Filter Design Toolbox Release Notes

The "Filter Design Toolbox 2.5 Release Notes" on page 1-1 describe the changes introduced in the latest version of the Filter Design Toolbox (Version 2.5), which is available via Web download. The following topics are discussed in these Release Notes:

The Filter Design Toolbox Release Notes also provide information about the earlier versions of the product, in case you are upgrading from a version that was released prior to Release 13. If you are upgrading from a release earlier than Release 13, you should also see these sections:

Printing the Release Notes

If you would like to print the Release Notes, you can link to a PDF version.

Contents

1 Filter Design Toolbox 2.5 Release Notes

2 Filter Design Toolbox 2.2 Release Notes

3 Filter Design Toolbox 2.1 Release Notes

4 Filter Design Toolbox 2.0 Release Notes

1

Filter Design Toolbox 2.5 Release Notes

New Features

This section introduces the new features and enhancements added to the Filter Design Toolbox 2.5 since Version 2.2 (Release 13).

If you are upgrading from a release earlier than Release 13, you should read "New Features" on page 2-2 in the Filter Design Toolbox 2.2 Release Notes.

New Object-Based Adaptive Filters

One of the most important additions to the toolbox is new filter objects for applying adaptive filters to data. Like qfilt objects, you construct the object using the appropriate constructor, and then apply your new object to your data. With more than 30 new adaptive filter types, we group them here by the fundamental algorithm each uses for adaptation.

Algorithms

For adaptive filter (adaptfilt) objects, the *`algorithm`* string determines which adaptive filter algorithm your adaptfilt object implements. Each available algorithm entry appears in one of the following tables along with a brief description of the algorithm. Click on the algorithm in the first column to get more information about the associated adaptive filter technique.

Least Mean Squares (LMS) Based FIR Adaptive Filters

adaptfilt.algorithm String	Description of the Adapting Algorithm Used to Generate Filter Coefficients During Adaptation
`adaptfilt.adjlms`	Use the adjoint LMS FIR adaptive filter algorithm
`adaptfilt.blms`	Use the block LMS FIR adaptive filter algorithm
`adaptfilt.blmsfft`	Use the FFT-based block LMS FIR adaptive filter algorithm
`adaptfilt.dlms`	Use the delayed LMS FIR adaptive filter algorithm
`adaptfilt.filtxlms`	Use the filtered-x LMS FIR adaptive filter algorithm
`adaptfilt.lms`	Use the LMS FIR adaptive filter algorithm
`adaptfilt.nlms`	Use the normalized LMS FIR adaptive filter algorithm
`adaptfilt.sd`	Use the sign-data LMS FIR adaptive filter algorithm
`adaptfilt.se`	Use the sign-error LMS FIR adaptive filter algorithm
`adaptfilt.ss`	Use the sign-sign LMS FIR adaptive filter algorithm

For further information about an adapting algorithm, refer to the reference page for the algorithm.

Recursive Least Squares (RLS) Based FIR Adaptive Filters

adaptfilt.algorithm String	Description of the Adapting Algorithm Used to Generate Filter Coefficients During Adaptation
`adaptfilt.ftf`	Use the fast transversal least squares adaptation algorithm
`adaptfilt.qrdrls`	Use the QR-decomposition RLS adaptation algorithm
`adaptfilt.hrls`	Use the householder RLS adaptation algorithm
`adaptfilt.hswrls`	Use the householder sliding window SWRLS adaptation algorithm
`adaptfilt.rls`	Use the recursive least squares (RLS) adaptation algorithm
`adaptfilt.swrls`	Use the sliding window (SW) RLS adaptation algorithm
`adaptfilt.swftf`	Use the sliding window FTF adaptation algorithm

For more complete information about an adapting algorithm, refer to the reference page for the algorithm.

Affine Projection (AP) FIR Adaptive Filters

adaptfilt.algorithm String	Description of the Adapting Algorithm Used to Generate Filter Coefficients During Adaptation
`adaptfilt.ap`	Use the affine projection (AP) algorithm that uses direct matrix inversion
`adaptfilt.apru`	Use the affine projection (AP) algorithm that uses recursive matrix updating
`adaptfilt.bap`	Use the block affine projection (AP) adaptation algorithm

To find more information about an adapting algorithm, refer to the reference page for the algorithm.

FIR Adaptive Filters in the Frequency Domain (FD)

adaptfilt.algorithm String	Description of the Adapting Algorithm Used to Generate Filter Coefficients During Adaptation
`adaptfilt.fdaf`	Use the frequency domain (FD) adaptation algorithm
`adaptfilt.pbfdaf`	Use the partition block version of the FDAF algorithm
`adaptfilt.pbufdaf`	Use the partition block unconstrained version of the FDAF algorithm
`adaptfilt.tdafdct`	Use the transform domain adaptation algorithm using DCT
`adaptfilt.tdafdft`	Use the transform domain adaptation algorithm using DFT
`adaptfilt.ufdaf`	Use the unconstrained FDAF algorithm for adaptation

For more information about an adapting algorithm, refer to the reference page for the algorithm.

