Maintain input frame size . The Þrst output frame is always all zeros.	The Framing parameter is set to Maintain input frame rate .

Creating Multilevel Dyadic Analysis Filter Banks

The Two-Channel Analysis Subband Filter block is the basic unit of a dyadic analysis Þlter bank. You can connect several of these blocks to implement an n-level Þlter bank, as illustrated in the following Þgure. For a review of dyadic analysis Þlter banks, see the Dyadic Analysis Filter Bank block reference page.

When you create a Þlter bank by connecting multiple copies of this block, the output values of the Þlter bank differ depending on whether there is latency. See the previous table,

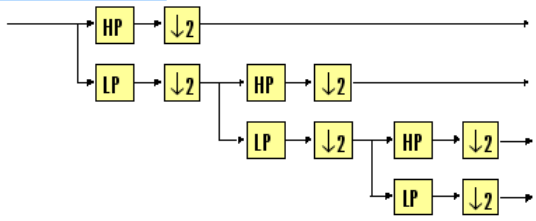
For instance, for frame-based inputs, the Þlter bank output values differ depending on whether you set the Framing parameter to Maintain input frame rate (no latency), or Maintain input frame size (one

frame of latency for every block). Though the output values differ, both sets of values are valid; the difference arises from changes in latency.

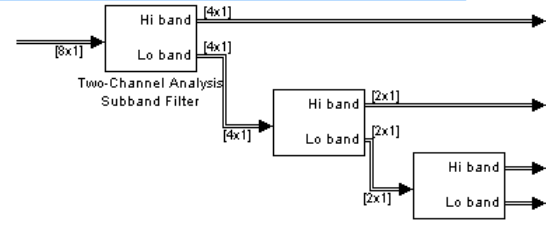
In some cases, rather than connecting several Two-Channel Analysis Subband Filter blocks, it is faster and requires less memory to use the Dyadic Analysis Filter Bank block. In particular, use the Dyadic Analysis Filter Bank block when you want to decompose a frame-based signal with frame size a multiple of 2^n into n+1 or 2^n subbands. In all other cases, use Two-Channel Analysis Subband Filter blocks to implement your filter banks.

3-Level Dyadic Analysis Filter Banks

Conceptual illustration



Two-Channel Analysis Subband Filter block implementation

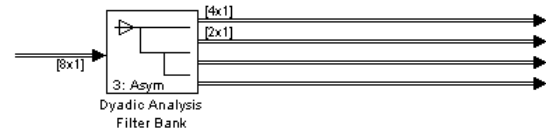


Both implementations of the dyadic analysis filter bank decompose a frame-based signal with frame size a multiple of 2^n into n+1 subbands, where n = 3.

In this case, the Dyadic Analysis Filter Bank block's implementation is more efficient.

Use the Two-Channel Analysis Subband Filter block implementation for other cases, such as to handle sample-based inputs, or to handle frame-based inputs whose frame size is not a multiple of 2^n.

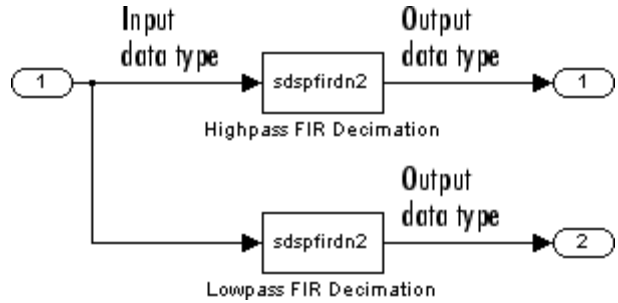
Dyadic Analysis Filter Bank block implementation



The Dyadic Analysis Filter Bank block allows you to specify the filter bank filters by providing vectors of filter coefficients, just as this block does. The Dyadic Analysis Filter Bank block provides an additional option of using wavelet-based filters that the block designs by using a wavelet you specify.

Fixed-Point Data Types

The Two-Channel Analysis Subband Filter block is comprised of two FIR Decimation blocks as shown in the following diagram.



For fixed-point signals, you can set the coefficient, product output, accumulator, and output data types of the FIR Decimation blocks as discussed in “Dialog Box” on page 2-1293. For a diagram showing the usage of these data types, see the FIR Decimation block reference page.

Examples

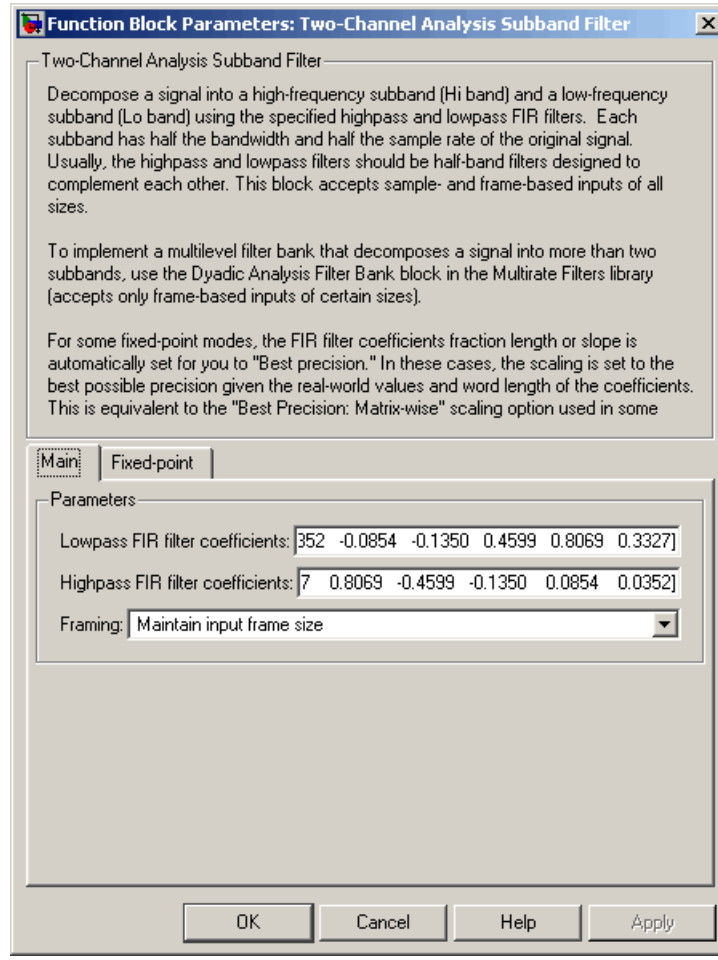
See the following Signal Processing Blockset demos, which use the Two-Channel Analysis Subband Filter block:

- Multilevel PR filter bank
- Denoising
- Wavelet transmultiplexer (WTM)

Note By default, the demos open the versions using the Two-Channel Analysis Subband Filter block. You can also see the version of the demos that use the Dyadic Analysis Filter Bank block by clicking the Frame-Based Demo button in the demos.

Dialog Box

The Main pane of the Two-Channel Analysis Subband Filter block dialog appears as follows.



Lowpass FIR filter coefficients

Specify a vector of lowpass FIR filter coefficients, in descending powers of z. The lowpass filter should be a half-band filter that

passes the frequency band stopped by the filter specified in the Highpass FIR filter coefficients parameter. The default values of this parameter specify a filter based on a 3rd-order Daubechies wavelet. When you use the Two-Channel Synthesis Subband Filter block to reconstruct the input to this block, you need to design perfect reconstruction filters to use in the synthesis subband filter. For more information, see "Specifying the FIR Filters" on page 2-1287.

Highpass FIR filter coefficients

Specify a vector of highpass FIR filter coefficients, in descending powers of z. The highpass filter should be a half-band filter that passes the frequency band stopped by the filter specified in the Lowpass FIR filter coefficients parameter. The default values of this parameter specify a filter based on a 3rd-order Daubechies wavelet. When you use the Two-Channel Synthesis Subband Filter block to reconstruct the input to this block, you need to design perfect reconstruction filters to use in the synthesis subband filter. For more information, see "Specifying the FIR Filters" on page 2-1287.

Framing

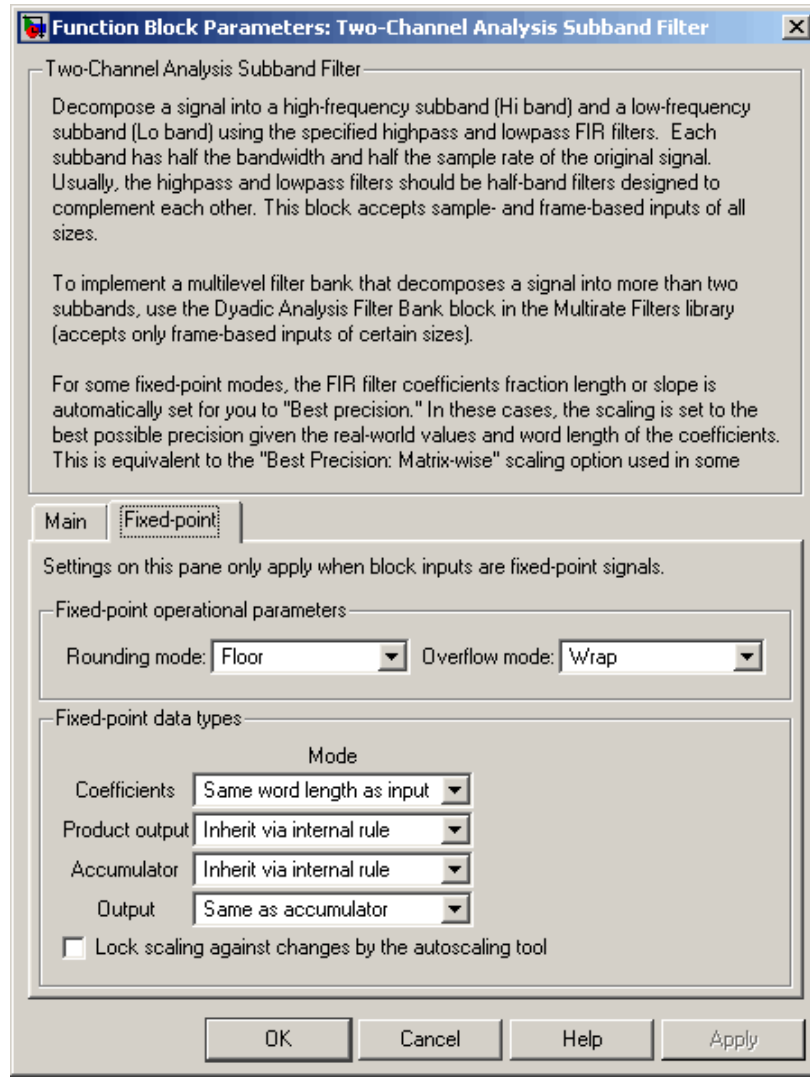
Specify the method by which to implement the decimation for frame-based inputs:

Select Maintain input frame size to halve the output frame rate

Select Maintain input frame rate to halve the output frame size

For more information, see "Frame-Based Operation" on page 2-1288. Some settings of this parameter causes the block to have nonzero latency, as described in "Latency" on page 2-1289.

The Fixed-point pane of the Two-Channel Analysis Subband Filter block dialog appears as follows.


Function Block Parameters: Two-Channel Analysis Subband Filter
Two-Channel Analysis Subband Filter
Decompose a signal into a high-frequency subband (Hi band) and a low-frequency subband (Lo band) using the specified highpass and lowpass FIR filters. Each subband has half the bandwidth and half the sample rate of the original signal. Usually, the highpass and lowpass filters should be half-band filters designed to complement each other. This block accepts sample- and frame-based inputs of all sizes.
To implement a multilevel filter bank that decomposes a signal into more than two subbands, use the Dyadic Analysis Filter Bank block in the Multirate Filters library (accepts only frame-based inputs of certain sizes).
For some fixed-point modes, the FIR filter coefficients fraction length or slope is automatically set for you to "Best precision." In these cases, the scaling is set to the best possible precision given the real-world values and word length of the coefficients. This is equivalent to the "Best Precision: Matrix-wise" scaling option used in some
Main
Fixed-point
Settings on this pane only apply when block inputs are fixed-point signals.
Fixed-point operational parameters
Rounding mode: Floor
Overflow mode: Wrap
Fixed-point data types
Mode
Coefficients
Same word length as input
Product output
Inherit via internal rule
Accumulator
Inherit via internal rule
Output
Same as accumulator
Lock scaling against changes by the autoscaling tool
OK
Cancel
Help
Apply

Rounding mode

Select the rounding mode for fixed-point operations. The filter coefficients do not obey this parameter; they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations. The filter coefficients do not obey this parameter; they are always saturated.

Coefficients

Choose how you specify the word length and the fraction length of the FIR filter coefficients:

When you select Same word length as input, the word length of the filter coefficients match that of the input to the block. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.

When you select Specify word length, you can enter the word length of the coefficients, in bits. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.

When you select Binary point scaling, you can enter the word length and the fraction length of the coefficients, in bits.

When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the coefficients. This block requires power-of-two slope and a bias of zero.

The filter coefficients do not obey the Rounding mode and the Overflow mode parameters; they are always saturated and rounded to Nearest.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-508 of the FIR De cimation reference page and “Multiplication Data Types” for illustrations depicting the use of the product output data type in the FIR Decimation blocks of this block:

When you select Inherit via internal rule , the product output word length and fraction length are calculated automatically. For informati on about how the product output word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.

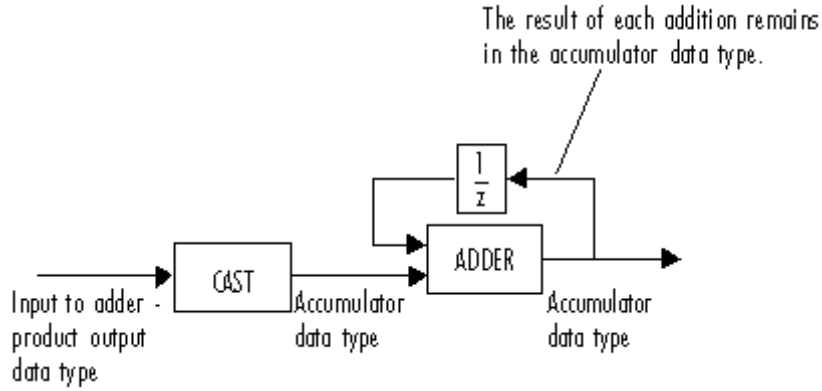
Note The actual product output word length may be equal to or greater than the calculated ideal product output word length, depending on the settings on the Hardware Implementation pane of the Con Þguration Parameters dialog box.

When you select Same as input , these characteristics match those of the input to the block.

When you select Binary point scaling , you can enter the word length and the fraction length of the product output, in bits.

When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator



As depicted above, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how you would like to designate this accumulator word and fraction lengths.

You also use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the FIR Decimation blocks in this block. See “Multiplication Data Types” for more information:

When you select Inherit via internal rule, the accumulator word length and fraction length are calculated automatically. For information about how the accumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.

When you select Same as product output, these characteristics match those of the product output

When you select Same as input, these characteristics match those of the input to the block.

When you select Binary point scaling, you can enter the word length and the fraction length of the accumulator, in bits.

When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length of the FIR Decimation blocks, as well as of the final overall filter output:

When you select Same as accumulator, these characteristics match those of the accumulator.

A special case occurs when Inherit via internal rule is specified for Accumulator, and block inputs and coefficients are complex. In that case, the output word length is one less than the accumulator word length.

When you select Same as product output, these characteristics match those of the product output

When you select Same as input, these characteristics match those of the input to the block.

When you select Binary point scaling, you can enter the word length and the fraction length of the output, in bits.

When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

References

Fliege, N. J. Multirate Digital Signal Processing: Multirate Systems, Filter Banks, Wavelets. West Sussex, England: John Wiley & Sons, 1994.

Strang, G. and T. Nguyen. Wavelets and Filter Banks . Wellesley, MA: Wellesley-Cambridge Press, 1996.

Vaidyanathan, P. P. Multirate Systems and Filter Banks . Englewood Cliffs, NJ: Prentice Hall, 1993.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Fixed point (signed only)
- 8-, 16-, and 32-bit signed integers

See Also

Dyadic Analysis Filter Bank	Signal Processing Blockset
FIR Decimation	Signal Processing Blockset
Two-Channel Synthesis Subband Filter	Signal Processing Blockset
fir1	Signal Processing Toolbox
fir2	Signal Processing Toolbox
firls	Signal Processing Toolbox
wfilters	Wavelet Toolbox

For related information, see “Multirate Filters”.

Purpose

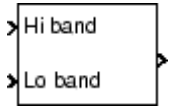
Reconstruct signal from high-freq uency subband and low-frequency subband

Library

Filtering / Multirate Filters

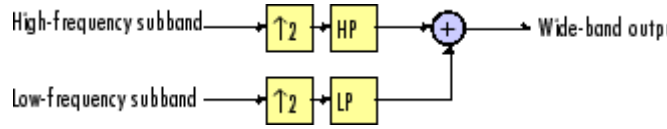
dspmlti4

Description



The Two-Channel Synthesis Subband Filter block reconstructs a signal from its high-frequency subband and low-frequency subband, each with half the bandwidth and half the sample rate of the original signal. Use this block to reconstruct signa ls decomposed by the Two-Channel Analysis Subband Filter block.

The block upsamples the high- and low-frequency subbands by 2, and then Þlters the results with a pair of highpass and lowpass FIR Þlters, as illustrated in the following Þgure.



Note that the block implements the FIR Þltering and downsampling steps together using a polyphase Þlter structure, which is more ef Þcient than the straightforwa rd interpolate-then- Þlter algorithm illustrated above.

You must provide the vector of Þlter coef Þcients for the two Þlters. Each Þlter should be a half-band Þlter that passes the frequency band that the other Þlter stops. To use this block to reconstruct the output of a Two-Channel Analysis Subband Filter block, the Þlters in this block must be designed to perfectly reconstruct the outputs of the analysis Þlters.

Note By connecting many copies of t his block, you can implement a multilevel dyadic synthesis Þlter bank. In some cases, it is more efÞcient to use the Dyadic Synthesis Filter Bank block instead. For more information, see “Creating Mu ltilevel Dyadic Synthesis Filter Banks” on page 2-1306.

Sections of This Reference Page

- “Specifying the FIR Filters” on page 2-1302
- “Sample-Based Operation” on page 2-1303
- “Frame-Based Operation” on page 2-1304
- “Latency” on page 2-1304
- “Creating Multilevel Dyadic Synthesis Filter Banks” on page 2-1306
- “Fixed-Point Data Types” on page 2-1307
- “Examples” on page 2-1308
- “Dialog Box” on page 2-1309
- “References” on page 2-1315
- “Supported Data Types” on page 2-1315
- “See Also” on page 2-1315

Specifying the FIR Filters

You must provide the vector of numerator coef Þcients for the lowpass and highpass Þlters in the Lowpass FIR **fi**lter coef **fi**cients and Highpass FIR **fi**lter coef **fi**cients parameters.

For example, to specify a Þlter with the following transfer function, enter the vector [b(1) b(2) ... b(m)] .

$$H(z) = B(z) = b_1 + b_2 z^{-1} + \ldots + b_m z^{-(m-1)}$$

Each filter should be a half-band filter that passes the frequency band that the other filter stops. To use this block to reconstruct the output of a Two-Channel Analysis Subband Filter block, the filters in this block must be designed to perfectly reconstruct the outputs of the analysis filters.

The best way to design perfect reconstruction filters is to use the Wavelet Toolbox wfilters function for the filters in both this block and in the corresponding Two-Channel Analysis Subband Filter block. You can also use Filter Design Toolbox functions and Signal Processing Toolbox functions. To learn how to design your own perfect reconstruction filters, see "References" on page 2-1315.

The block initializes all filter states to zero.

Sample-Based Operation

- "Valid Sample-Based Inputs" on page 2-1303
- "Sample-Based Outputs" on page 2-1303

Valid Sample-Based Inputs

The block accepts any two M-by-N sample-based matrices with the same sample rates. The block treats each M-by-N matrix as MxN independent subbands, where MxN is the product of the matrix dimensions. Each matrix element is the high- or low-frequency subband of the corresponding channel in the output matrix. The input to the topmost input port should contain the high-frequency subbands.

Sample-Based Outputs

Given valid sample-based inputs, the block outputs one sample-based matrix with the same dimensions as the inputs. The output sample rate is twice that of the input. Each element of the output is a single channel, reconstructed from the corresponding elements in each input matrix. Depending on the Simulink configuration parameters, some sample-based outputs can have one sample of latency, as described in "Latency" on page 2-1304.

Frame-Based Operation

- “Valid Frame-Based Inputs” on page 2-1304
- “Frame-Based Outputs” on page 2-1304

Valid Frame-Based Inputs

The block accepts any two M-by-N f rame-based matrices with the same frame rates. The block treats each input column as the high- or low-frequency subbands of the corresponding output channel. The input to the topmost input port should contain the high-frequency subbands.

Frame-Based Outputs

Given valid frame-based inputs, the block outputs a frame-based matrix. Each output column is a sing le channel, reconstructed from the corresponding columns in each input matrix.

The sample rate of the output is twice that of the input. The Framing parameter sets whether the block doubles the sample rate by doubling the output frame size, or doubling the output frame rate:

- Maintain input frame size — The input and output frame sizes are the same, but the frame rate of the output is twice that of the input. So, the overall sample rate of the output is twice that of the input. This setting causes the blo ck to have one frame of latency, as described in “Latency” on page 2-1289.
- Maintain input frame rate — The input and output frame rates are the same, but the frame size of the output is twice that of the input. So, the overall sample rate of the output is twice that of the input.

Latency

In some cases, the block has nonzero tasking latency, which means that there is a constant delay between the time that the block receives an input, and produces the correspon ding output, as summarized below and in the following table:

- For sample-based inputs, there are cases where the block exhibits one-sample latency. In such cases, when the block receives the nth input sample, it produces the outputs corresponding to the n-1th input sample. When the block receives the first input sample, the block outputs an initial value of zero in each output channel.
- For frame-based inputs, there are cases where the block exhibits one-frame latency. In such cases, when the block receives the nth input frame, it produces the outputs corresponding to the n-1th input frame. When the block receives the first input frame, the block outputs a frame of zeros.

Note For more information on latency and the Simulink tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Models with Multiple Sample Rates” in the Real-Time Workshop User’s Guide .

Amount of Block Latency for All Possible Block Settings

Input	Latency	No Latency
Sample based	One sample of latency when the Tasking mode for periodic sample times parameter is set to MultiTasking or Auto in the Solver pane of the Configuration Parameters dialog box. The first output sample of each channel is always 0.	The Tasking mode for periodic sample times parameter is set to SingleTasking in the Solver pane of the Configuration Parameters dialog box.
Frame based	One frame of latency when the Framing parameter is set to Maintain input frame size . The first output frame is always all zeros.	The Framing parameter is set to Maintain input frame rate.

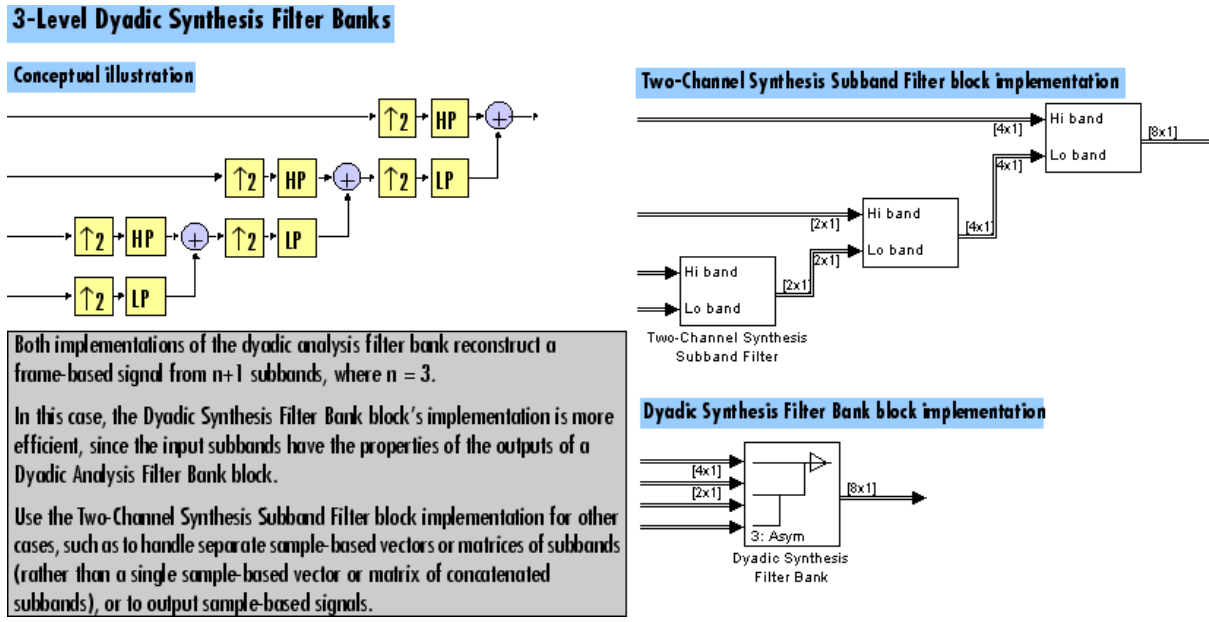
Creating Multilevel Dyadic Synthesis Filter Banks

The Two-Channel Synthesis Subband Filter block is the basic unit of a dyadic synthesis filter bank. You can connect several of these blocks to implement an n-level filter bank, as illustrated in the following figure. For a review of dyadic synthesis filter banks, see the Dyadic Synthesis Filter Bank block reference page.

When you create a filter bank by connecting multiple copies of this block, the output values of the filter bank differ depending on whether there is latency. See the previous table, Amount of Block Latency for All Possible Block Settings on page 2-1305.

For instance, for frame-based inputs, the filter bank output values differ depending on whether you set the Framing parameter to Maintain input frame rate (no latency), or Maintain input frame size (one frame of latency for every block). Though the output values differ, both sets of values are valid; the difference arises from changes in latency.