Lattice Based (L) FIR Adaptive Filters

adaptfilt.algorithm String	Description of the Adapting Algorithm Used to Generate Filter Coefficients During Adaptation
`adaptfilt.gal`	Use the gradient adaptive lattice filter adaptation algorithm
`adaptfilt.lsl`	Use the least squares lattice adaptation algorithm
`adaptfilt.qrdlsl`	Use the QR decomposition LSL adaptation algorithm

For more information about an adapting algorithm, refer to the reference page for the algorithm.

New Object-Based Multirate Filters

Along with new adaptive filter object, Version 2.5 adds many multirate filter objects as well. In addition to the existing `cicdecimate` and `cicinterpolate` functions already available, you now have the following multirate filtering objects at hand, as shown in this table.

Multirate Filter Object	Filter Object Description
`mfilt.cicdecim`	Construct a cascaded integrator-comb decimation filter object
`mfilt.cicdecimzerolat`	Construct a cascaded integrator-comb decimation filter object that does not display latency
`mfilt.cicinterp`	Construct a cascaded integrator-comb interpolation filter object
`mfilt.cicinterpzerolat`	Construct a cascaded integrator-comb interpolation filter object that does not display latency

Multirate Filter Object	Filter Object Description
`mfilt.fftfirinterp`	Construct an overlap-add FIR polyphase interpolation filter object
`mfilt.firdecim`	Construct a direct-form FIR polyphase decimation filter object
`mfilt.firfracdecim`	Construct a direct-form FIR polyphase fractional decimation filter object
`mfilt.firfracinterp`	Construct a direct-form FIR polyphase fractional interpolation filter object
`mfilt.firinterp`	Construct a direct-form FIR polyphase interpolation filter object
`mfilt.firsrc`	Construct a direct-form FIR polyphase sample rate conversion filter object
`mfilt.firtdecim`	Construct a direct-form, transposed FIR polyphase decimation filter object
`mfilt.holdinterp`	Construct an FIR interpolation filter object that uses “hold” interpolation between input samples
`mfilt.linearinterp`	Construct an FIR linear interpolation filter object that applies linear interpolation between input samples

Updated and New Methods for Adaptive Filter and Multirate Filter Objects

To enable you to work with the new filter objects in the toolbox, Version 2.5 includes the following new methods that support adaptive filter and multirate

filter objects. In addition, existing filter analysis methods such as `freqz` work with the new objects just as they do for qfilt and dfilt objects.

Method Name	Supported Objects	Description
`block`	Multirate (some)	Generate a DSP Blockset block that duplicates the filter object. Works only when DSP Blockset is installed. Some multirate objects cannot be modelled as blocks since the blockset does not provide blocks for all the multirate filters in the toolbox.
`coefficients`	Multirate	Return the multirate filter objects coefficients.
`euclidfactors`	Multirate	Use Euclid's theorem to determine the integer factors for an `mfilt` object.
`msepred`	Adaptive filter	Return the predicted mean-square error for the `adaptfilt` object.
`msesim`	Adaptive filter	Return the measured mean-square error for the `adaptfilt` object via simulation.
`maxstep`	Adaptive filter	Return the maximum step size for an `adaptfilt` object.
`nstates`	Multirate filter	Return the number of states in an `mfilt` object.
`polyphase`	Multirate filter	Return the polyphase matrix for an `mfilt` object.

To see the full listing of analysis methods that apply to the new adaptfilt and mfilt objects, type `help adaptfilt` and `help mfilt` at the MATLAB prompt.

The Filter Visualization Tool (FVTool) supports these new objects so you can use the full power of FVTool to analyze the objects you create.

New Demos for Adaptive Filters and Multirate Filters

To help you learn about the new objects in the toolbox, Filter Design Toolbox 2.5 includes many new demos to introduce the new adaptive and multirate objects. To see the new demos, select **Demos** from the **Help** menu in MATLAB. In the Help browser, look under **Toolboxes** for the **Filter Design** entry.

Import XILINX Coefficient (.COE) Files with FDATool

Version 2.2 of the toolbox introduced the ability to read and write XILINX coefficients files from the MATLAB command line with `coeread` and `coewrite`. Now you can import XILINX coefficients (.COE) files into FDATool to create quantized filters directly using the imported filter coefficients.

To use the new import feature:

1. Select **File->Import Filter From XILINX Coefficient (.COE) File** in FDATool.

2. In the **Import Filter From XILINX Coefficient (.COE) File** dialog, find and select the `.coe` file to import.

3. Click **Open** to dismiss the dialog and start the import process.

   FDATool imports the coefficient file and creates a quantized, single-section, direct-form FIR filter.

Major Bug Fixes

The Filter Design Toolbox 2.5 includes several bug fixes made since Version 2.2. This section describes the particularly important Version 2.5 bug fixes.

If you are viewing these Release Notes in PDF form, please refer to the HTML form of the Release Notes, using either the Help browser or the MathWorks Web site and use the link provided.

Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving to the Filter Design Toolbox 2.5 from Version 2.2.