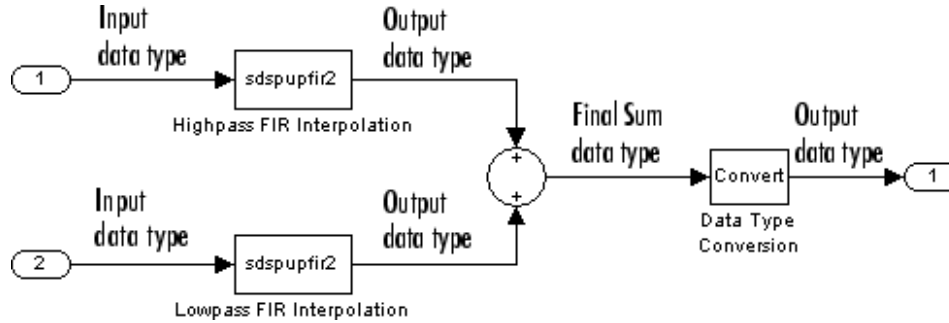
In some cases, rather than connecting several Two-Channel Synthesis Subband Filter blocks, it is faster and requires less memory to use the Dyadic Synthesis Filter Bank block. In particular, use the Dyadic Synthesis Filter Bank block to reconstruct a frame-based signal (with frame size a multiple of 2^n) from 2^n or n+1 subbands whose properties match those of the Dyadic Analysis Filter Bank block's outputs. These properties are described in the Dyadic Analysis Filter Bank reference page.



The Dyadic Synthesis Filter Bank block allows you to specify the filter bank filters by providing vectors of filter coefficients, just as this block does. The Dyadic Synthesis Filter Bank block provides an additional option of using wavelet-based filters that the block designs by using a wavelet you specify.

Fixed-Point Data Types

The Two-Channel Synthesis Subband Filter block is comprised of two FIR Interpolation blocks as shown in the following diagram.



For fixed-point signals, you can set the coefficient, product output, accumulator, and output data types used in the FIR Interpolation blocks as discussed in "Dialog Box" on page 2-1309 below. For a diagram showing the usage of these data types within the FIR blocks, see the FIR Interpolation block reference page.

In addition, the inputs to the Sum block in the diagram above are accumulated using the accumulator data type. The output of the Sum block is then cast from the accumulator data type to the output data type. Therefore the output of the Two-Channel Synthesis Subband Filter block is in the output data type. You also set these data types in the block dialog as discussed in "Dialog Box" on page 2-1309 below.

Examples

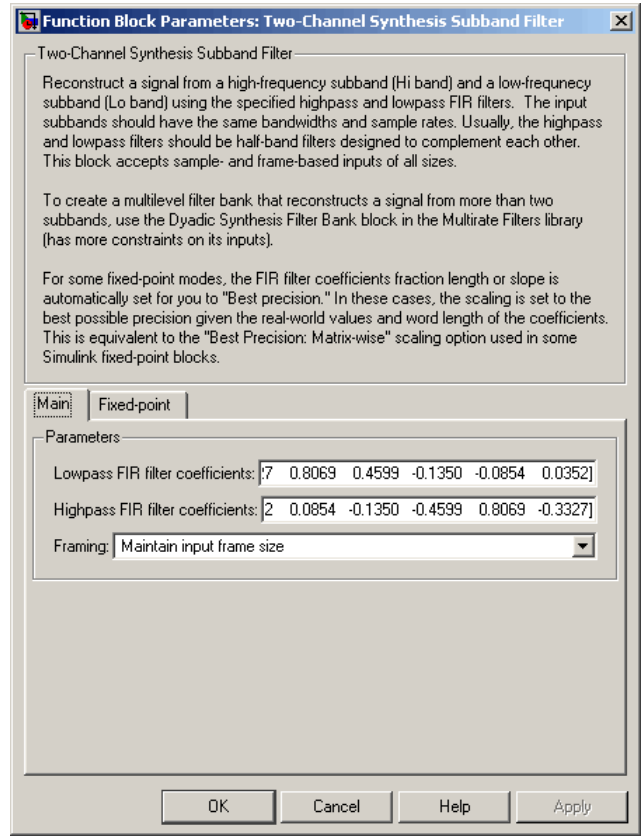
See the following Signal Processing Blockset demos, which use the Two-Channel Synthesis Subband Filter block:

- Multilevel PR filter bank
- Denoising
- Wavelet transmultiplexer (WTM)

Note By default, the demos open the versions using the Two-Channel Synthesis Subband Filter block. You can also see the version of the demos that use the Dyadic Synthesis Filter Bank block by clicking the Frame-Based Demo button in the demos.

Dialog Box

The Main pane of the Two-Channel Synthesis Subband Filter block dialog appears as follows.



Lowpass FIR filter coefficients

A vector of lowpass FIR filter coefficients, in descending powers of z. The lowpass filter should be a half-band filter that passes the frequency band stopped by the filter specified in the Highpass FIR filter coefficients parameter. To use this block to reconstruct the output of a Two-Channel Analysis Subband Filter block, you must design the filters in this block to

perfectly reconstruct the outputs of the analysis filters. For more information, see “Specifying the FIR Filters” on page 2-1302.

Highpass FIR filter coefficients

A vector of highpass FIR filter coefficients, in descending powers of z. The highpass filter should be a half-band filter that passes the frequency band stopped by the filter specified in the Lowpass FIR filter coefficients parameter. To use this block to reconstruct the output of a Two-Channel Analysis Subband Filter block, you must design the filters in this block to perfectly reconstruct the outputs of the analysis filters. For more information, see “Specifying the FIR Filters” on page 2-1302.

Framing

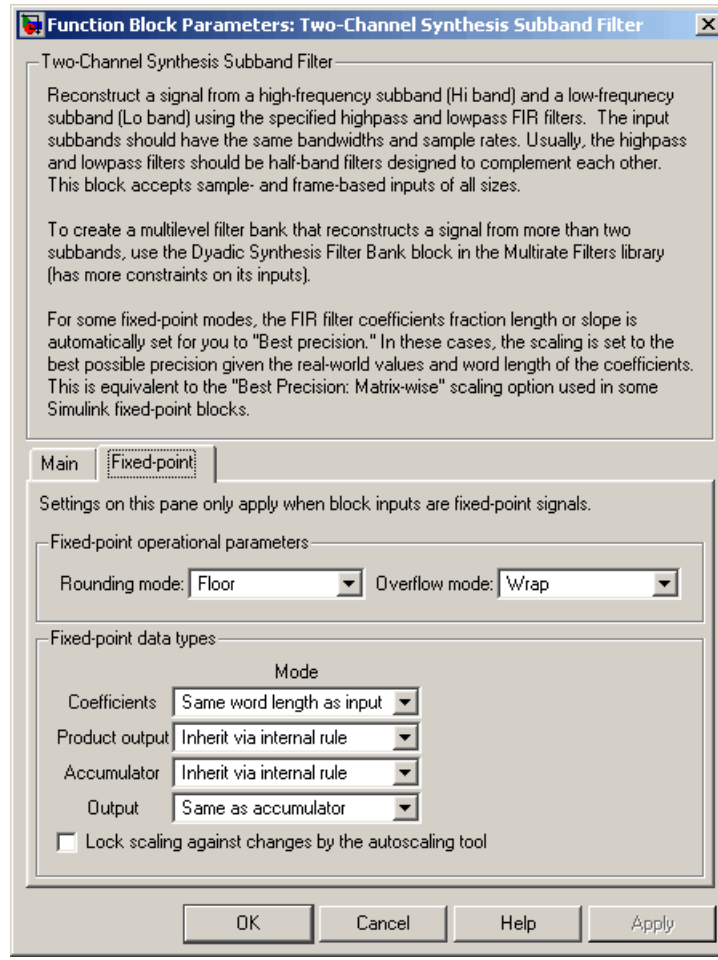
Select the method by which to implement the interpolation for frame-based inputs:

Select Maintain input frame size to double the output frame rate

Select Maintain input frame rate to double the output frame size

For more information, see “Frame-Based Operation” on page 2-1288. Some settings of this parameter causes the block to have nonzero latency, as described in “Latency” on page 2-1289.

The Fixed-point pane of the Two-Channel Synthesis Subband Filter block dialog appears as follows.



Round mode

Select the rounding mode for Þxed-point operations. The Þlter coefÞcients do not obey this parameter; they always round to Nearest .

Overflow mode

Select the overflow mode for fixed-point operations. The filter coefficients do not obey this parameter; they are always saturated.

Coefficients

Choose how you specify the word length and the fraction length of the FIR filter coefficients:

When you select Same word length as input, the word length of the filter coefficients match that of the input to the block. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.

When you select Specify word length, you can enter the word length of the coefficients, in bits. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.

When you select Binary point scaling, you can enter the word length and the fraction length of the coefficients, in bits.

When you select Slope and bias scaling, you can enter the word length, in bits, and the slope of the coefficients. This block requires power-of-two slope and a bias of zero.

The filter coefficients do not obey the Rounding mode and the Overflow mode parameters; they are always saturated and rounded to Nearest.

Product output

Use this parameter to specify how you would like to designate the product output word and fraction lengths. See “Fixed-Point Data Types” on page 2-527 of the FIR Interpolation reference page and “Multiplication Data Types” for illustrations depicting the use of

the product output data type in th e FIR Interpolation blocks of this block:

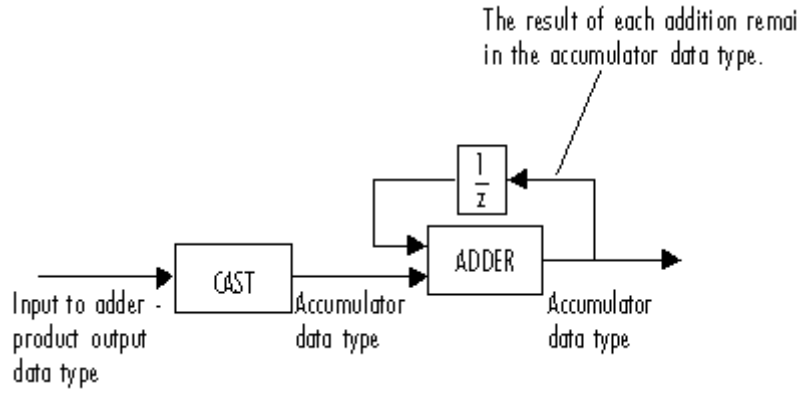
When you select Inherit via internal rule , the product output word length and fraction length are calculated automatically. For informati on about how the product output word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.

When you select Same as input , these characteristics match those of the input to the block.

When you select Binary point scaling , you can enter the word length and the fraction length of the product output, in bits.

When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator



As depicted above, inputs to the accumulator are cast to the accumul ator data type. The output of the adder remains in the accumul ator data type as each element of the input is added to

it. Use this parameter to specify how you would like to designate this accumulator word and fraction lengths.

You also use this parameter to s pecify the accumulator word and fraction lengths resulting from a c omplex-complex multiplication in the FIR Interpolation blocks in this block. See “Multiplication Data Types” for more information:

When you select Inherit via internal rule , the accumulator word length and fraction length are calculated automatically. For information about how the a ccumulator word and fraction lengths are calculated when an internal rule is used, see “Inherit via Internal Rule”.

When you select Same as product output , these characteristics match those of the product output

When you select Same as input , these characteristics match those of the input to the block.

When you select Binary point scaling , you can enter the word length and the fraction length of the accumulator, in bits.

When you select Slope and bias scaling , you can enter the word length, in bits, and the slop e of the accumulator. This block requires power-of-two slope and a bias of zero.

Output

Choose how you specify the output word length and fraction length of the FIR Interpolation blocks, as well as of the Þnal overall Þlter output:

When you select Same as accumulator , these characteristics match those of the accumulator.

A special case occurs when Inherit via internal rule is speciÞed for Accumulator , and block inputs and coef Þcients are

complex. In that case, the output word length is one less than the accumulator word length.

When you select Same as product output , these characteristics match those of the product output

When you select Same as input , these characteristics match those of the input to the block.

When you select Binary point scaling , you can enter the word length and the fraction length of the output, in bits.

When you select Slope and bias scaling , you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of zero.

References

Fliege, N. J. Multirate Digital Signal Pr ocessing: Multirate Systems, Filter Banks, Wavelets . West Sussex, England: John Wiley & Sons, 1994.

Strang, G. and T. Nguyen. Wavelets and Filter Banks . Wellesley, MA: Wellesley-Cambridge Press, 1996.

Vaidyanathan, P. P. Multirate Systems and Filter Banks . Englewood Cliffs, NJ: Prentice Hall, 1993.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point
- Fixed point (signed only)
- 8-, 16-, and 32-bit signed integers

See Also

Dyadic Synthesis Filter Bank — Signal Processing Blockset

FIR Interpolation — Signal Processing Blockset

Two-Channel Analysis Subband Filter	Signal Processing Blockset
fir1	Signal Processing Toolbox
fir2	Signal Processing Toolbox
firls	Signal Processing Toolbox
wfilters	Wavelet Toolbox

For related information, see “Multirate Filters”.

Purpose

Unbuffer input frame into sequence of scalar outputs

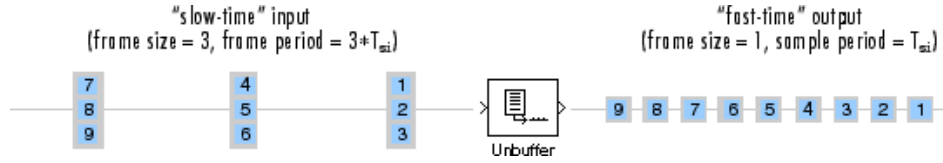
Library

Signal Management / Buffers

dspbuff3

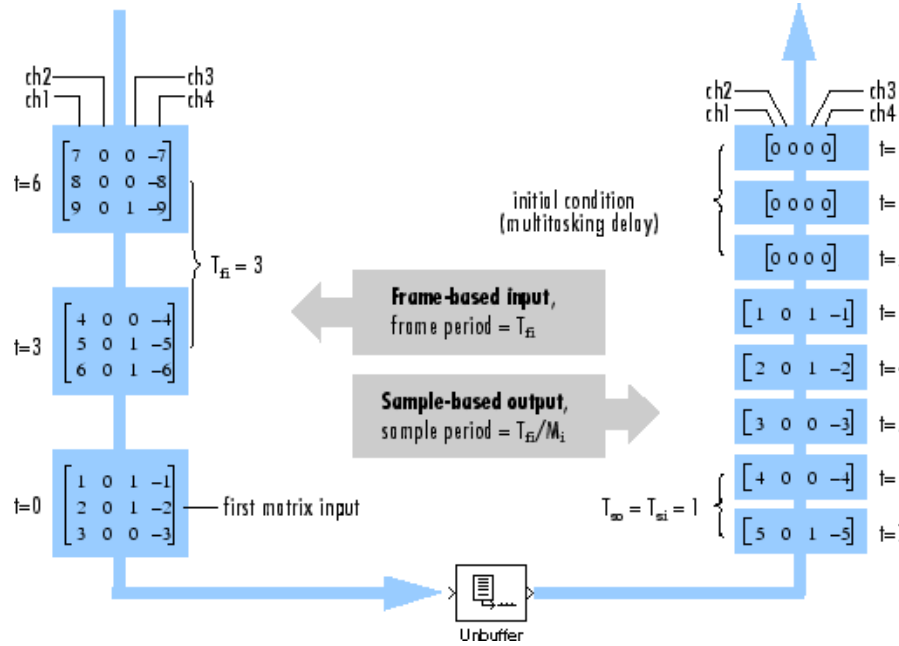
Description

The Unbuffer block unbuffers an M_i-by-N frame-based input into a 1-by-N sample-based output. That is, inputs are unbuffered row-wise so that each matrix row becomes an independent time-sample in the output. The rate at which the block receives inputs is generally less than the rate at which the block produces outputs.



The block adjusts the output rate so that the sample period is the same at both the input and output, $T_{so}=T_{si}$. Therefore, the output sample period for an input of frame size M_i and frame period T_p is T_p/M_i, which represents a rate M_i times higher than the input frame rate. In the example above, the block receives inputs only once every three sample periods, but produces an output once every sample period. To rebuffer frame-based inputs to a larger or smaller frame size, use the Buffer block.

In the model below, the block unbuffers a four-channel frame-based input with frame size 3. The Initial conditions parameter is set to zero and the tasking mode is set to multitasking, so the first three outputs are zero vectors.



Zero Latency

The Unbuffer block has zero-tasking latency in the Simulink single-tasking mode. Zero-tasking latency means that the Þrst input sample (received at t=0) appears as the Þrst output sample.

Nonzero Latency

For multitasking operation, the Unbuffer block's buffer is initialized with the value speciÞed by the Initial condition parameter, and the block begins unbuffering this frame at the start of the simulation. Inputs to the block are therefore delayed by one buffer length, or M_i samples.

The Initial condition parameter can be one of the following:

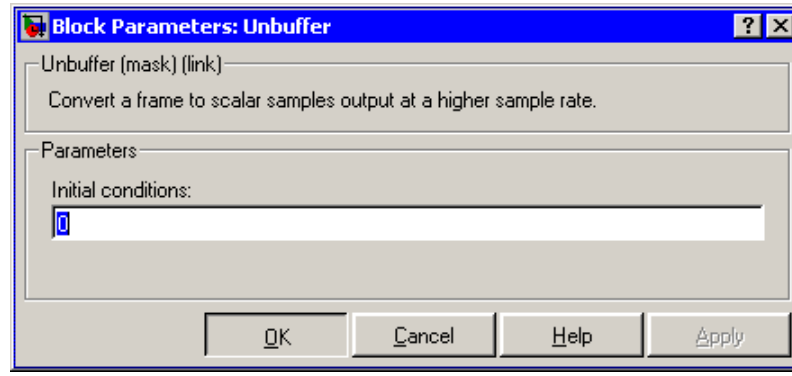
- A scalar to be repeated for the Þrst M_i output samples of every channel

- A length-M_i vector containing the values of the first M_i output samples for every channel
- An M_i-by-N matrix containing the values of the first M_i output samples in each of N channels

---

Note For more information on latency and the Simulink tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Models with Multiple Sample Rates” in the Real-Time Workshop User’s Guide .

Dialog Box



Initial conditions

The value of the block’s initial output for cases of nonzero latency; a scalar, vector, or matrix.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Buffer Signal Processing Blockset

See “Unbuffering Frame-Based Signals into Sample-Based Signals” for related information.

Purpose

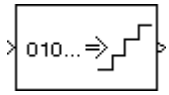
Decode integer input into floating-point output

Library

Quantizers

dspquant2

Description



The Uniform Decoder block performs the inverse operation of the Uniform Encoder block, and reconstructs quantized floating-point values from encoded integer input. The block adheres to the definition for uniform decoding specified in ITU-T Recommendation G.701.

Inputs can be real or complex values of the following six integer data types: uint8, uint16, uint32, int8, int16, or int32.

The block first casts the integer input values to floating-point values, and then uniquely maps (decodes) them to one of 2^B uniformly spaced floating-point values in the range $[-V, (1-2^{1-B})V]$, where you specify B in the Bits parameter (as an integer between 2 and 32) and V is a floating-point value specified by the Peak parameter. The smallest input value representable by B bits (0 for an unsigned input data type; -2^{B-1} for a signed input data type) is mapped to the value -V. The largest input value representable by B bits (2^B-1 for an unsigned input data type; $2^{B-1}-1$ for a signed input data type) is mapped to the value $(1-2^{1-B})V$. Intermediate input values are linearly mapped to the intermediate values in the range $[-V, (1-2^{1-B})V]$.

To correctly decode values encoded by the Uniform Encoder block, the Bits and Peak parameters of the Uniform Decoder block should be set to the same values as the Bits and Peak parameters of the Uniform Encoder block. The Overflow mode parameter specifies the Uniform Decoder block's behavior when the integer input is outside the range representable by B bits. When you select Saturate, unsigned input values greater than 2^B-1 saturate at 2^B-1; signed input values greater than $2^{B-1}-1$ or less than -2^{B-1} saturate at those limits. The real and imaginary components of complex inputs saturate independently.

When you select Wrap, unsigned input values, u, greater than 2^B-1 are wrapped back into the range $[0, 2^B-1]$ using mod-2^B arithmetic.

u = mod(u,2^B)

Signed input values, u, greater than 2^{B-1}-1 or less than -2^{B-1} are wrapped back into that range using mod-2^{B} arithmetic.

u = (mod(u+2^B/2,2^B)-(2^B/2))

The real and imaginary components of complex inputs wrap independently.

The Output type parameter specifies whether the decoded floating-point output is single or double precision. Either level of output precision can be used with any of the six integer input data types.

Examples

Consider a Uniform Decoder block with the following parameter settings:

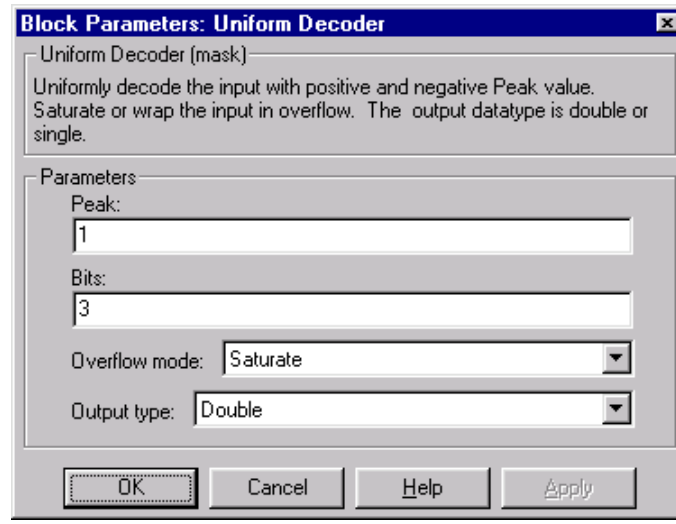
- Peak = 2
- Bits = 3

The input to the block is the uint8 output of a Uniform Encoder block with comparable settings: Peak = 2, Bits = 3, and Output type = Unsigned. (Comparable settings ensure that inputs to the Uniform Decoder block do not saturate or wrap. See the example on the Uniform Encoder block reference page for more about these settings.)

The real and complex components of each input are independently mapped to one of 2^3 distinct levels in the range [-2.0,1.5].

0 is mapped to -2.0
1 is mapped to -1.5
2 is mapped to -1.0
3 is mapped to -0.5
4 is mapped to 0.0
5 is mapped to 0.5
6 is mapped to 1.0
7 is mapped to 1.5

Dialog Box



Peak

Specify the largest amplitude represented in the encoded input. To correctly decode values encoded with the Uniform Encoder block, set the Peak parameters in both blocks to the same value.

Bits

Specify the number of input bits, B, used to encode the data. (This can be less than the total number of bits supplied by the input data type.) To correctly decode values encoded with the Uniform Encoder block, set the Bits parameters in both blocks to the same value.

Overflow mode

Specify the block's behavior when the integer input is outside the range representable by B bits. Out-of-range inputs can either saturate at the extreme value, or wrap back into range.

Output type

Specify the precision of the floating-point output, single or double.

References

General Aspects of Digital Transmission Systems: Vocabulary of Digital Transmission and Multipl exing, and Pulse Code Modulation (PCM) Terms, International Telecommunication Union, ITU-T Recommendation G.701, March, 1993

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Data Type Conversion	Simulink
Quantizer	Simulink
Scalar Quantizer Decoder	Signal Processing Blockset
Uniform Encoder	Signal Processing Blockset
udecode	Signal Processing Toolbox
uencode	Signal Processing Toolbox

Purpose

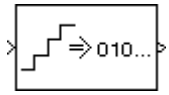
Quantize and encode floating-point input into integer output

Library

Quantizers

dspquant2

Description



The Uniform Encoder block performs the following two operations on each floating-point sample in the input vector or matrix:

1 Quantizes the value using the same precision

2 Encodes the quantized floating-point value to an integer value

In the first step, the block quantizes an input value to one of 2^B uniformly spaced levels in the range $[-V, (1-2^{1-B})V]$, where you specify B in the Bits parameter and you specify V in the Peak parameter. The quantization process rounds both positive and negative inputs downward to the nearest quantization level, with the exception of those that fall exactly on a quantization boundary. The real and imaginary components of complex inputs are quantized independently.

The number of bits, B, can be any integer value between 2 and 32, inclusive. Inputs greater than $(1-2^{1-B})V$ or less than -V saturate at those respective values. The real and imaginary components of complex inputs saturate independently.

In the second step, the quantized floating-point value is uniquely mapped (encoded) to one of 2^B integer values. When the Output type is set to Unsigned integer, the smallest quantized floating-point value, -V, is mapped to the integer 0, and the largest quantized floating-point value, $(1-2^{1-B})V$, is mapped to the integer 2^B-1. Intermediate quantized floating-point values are linearly (uniformly) mapped to the intermediate integers in the range $[0, 2^B-1]$. For efficiency, the block automatically selects an unsigned output data type (uint8, uint16, or uint32) with the minimum number of bits equal to or greater than B.

When the Output type is set to Signed integer, the smallest quantized floating-point value, -V, is mapped to the integer -2^{B-1}, and the largest quantized floating-point value, $(1-2^{1-B})V$, is mapped to the

integer $2^{B-1}-1$. Intermediate quantized ßoating-point values are linearly mapped to the intermediate integers in the range $[-2^{B-1}, 2^{B-1}-1]$. The block automatically selects a signed output data type (int8 , int16 , or int32) with the minimum number of bits equal to or greater than B.

Inputs can be real or complex, d ouble or single precision. The output data types that the block uses are shown in the table below. Note that most of the Signal Processing Blockset blocks accept only double-precision inputs. Use the Simulink Data Type Conversion block to convert integer data types to double precision. See “Working with Data Types” in the Simulin k documentation for a complete discussion of data types, as well as a list of Simulink blocks capable of reduced-precision operations.

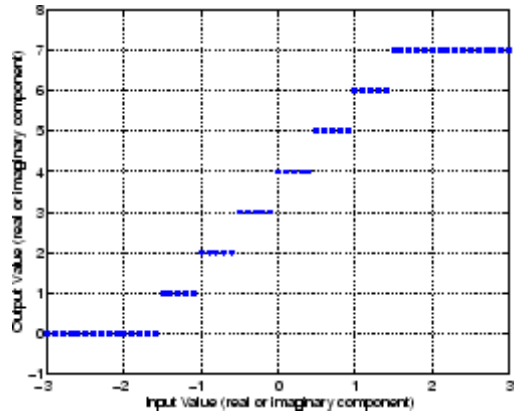
Bits	Unsigned Integer	Signed Integer
2 to 8	uint8	int8
9 to 16	uint16	int16
17 to 32	uint32	int32

The Uniform Encoder block operations adhere to the de Þnition for uniform encoding speci Þed in ITU-T Recommendation G.701.

Examples

The following Þgure illustrates uniform en coding with the following parameter settings:

- Peak = 2
- Bits = 3
- Output type = Unsigned



The real and complex components of each input (horizontal axis) are independently quantized to one of 2^3 distinct levels in the range [-2,1.5] and then mapped to one of 2^3 integer values in the range [0,7] .

-2.0 is mapped to 0
-1.5 is mapped to 1
-1.0 is mapped to 2
-0.5 is mapped to 3
0.0 is mapped to 4
0.5 is mapped to 5
1.0 is mapped to 6
1.5 is mapped to 7

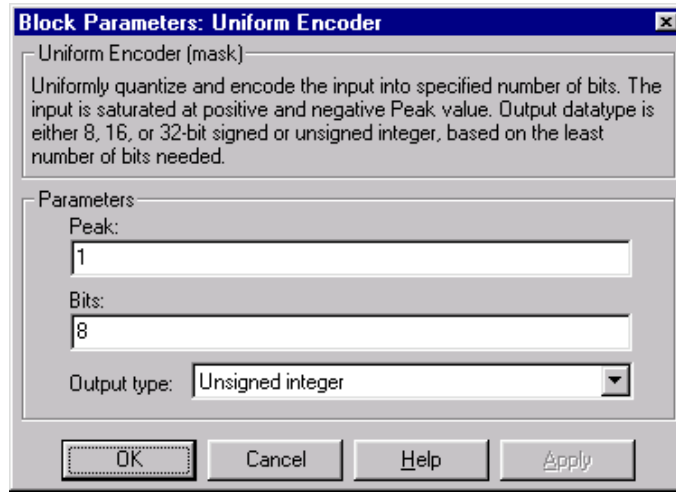
The table below shows the result s for a few particular inputs.

Input	Quantized Input	Output	Notes
1.6	1.5+0.0i	7+4i	
-0.4	-0.5+0.0i	3+4i	
-3.2	-2.0+0.0i	4i	Saturation (real)

Input	Quantized Input	Output	Notes
0.4-1.2i	0.0-1.5i 4+i		
0.4-6.0i	0.0-2.0i 4		Saturation (imaginary)
-4.2+3.5i	-2.0+2.0i 7i		Saturation (real and imaginary)

The output data type is automatically set to uint8 , the most efficient format for this input range.

Dialog Box



Peak

The largest input amplitude to be encoded, V. Real or imaginary input values greater than (1-2 $^{1-B}$)V or less than -V saturate (independently for complex inputs) at those limits.

Bits

The number of levels at which to quantize the ßoating-point input. (Also the number of bits needed to represent the integer output.)

Output type

The data type of the block's output, Unsigned integer or Signed integer . Unsigned outputs are uint8 , uint16 , or uint32 , while signed outputs are int8 , int16 , or int32 .

References

General Aspects of Digital Transmission Systems: Vocabulary of Digital Transmission and Multipl exing, and Pulse Code Modulation (PCM) Terms, International Telecommunication Union, ITU-T Recommendation G.701, March, 1993

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

Data Type Conversion	Simulink
Quantizer	Simulink
Scalar Quantizer Decoder	Signal Processing Blockset
Uniform Decoder	Signal Processing Blockset
udecode	Signal Processing Toolbox
uencode	Signal Processing Toolbox

Unwrap

Purpose

Unwrap signal phase

Library

Signal Operations

dspsigops

Description

The Unwrap block unwraps each input channel by adding or subtracting appropriate multiples of 2 π to each channel element. The input can be any matrix or 1-D vector, and must have radian phase entries. The block recognizes phase discontinuities larger than the Tolerance parameter setting.

The block preserves the input size, dimension, and frame status, and the output port rate equals the input port rate. For a detailed discussion of the Unwrap block, see other sections of this reference page.

Sections of This Reference Page

- "Acceptable Inputs and Corresponding Output Characteristics"
- "The Two Unwrap Modes"
- "Unwrap Method"
- "Definition of Phase Unwrap"

Acceptable Inputs and Corresponding Output Characteristics

The Unwrap block preserves the input size, dimension, and frame status, and the output port rate equals the input port rate.

Characteristics of Valid Input	Characteristics of Corresponding Output
Input elements must be phase values in radians. Sample- or frame-based M-by-N 2-D matrix or a 1-D vector	Output elements are phase values in radians. Same frame status as input Same size and dimension as input Output port rate = input port rate

The Two Unwrap Modes

You must specify the unwrap mode by setting the parameter, Do not unwrap phase discontinuities between successive frames . The unwrap modes are summarized in the next table.

Two Unwrap Modes	
In both unwrap modes, the block adds 2 πk to each input channel's elements, where it updates k at each phase discontinuity. (For more on the updating of k, see "Unwrap Method" on page 2-1334.) The number of times that k is reset to 0 depends on the unwrap mode.	
Default Unwrap Mode: Initialize k to 0 for Only the First Input Frame	Nondefault Unwrap Mode: Set k to 0 for Each Successive Input Matrix or Input Vector
In this mode, k is initialized to 0 for only the first input matrix or input vector. As k gets updated, the value of k is retained between successive input matrices or input vectors. That is, the block unwraps each input's channel by considering phase discontinuities in all previous frames and the current frame.	In this mode, k is reset to 0 for each successive input matrix or input vector. As k gets updated, the value of k is only retained within the current input matrix or vector. That is, the block unwraps each input's channel by considering phase discontinuities in the current input matrix or input vector only, ignoring discontinuities in previous inputs.

Two Unwrap Modes	
In both unwrap modes, the block adds $2\pi k$ to each input channel's elements, where it updates k at each phase discontinuity. (For more on the updating of k, see "Unwrap Method" on page 2-1334.) The number of times that k is reset to 0 depends on the unwrap mode.	
Default Unwrap Mode: Initialize k to 0 for Only the First Input Frame	Nondefault Unwrap Mode: Set k to 0 for Each Successive Input Matrix or Input Vector
In this mode, the block unwraps the columns or each individual element of the input: • Frame-based inputs — unwrap columns • Sample-based inputs — unwrap each element of the input. • 1-D vector inputs — treat as frame-based column	In this mode, the block unwraps the columns or rows of the input: • Frame-based inputs — unwrap columns • Sample-based nonrow inputs — unwrap columns • Sample-based row vector inputs — unwrap the row. • 1-D vector inputs — treat as frame-based column
See the following diagrams.	See the following diagrams.

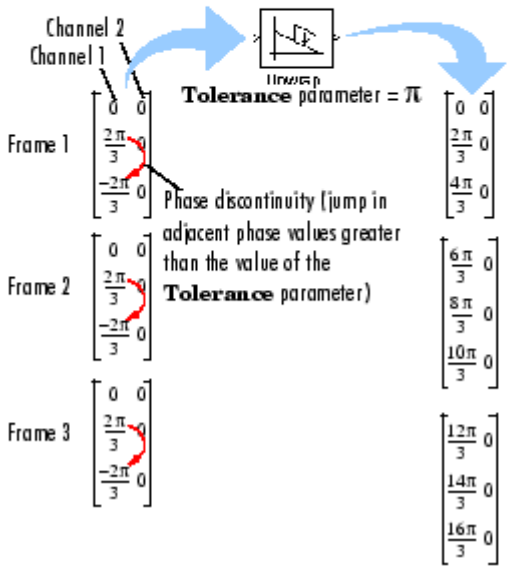
The following diagrams illustrate how the two unwrap modes operate on various inputs.

Default Unwrap Mode Operation:

☐ Do not unwrap phase discontinuities between successive frames

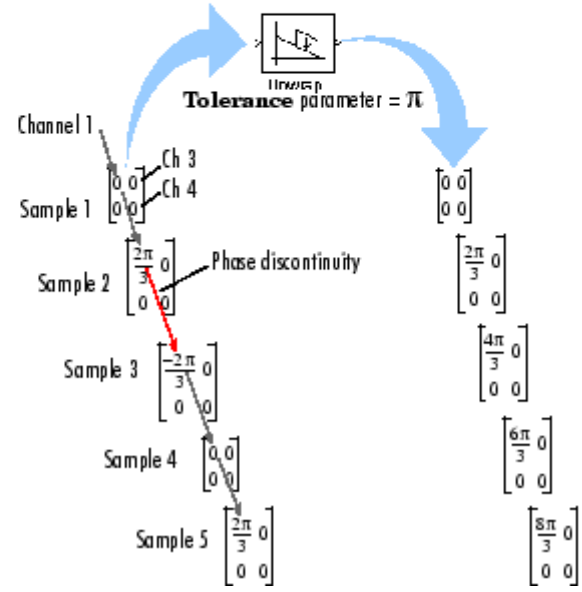
Frame-Based Inputs

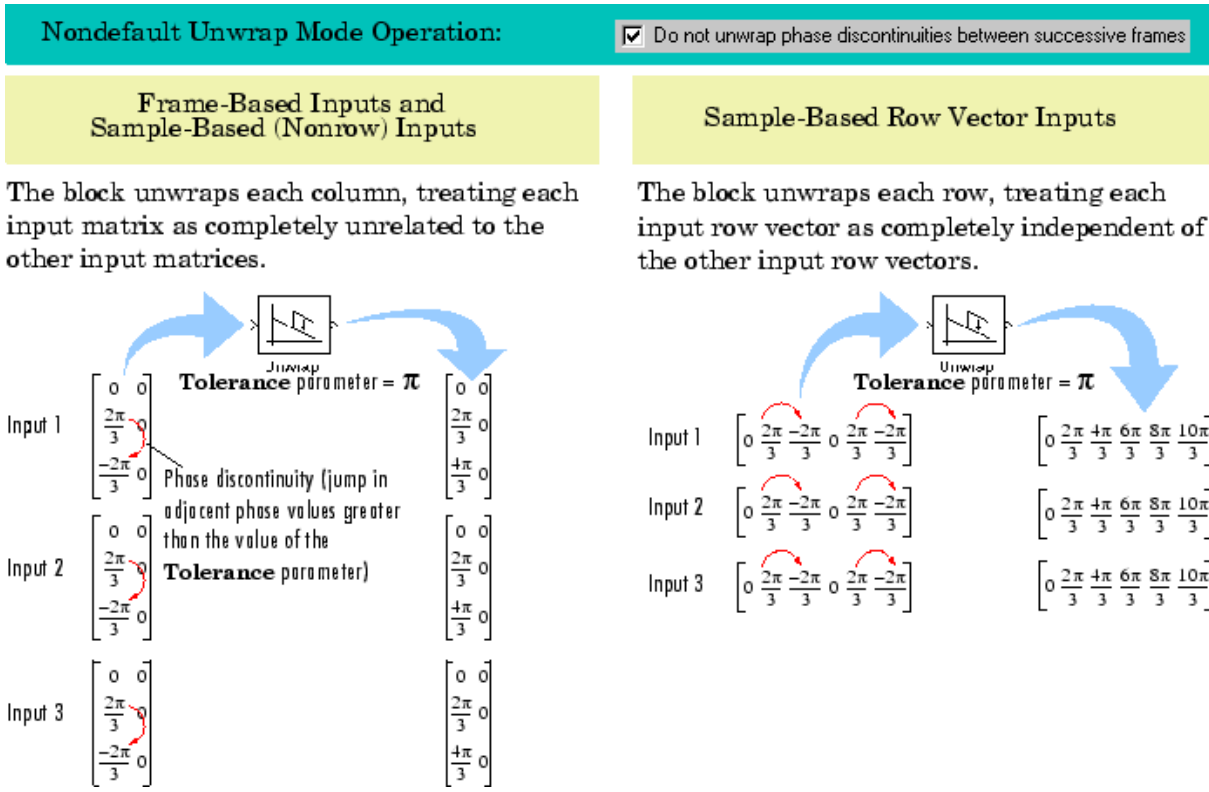
The block treats each input column as an independent channel. It unwraps by treating Channel 1 of Frame 2 as a continuation of Channel 1 of Frame 1.



Sample-Based Inputs

The block treats each element of the input matrix as an independent channel. (The first sample in Channel 1 is in the upper left corner of the Sample 1 matrix. The second sample of Channel 1 is in the corresponding corner of the Sample 2 matrix, and so on.)





Unwrap M ethod

The Unwr ap block unwraps each channel of its input matrix or input vector by adding 2 πk to each successive channel element, and updating k at each phase jump. See the following steps to the unwrap method for det ails.

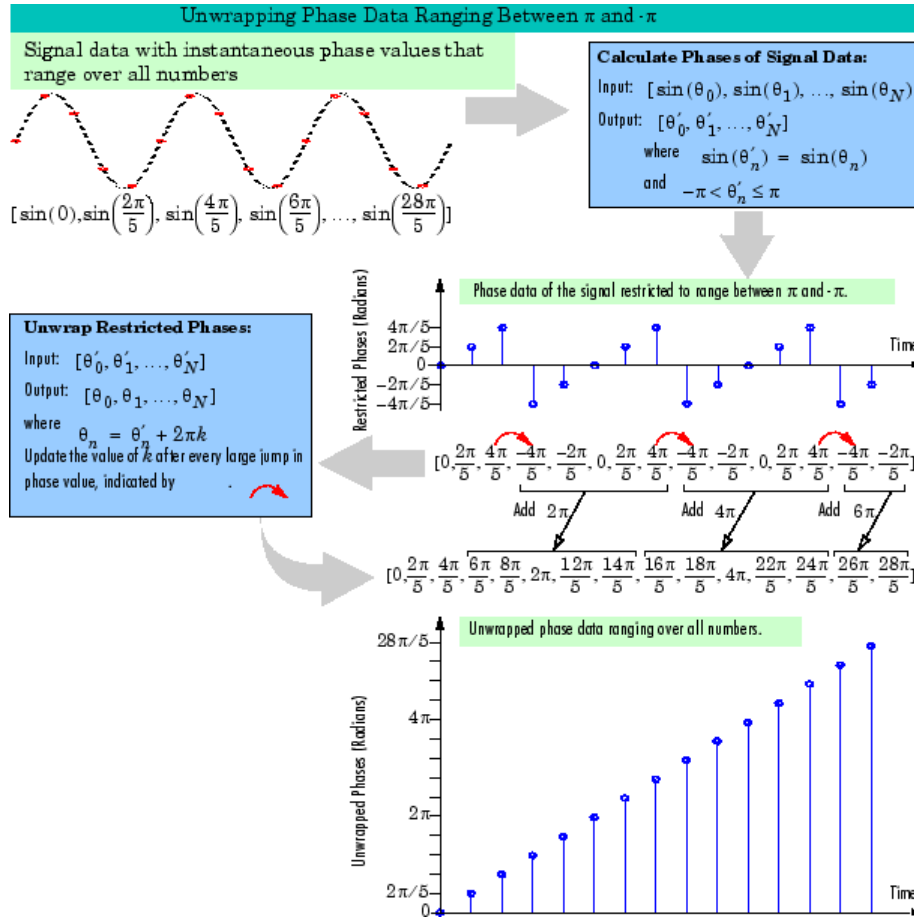
Relevant Unwrap Terms:
• u_i — ith element of the input channel on which the algorithm operates • α —Tolerance parameter value • Phase jump or phase discontinuity — difference between phase values of two adjacent channel entries that exceeds α. The diagram in the next section indicates p hase jumps with red arrows.
Steps to the Unwrap Method:
1 Set k to 0 (See “The Two Unwrap Modes” on page 2-1331 for more on how often this step occurs.) 2 Check for a phase jump between adjacent channel elements u_i and u_{i+1}: • When there is no phase jump between u_i and u_{i+1} $\left(\lvert u_{i+1} - u_i \rvert \le \lvert \alpha \rvert\right)$, add $2\pi k$ to u_i, and then repeat step 2 to continue checking for phase jumps. • When there is a phase jump between u_i and u_{i+1} $\left(\lvert u_{i+1} - u_i \rvert > \lvert \alpha \rvert\right)$, add $2\pi k$ to u_i, and then go to step 3 to update k. 3 Update k as follows when there is a phase jump between u_i and u_{i+1}. Then go back to step 2 to add the updated $2\pi k$ value to u_{i+1} and succeeding channel elements until the next phase jump: • When $u_{i+1} < u_i$ (phase jump is negative), increment k. • When $u_{i+1} > u_i$ (phase jump is positive), decrement k.

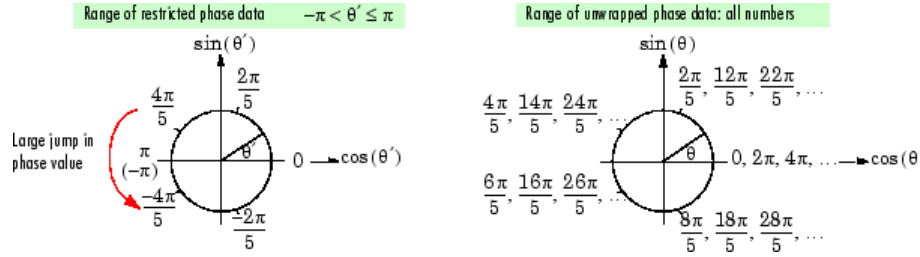
Definition of Phase Unwrap

Algorithms that compute the phase of a signal often only output phases between $-\pi$ and π. For instance, such algorithms compute the phase of $\sin(2\pi + 3)$ to be 3, since $\sin(3) = \sin(2\pi + 3)$, and since the actual phase,

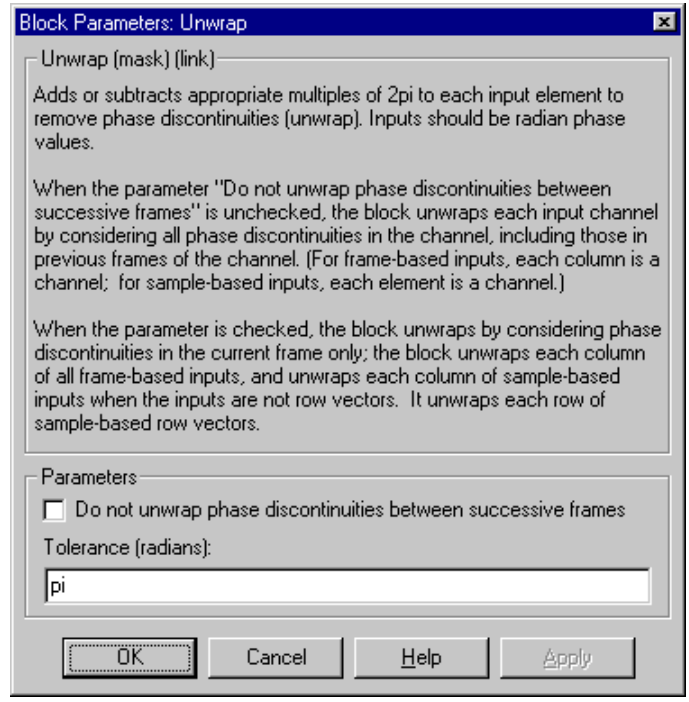
2π + 3, is not between – π and π. Such algorithms compute the phases of sin(–4π + 3) and sin(16 π + 3) to be 3 as well.

Phase unwrap or unwrap is a process often used to reconstruct a signal's original phase. Unwrap algorithms add appropriate multiples of 2 π to each phase input to restore original phase values, as illustrated in the following diagram. For more on phase unwrap, see the previous section, "Unwrap Method" on page 2-1334.





Dialog Box



Do not unwrap phase discontinuities between successive frames

When this parameter is cleared, the block unwraps each input's channels (the input channels are the columns of frame-based inputs and each element of sample-based inputs). When you select this parameter, the block unwraps each row of sample-based

row vector inputs, and unwraps the columns of all other inputs, where each input matrix or input vector is treated as completely unrelated to the other input matrices or input vectors. 1-D vector inputs are always treated as frame-based column vectors. See “The Two Unwrap Modes” on page 2-1331.

Tolerance

The jump size that the block recognizes as a true phase discontinuity. The default is set to π (rather than a smaller value) to avoid altering legitimate s ignal features. To increase the block’s sensitivity, set Tolerance to a value slightly less than π.

Supported Data Types

- Double-precision ßoating point
- Single-precision ßoating point

See Also

unwrap MATLAB

Purpose

Resample input at higher rate by inserting zeros

Library

Signal Operations

dspsigops

Description

The Upsample block resamples each channel of the M_i-by-N input at a rate L times higher than the input sample rate by inserting L-1 zeros between consecutive samples. You specify the integer L in the Upsample factor parameter. The Sample offset parameter delays the output samples by an integer number of sample periods D, where

$0 \le D < (L-1)$, so that any of the L possible output phases can be selected.

This block supports trigg ered subsystems if, for Frame-based mode , you select Maintain input frame rate .

Sample-Based Operation

When the input is sample based, the block treats each of the M*N matrix elements as an independ ent channel, and upsamples each channel over time. The Frame-based mode parameter must be set to Maintain input frame size . The output sample rate is L times higher than the input sample rate ($T_{so} = T_{si}/L$), and the input and output sizes are identical.

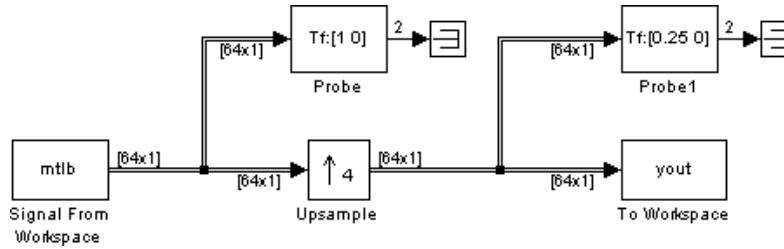
Frame-Based Operation

When the input is frame based, the block treats each of the N input columns as a frame containing M_i sequential time samples from an independent channel. The block upsamples each channel independently by inserting L-1 rows of zeros bet ween each row in the input matrix. The Frame-based mode parameter determines how the block adjusts the rate at the output to accommodate the added rows. There are two available options:

- Maintain input frame size

The block generates the output at the faster (upsampled) rate by using a proportionally shorter frame period at the output port than at the input port. For upsampling by a factor of L, the output frame period is L times shorter than the input frame period ($T_{fo} = T_p/L$), but the input and output frame sizes are equal.

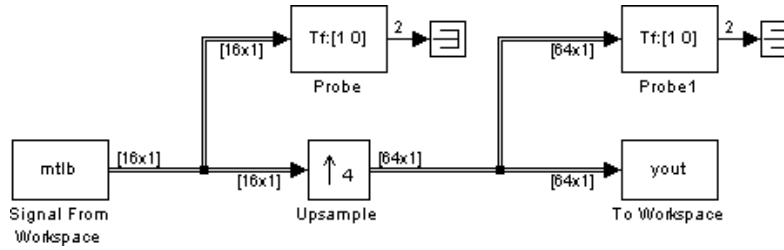
The model below shows a single-channel input with a frame period of 1 second being upsampled by a factor of 4 to a frame period of 0.25 second. The input and output frame sizes are identical.



- Maintain input frame rate

  The block generates the output at the faster (upsampled) rate by using a proportionally larger frame size than the input. For upsampling by a factor of L, the output frame size is L times larger than the input frame size ($M_o = M_i*L$), but the input and output frame rates are equal.

  The model below shows a single-channel input of frame size 16 being upsampled by a factor of 4 to a frame size of 64. The input and output frame rates are identical.



Zero Latency

The Upsample block has zero-tasking latency for all single-rate operations. The block is single-rate for the particular combinations of sampling mode and parameter settings shown in the table below.

Sampling Mode	Parameter Settings
Sample based	Upsample factor parameter, L, is 1.
Frame based	Upsample factor parameter, L, is 1, or Frame-based mode parameter is Maintain input frame rate.

The block also has zero latency for all multirate operations in the Simulink single-tasking mode.

Zero-tasking latency means that the block propagates the first input (received at t=0) immediately following the D consecutive zeros specified by the Sample offset parameter. This output (D+1) is followed in turn by the L-1 inserted zeros and the next input sample. The Initial condition parameter value is not used.

Nonzero Latency

The Upsample block has tasking latency only for multirate operation in the Simulink multitasking mode:

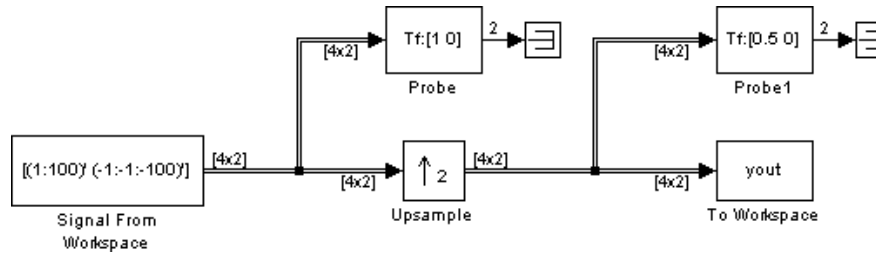
- In sample-based mode, the initial condition for each channel appears as output sample D+1, and is followed by L-1 inserted zeros. The channel's first input appears as output sample D+L+1. The Initial condition value can be an M_i-by-N matrix containing one value for each channel, or a scalar to be applied to all signal channels.
- In frame-based mode, the first row of the initial condition matrix appears as output sample D+1, and is followed by L-1 inserted rows of zeros, the second row of the initial condition matrix, and so on. The first row of the first input matrix appears in the output as sample M_iL+D+1. The Initial condition value can be an M_i-by-N matrix, or

a scalar to be repeated across all elements of the M ¡-by-N matrix. See the example below for an illustration of this case.

Note For more information on latency and the Simulink tasking modes, see “Excess Algorithmic Delay (Tasking Latency)” and “Models with Multiple Sample Rates” in the Real-Time Workshop User’s Guide .

Examples

Construct the frame-bas ed model shown below.



Adjust the block parameters as follows:

- ConÞgure the Signal From Workspace block to generate a two-channel signal with frame si ze of 4 and sample period of 0.25. This represents an output frame period of 1 (0.25*4). The Þrst channel should contain the positive ramp signal 1, 2, ..., 100, and the second channel should contain the negative ramp signal -1, -2, ..., -100.
  - Signal = [(1:100)' (-1:-1:-100)']
  - Sample time = 0.25
  - Samples per frame = 4
- ConÞgure the Upsample block to upsa mple the two-channel input by increasing the output frame rate by a factor of 2 relative to the input frame rate. Set a sample offset of 1, and an initial condition matrix of

$$\begin{bmatrix} 11 & -11 \\ 12 & -12 \\ 13 & -13 \\ 14 & -14 \end{bmatrix}$$

- Upsample factor = 2
- Sample offset = 1
- Initial condition = [11 -11;12 -12;13 -13;14 -14]
- Frame-based mode = Maintain input frame size

• Conþgure the Probe blocks by clearing the Probe width and Probe complex signal check boxes (if desired).

This model is multirate because the re are at least two distinct frame rates, as shown by the two Probe blocks. To run this model in the Simulink multitasking mode, open the Con þguration Parameters dialog box. In the Select pane, click Solver . From the Type list, select Fixed-step , and from the Solver list, select Discrete (no continuous states) . From the Tasking mode for periodic sample times list, select MultiTasking . Also set the Stop time to 30.

Run the model and look at the output, yout . The þrst few samples of each channel are shown below.

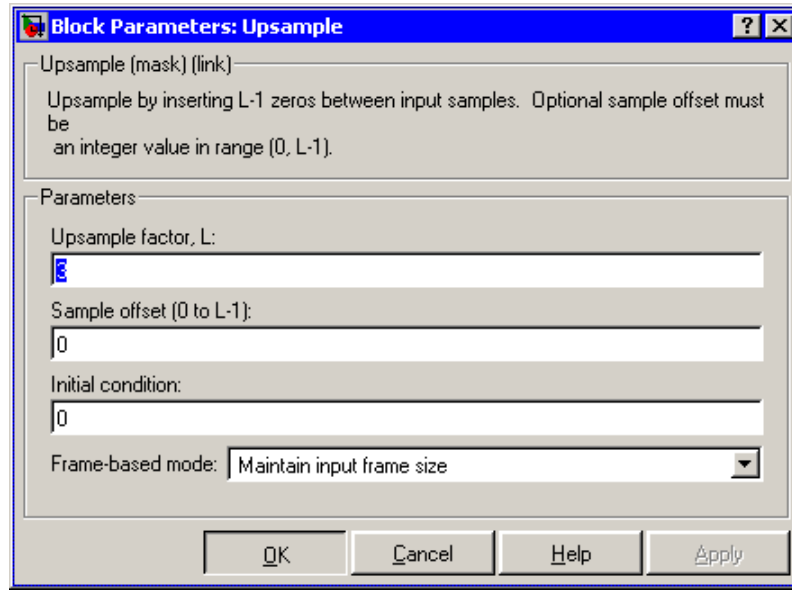
```
yout =

     0     0
    11   -11
     0     0
    12   -12
     0     0
    13   -13
     0     0
    14   -14
     0     0
     1    -1
```

```
0     0
2    -2
0     0
3    -3
0     0
4    -4
0     0
5    -5
0     0
```

Since we ran this frame-based multi rate model in multitasking mode, the Þrst row of the initial condition m atrix appears as output sample 2 (that is, sample D+1, where D is the Sample offset value). It is followed by the other three initial co ndition rows, each separated by L-1 inserted rows of zeros, where L is the Upsample factor value of 2. The Þrst row of the Þrst input matrix appears in the output as sample 10 (that is, sample $M_i L+D+1$, where M_i is the input frame size).

Dialog Box



Upsample factor
: The integer factor, L, by which to increase the input sample rate.

Sample offset
: The sample offset, D, which must be an integer in the range [0,L-1].

Initial condition
: The value with which the block is initialized for cases of nonzero latency, a scalar or matrix. This value (Þrst row in frame-based mode) appears in the output as sample D+1.

Frame-based mode
: For frame-based operation, the method by which to implement the upsampling: Maintain input frame size (that is, increase the frame rate), or Maintain input frame rate (that is, increase the frame size). The Framing parameter must be set to Maintain input frame size for sample-base inputs.

Supported Data Types

Port	Supported Data Types
Input	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision ßoating point • Single-precision ßoating point • Fixed point (signed and unsigned) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

See Also

Downsample	Signal Processing Blockset
FIR Interpolation	Signal Processing Blockset
FIR Rate Conversion	Signal Processing Blockset
Repeat	Signal Processing Blockset

Purpose

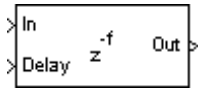
Delay input by time-varying fractional number of sample periods

Library

Signal Operations

dspsigops

Description



The Variable Fractional Delay block delays each element of the discrete-time N-D input array, u, by a variable number of sample intervals. The input delay values can be integer or noninteger values. The block provides three different interpolation modes: Linear, FIR, and Farrow.

The block computes the value for each channel of the output based on the stored samples in memory most closely indexed by the Delay input, v, and the interpolation method specified by the Interpolation mode parameter.

- In Linear interpolation mode, the block stores the $D_{max}+1$ most recent samples received at the In port for each channel, where D_{max} is the value specified for the Maximum delay (Dmax) in samples parameter.
- In FIR interpolation mode, the block stores the $D_{max}+P+1$ most recent samples received at the In port for each channel, where P is the value specified for the Interpolation filter half-length (P) parameter.
- In Farrow interpolation mode, the block stores the $D_{max}+\frac{N}{2}+1$ most recent samples received at the In port for each channel, where N is the value specified for the Farrow filter length (N) parameter.

The Variable Fractional Delay block assumes that the input values at the Delay port are between D_{min} and D_{max}, where D_{min} appears in the Valid delay range section on the Main pane of the block mask, and D_{max} is the value of the Maximum delay (Dmax) in samples parameter. The block clips delay values less than D_{min} to D_{min} and delay values greater than D_{max} to D_{max}.

If you are working with frame-based signals and select the Disable direct feedthrough by increasing the minimum possible delay by one check box, the smallest possible delay value is increased by *frame-size* – 1. Thus, all input delay values less than Dmin + *frame-size* – 1 are clipped to Dmin + *frame-size* – 1.

You must consider additional factors when selecting valid Delay values for the FIR and Farrow interpolation modes. For more information about these considerations, refer to FIR Interpolation Mode on page 1350 or Farrow Interpolation Mode on page 1352, respectively.

The Variable Fractional Delay block is similar to the Variable Integer Delay block, in that they both store a minimum of $D_{max}+1$ past samples in memory. The Variable Fractional Delay block differs only in the way that these stored samples are accessed; a fractional delay requires the computation of a value by interpolation from the nearby samples in memory.

Sample-Based Operation

For sample-based inputs, the block treats each element of the N-D input array, u, as an independent channel. The input to the Delay port, v, must either be an N-D array of the same size and dimension as the input u, or be a scalar value, such that $D_{min} \le v \le D_{max}$.

For example, consider an M-by-N input matrix. The block treats each of the M*N matrix elements as independent channels. The input to the Delay port can be an M-by-N matrix of floating-point values in the range $D_{min} \le v \le D_{max}$ that specifies the number of sample intervals to delay each channel of the input, or it can be a scalar floating-point value, $D_{min} \le v \le D_{max}$, by which to equally delay all channels.

The block treats a 1-D vector input as an M-by-1 matrix, and outputs a 1-D vector.

The Initial conditions parameter specifies the values in the block's memory at the start of the simulation in the same manner as the Variable Integer Delay block. See the Variable Integer Delay block reference page for more information.

Frame-Based Operation

For frame-based inputs, the block treats each of the N input columns as a frame containing M_i sequential time samples from an independent channel.

The input to the Delay port, v, contains floating-point values that specify the number of sample intervals to delay the current input. When you clear the Disable direct feedthrough by increasing minimum possible delay by one check box, the valid range of delay values is $D_{min} \le v \le D_{max}$. When you select the Disable direct feedthrough by increasing minimum possible delay by one check box, the valid range of delay values is $D_{min} + \mathit{frame\text{-}size} - 1 \le v \le D_{max}$.

The input to the Delay port can be a scalar value to uniformly delay every sample in every channel. It can also be a column-based length-M vector, containing one delay for each sample in the input frame. The block applies the set of delays contained in the vector identically to every channel of a multichannel input. The Delay port entry can also be a row-based length-N vector, containing one delay for each channel. Finally, the Delay port entry can be an M-by-N matrix, containing a different delay for each corresponding element of the input.

For example, if v is the M_i-by-1 matrix [v(1) v(2) ... v(Mi)]', the earliest sample in the current frame is delayed by v(1) fractional sample intervals, the following sample in the frame is delayed by v(2) fractional sample intervals, and so on. The block applies the set of fractional delays contained in v identically to every channel of a multichannel input.

The Initial conditions parameter specifies the values in the block's memory at the start of the simulation in the same manner as the Variable Integer Delay block. See the Variable Integer Delay block reference page for more information.

Interpolation Modes

The delay value specified at the Delay port serves as an index into the block's memory, U, which stores, at a minimum, the D_{max}+1 most recent samples received at the In port for each channel. For example, an

integer delay of 5 on a scalar input sequence retrieves and outputs the Þfth most recent input sample from the block's memory, U(6) . The block computes fractional delays by inte rpolating between stored samples; the three available interpolation modes are Linear , FIR and Farrow .

Linear Interpolation Mode

For noninteger delays, at each sample time, the Linear Interpolation mode uses the two samples in memory nearest to the speci Þed delay to compute a value for the sample at that time. If v is the speci Þed fractional delay for a scalar input, the output sample, y, is computed as follows.

```
vi = floor(v)        % vi = integer delay
vf = v-vi            % vf = fractional delay
y = (1-vf)*U(vi+1) + vf*U(vi)
```

FIR Interpolation Mode

In FIR Interpolation mode, the block provides a discrete set of fractional delays described by:

$$v + \frac{i}{L}, \qquad v \geq P-1, \;\; i = 0,1,...,L-1$$

If v is less than P-1, the block's behavior depends on the setting of the For small input delay values parameter. You can specify the block's behavior when the input delay value is too small to center the kernel (less than P-1), by setting the For small input delay values parameter:

- If you select Clip to the minimum value necessary for centered kernel , the block remains in FIR interpolation mode by clipping small input delay values to the smallest value necessary to center the kernel.

  To determine the minimum delay value, select Clip to the minimum value necessary for centered kernel , and click Apply on the block mask. All input delay values less than the value displayed for D_{min} will be clipped to D_{min}.

- If you select Switch to linear interpolation if kernel cannot be centered , the block computes fractional delays using linear interpolation when the inpu t delay value is less than P-1.

To add an extra delay to the minimum possible delay value, select the Disable direct feedthrough by increasing minimum possible delay by one check box. Checking this box prevents algebraic loops from occurring when you use the b lock inside a feedback loop.

Note If the input to the block is fr ame based and you select the Disable direct feedthrough by increasing minimum possible delay by one check box, the minimum possible delay (Dmin) increases by *frame-size* – 1.

In FIR Interpolation mode, the block implements a polyphase structure to compute a value for each sample at the desired delay. Each arm of the structure corresponds to a different delay value and the output computed for each sample corresponds to the output of the arm with a delay value nearest to the desired input delay. Thus, only a discrete set of delays is actua lly possible. The number of coef Þcients in each of the L Þlter arms of the polyphase structure is 2 P. In most cases, using values of P between 4 and 6 will provide you with reasonably accurate interpolation values.

In this mode, the Signal Processing Toolbox intfilt function computes an FIR Þlter for interpolation.

For example, when you set the parameters on the block mask to the following values:

- Interpolation Þlter half-length (P): 4
- Interpolation points per input sample: 10
- Normalized input bandwidth: 1

The Þlter coef Þcients are given by:

```
b = intfilt(10,4,1);
```

The block then implements this filter as a polyphase structure, as described previously.

Increasing the Interpolation filter half length (P) increases the accuracy of the interpolation, but also increases the number of computations performed per input sample, as well as the amount of memory needed to store the filter coefficients. Increasing the Interpolation points per input sample (L) increases the number of representable discrete delay points, but also increases the simulation's memory requirements and does not affect the computational load per sample.

The Normalized input bandwidth (0 to 1) parameter allows you to take advantage of the bandlimited frequency content of the input. For example, if you know that the input signal does not have frequency content above $F_s/4$, you can specify a value of 0.5 for the Normalized input bandwidth (0 to 1) to constrain the frequency content of the output to that range.

Note You can consider each of the L interpolation filters to correspond to one output phase of an "upsample-by- L" FIR filter. Thus, the Normalized input bandwidth (0 to 1) value improves the stopband in critical regions, and relaxes the stopband requirements in frequency regions where there is no signal energy.

Farrow Interpolation Mode

In Farrow interpolation mode, the block uses the LaGrange method to interpolate values.

To add an extra delay of 1 to the minimum possible delay value, select the Disable direct feedthrough by increasing minimum possible delay by one check box. Checking this box prevents algebraic loops from occurring when you use the block inside a feedback loop.

Note If the input to the block is frame-based and you select the Disable direct feedthrough by increasing minimum possible delay by one check box, the minimum possible delay (Dmin) increases by *frame-size* – 1.

To specify the block's behavior when the input delay value is too small to center the kernel (less than $\frac{N}{2}$ -1), set the For small input delay values parameter:

- If you select Clip to the minimum value necessary for centered kernel , the block clips small input delay values to the smallest value necessary to keep th e kernel centered. This increases D_{min} but yields more accurate interpolation values.

  To determine the minimum delay value, select Clip to the minimum value necessary for centered kernel , and click Apply on the block mask. All input delay values less than the value displayed for D_{min} will be clipped to D_{min}.

- If you select Use off-centered kernel , the block computes fractional delays using a Farrow Þlter with an off-centered kernel. This mode does not increase D_{min}, but if there are input delay values less than $\frac{N}{2}$ -1, the results are less accura te than the results achieved by keeping the kernel centered.

Fixed-Point Data Types

The diagrams in the following secti ons show the data types used within the Variable Fractional Delay block for Þxed-point signals.

Although you can specify most of the se data types on the Fixed-point pane of the block mask, the following data types are computed internally by the block and cannot be directly speci Þed on the block mask.

Data Type	Word Length	Fraction Length
vf data type	Same word length as the Coefficients	Same as the word length
HoldInteger data type	Same word length as the input delay value	0 bits
Integer data type	32 bits	0 bits

To compute the integer (v_i) and fractional (v_f) parts of the input delay value (v), the Variable Fractional Delay block uses the following equations:

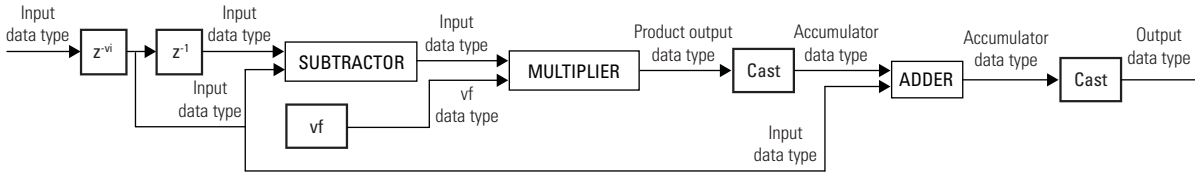
$$D_{min} < v < D_{max} \Rightarrow \begin{cases} v_i = \text{floor}(v) \\ v_f = v - vi \end{cases}$$

$$v \leq D_{min} \Rightarrow \begin{cases} v_i = D_{min} \\ v_f = 0 \end{cases}$$

$$v \geq D_{max} \Rightarrow \begin{cases} v_i = D_{max} \\ v_f = 0 \end{cases}$$

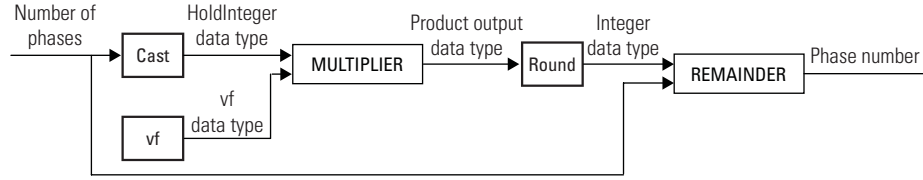
Linear Interpolation Mode

The following diagram shows the fixed-point data types used by the Linear interpolation mode of the Variable Fractional Delay block.



FIR Interpolation Mode

The following diagram illustrates how the Variable Fractional Delay block selects the arm of the polyphase filter structure that most closely matches the fractional delay value (v_f).

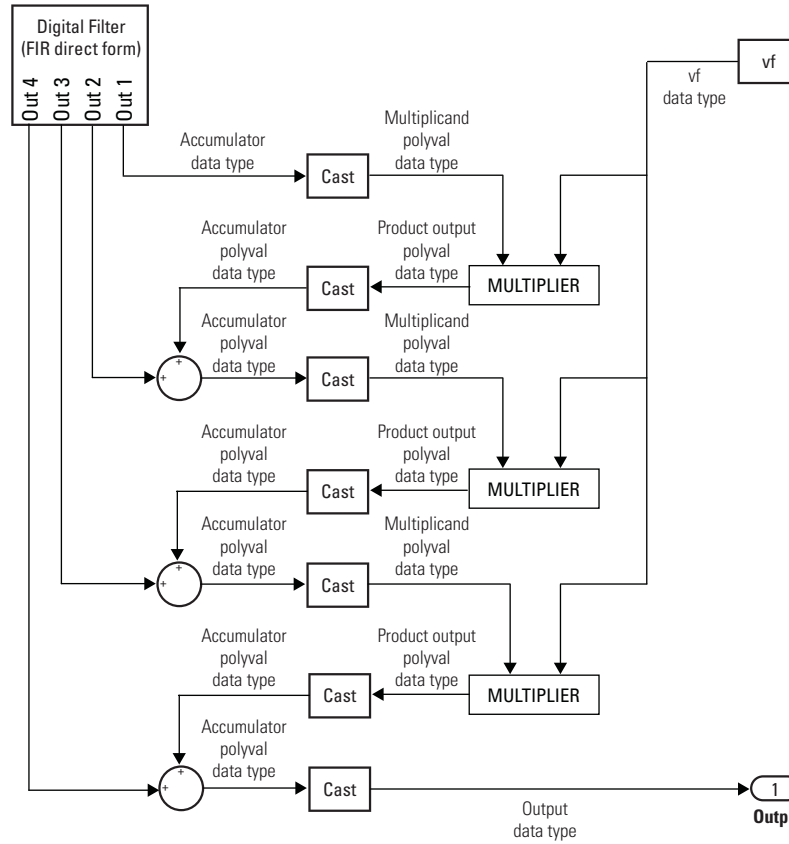


See the "Fixed-Point Data Types" on page 2-527 section of the FIR Interpolation reference page for a diagram showing the fixed-point data types used by the Variable Fractional Delay block in FIR interpolation mode.

Farrow Interpolation Mode

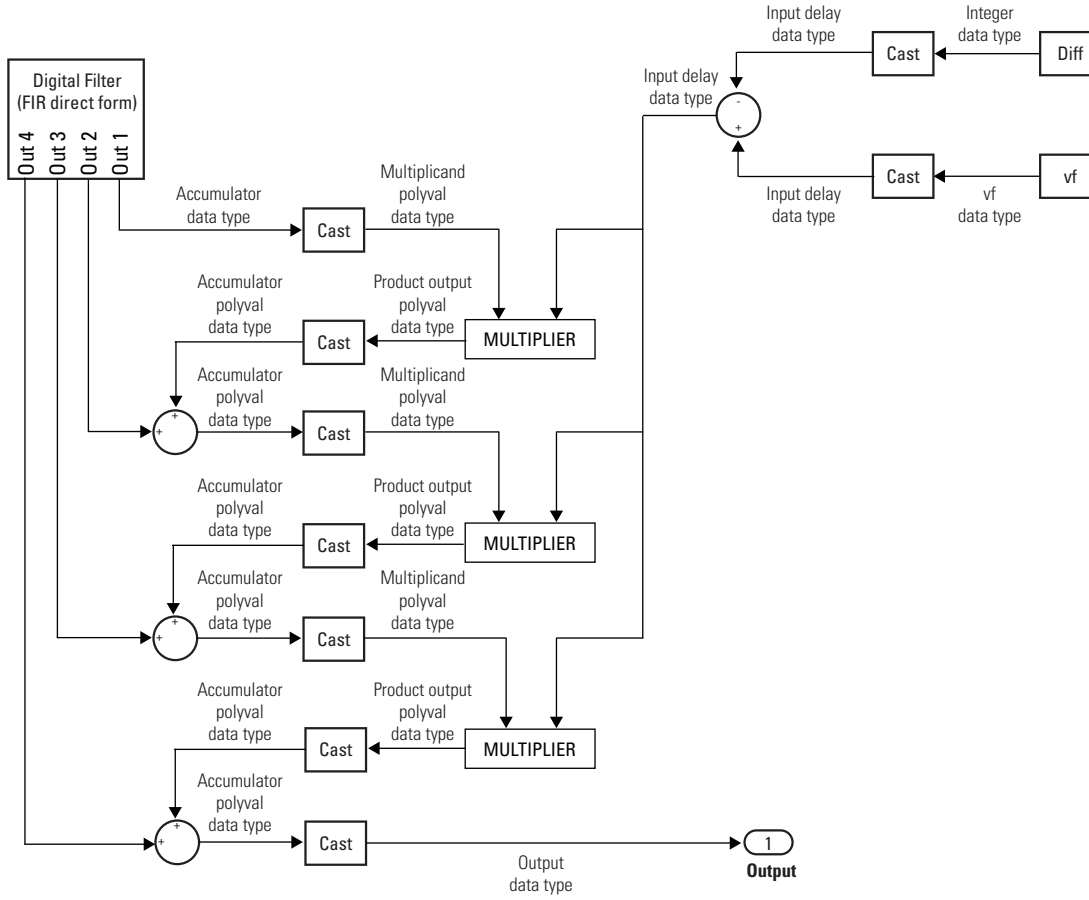
The following diagram shows the fixed-point data types used by the Farrow interpolation mode of the Variable Fractional Delay block where:

- Farrow filter length (N) = 4
- For small input delay values = Clip to the minimum value necessary to for centered kernel



The following diagram shows the fixed-point data types used by the Farrow interpolation mode of the Variable Fractional Delay block where:

- Farrow filter length (N) = 4
- For small input delay values = Use off-centered kernel



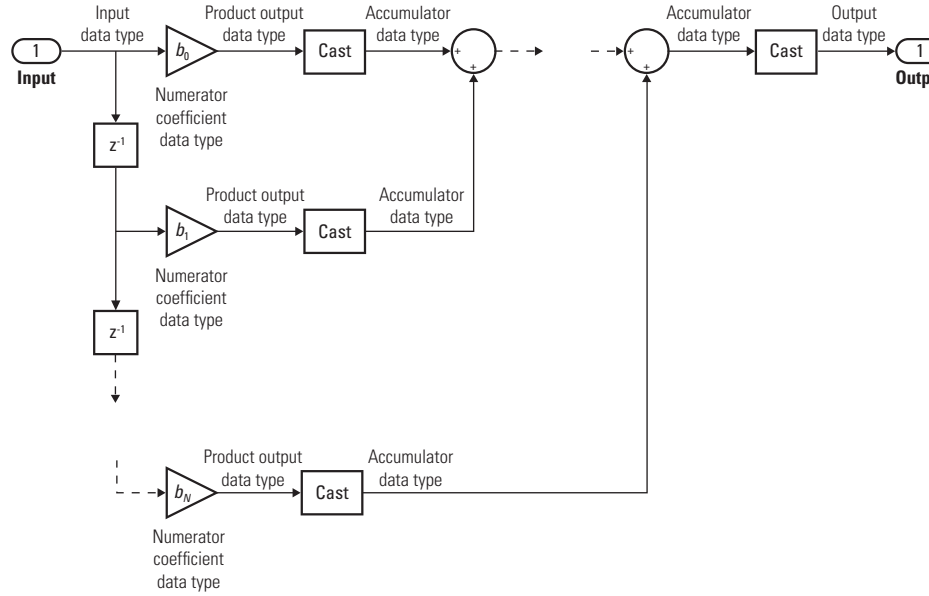
Diff is computed from the integer part of the delay value (v_i) and the Farrow filter length (N) according to the following equation:

$$Diff = v_i - \left(\frac{N-1}{2}\right)$$

$$Diff \geq 0 \Rightarrow Diff = 0$$

$$Diff < 0 \Rightarrow Diff = -Diff$$

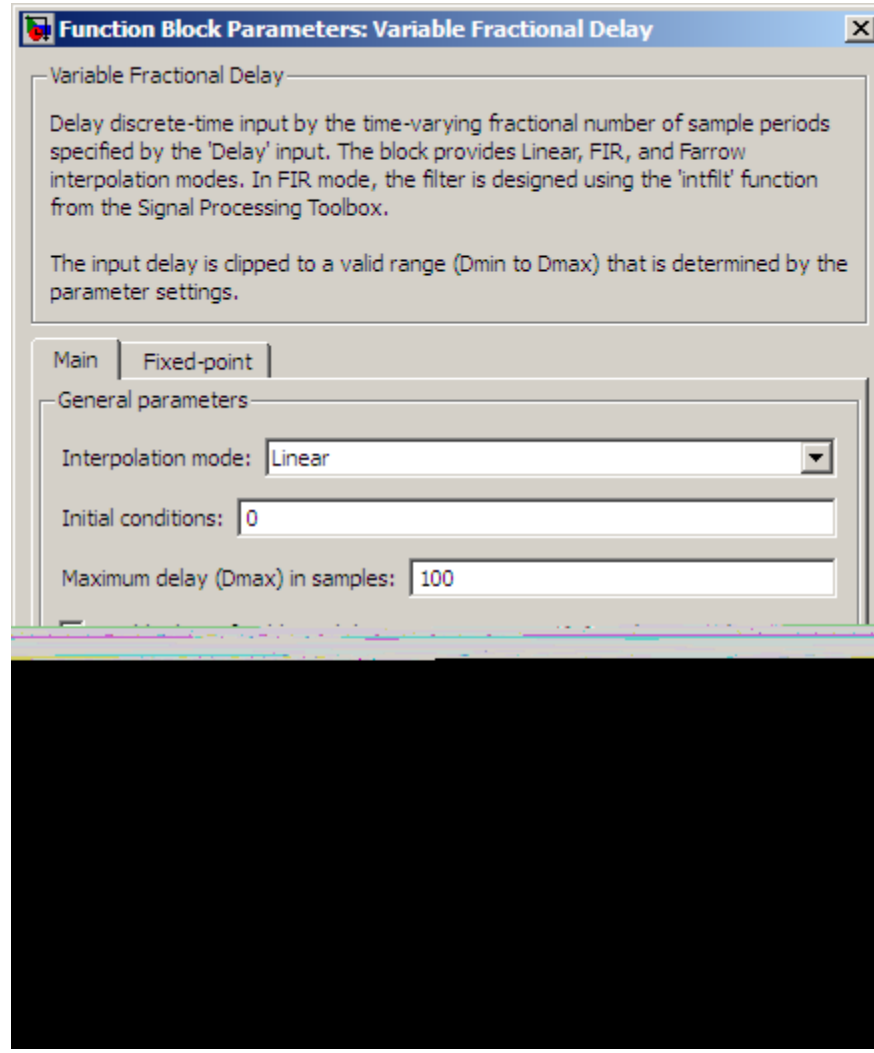
The following diagram shows the fixed-point data types used by the Digital Filter block's FIR direct form filter.



Examples

The dspaudioeffects demo illustrates three audio effects applied to a short segment of music. When you set the Audio effect of the Effect block to Flanging, the model uses the Variable Fractional Delay block to mix the original signal with a delayed version of itself.

To see the Flanging subsystem, right-click the Effect block, and select Look Under Mask. Next, double-click the Flanging block in the Effect block subsystem that just opened. The Flanging subsystem opens, and you can see the parameters of the Variable Fractional Delay block.

Dialog Box

The Main pane of the Variable Fractional Delay block dialog appears as follows.