Previous Adaptive Filter Functions Now Obsolete

The following adaptive filter functions no longer appear in the toolbox. When you use one of these functions, you get an error suggesting that you replace your function with one of the adaptive filter objects that are new in Version 2.5.

- `adaptkalman`
- `adaptlms`
- `adaptrls`
- `adaptsd`
- `adaptse`
- `adaptss`

References to these functions have been removed from the documentation, although they will continue to work until a future release. There is no replacement object for the `adaptkalman` function.

Uninstall Previous Versions Before Installing Version 2.5

Because Version 2.5 of the toolbox relocates some files to improve performance and consistency, you must uninstall all earlier toolbox versions of the Filter Design Toolbox before you install Version 2.5. In addition, you must remove earlier versions of the Signal Processing Toolbox and install the latest version, Signal Processing Toolbox 6.1. Refer to the release notice for the Signal Processing Toolbox 6.1 for information about installing the Signal Processing Toolbox 6.1.

Caution Uninstalling the toolbox removes all the files in the toolbox directory `$MATLAB/toolbox/filterdesign`, including all files you may have stored there. Before performing the uninstall operation, make backup copies in another directory of all the files to save.

Uninstalling Previous Versions of the Filter Design Toolbox

To uninstall previous versions of the Filter Design Toolbox, perform these steps.

1. Close MATLAB. You cannot uninstall toolboxes while MATLAB is running.
2. Do one of the following depending on whether you are removing the toolbox from a Microsoft Windows® platform or a UNIX platform.
   - On Microsoft Windows platforms—use the MATLAB Uninstaller. Select **Programs->MATLAB 6.5->Uninstall MATLAB 6.5** from the **Start** Menu. Select **Filter Design Toolbox** from the **Uninstall Product List** and click **OK**.
   - On UNIX platforms, use the following command to remove the toolbox

     ```
     rm -rf $MATLAB/toolbox/filterdesign
     ```

     where `$MATLAB` in the full path name of the root directory where you installed MATLAB.

Changed Calattice and Calatticepc Dfilt Objects for Version 2.5

In this release of the toolbox, we changed the `dfilt.calattice` and `dfilt.calatticepc` filter objects. Older versions will no longer work. With the new objects, the `Allpass1` and `Allpass2` properties now store a vector of filter coefficients rather than a `dfilt.latticeallpass` object as they did in earlier versions.

Known Software and Documentation Problems

This section includes a link to a description of known software and documentation problems in Version 2.5.

If you are viewing these Release Notes in PDF form, please refer to the HTML form of the Release Notes, using either the Help browser or the MathWorks Web site and use the link provided.

If you are upgrading from a release earlier than Release 13, you should read "Known Software and Documentation Problems" on page 3-6 in the Filter Design Toolbox 2.1 Release Notes.

2

Filter Design Toolbox 2.2 Release Notes

New Features

This section introduces the new features and enhancements added to the Filter Design Toolbox 2.2 since Version 2.1 (Release 12.1).

If you are upgrading from a release earlier than Release 12.1, you should read "New Features" on page 3-2 in the Filter Design Toolbox 2.1 Release Notes.

Multiple Precision Fixed-Point Filters Support

The Filter Design Toolbox 2.2 now supports word lengths greater than 64 bits. Earlier versions limited your fixed-point word length to 53 bits or fewer and 64 bits or fewer for floating-point word length.

Now you can use word lengths up to the memory capacity of your machine.

In addition, fraction lengths can now be greater than the associated word length, and can be less than zero.

All MATLAB Platforms Supported

With this release, the Filter Design Toolbox runs on all MATLAB platforms.

Four Filter Design Functions Added

This version of the toolbox adds four new filter design functions:

- `firceqrip`—design constrained equiripple FIR filters
- `iirnotch`—design second-order IIR notch digital filters
- `iircomb`—design IIR comb notching or peaking digital filters
- `iirpeak`—design second-order IIR peaking (also called resonator) digital filters

Enhanced Filter Design Functions

The following design functions have been upgraded to use the Leja algorithm to determine the minimum phase polynomial and from that the filter coefficients:

- `firminphase`—design higher-order minimum phase filters (n=100 or more), with better minimum phase characteristics. Also, added a new syntax option

```
firminphase(b, nz)
```

so you can specify the number of zeros that lie on the unit circle. Providing your estimate of the number of zeros can significantly improve the computation.

- `gremez`—design higher-order minimum phase filters (n=100 or more) with the `'minphase'` input argument. No changes to the syntax.

Five New Filter-Related Functions

Filter Design Toolbox 2.2 adds the ability to read and write Xilinx CORE Generator files, and the ability to realize models of quantized filters in Simulink™ if you own the DSP Blockset and Fixed-Point Blockset.

These three functions expand your filter options and capabilities:

- `coeread`—Read a XILINX CORE Generator™ coefficient (`.coe`) file
- `coewrite`—Write a XILINX CORE Generator™ coefficient (`.coe`) file

  FDATool has a new option for exporting quantized filters as `.coe` files when you own the Filter Design Toolbox. From the Targets menu option on the toolbar, you select **Export to XILINX Coefficient (.COE) file** to export your existing quantized filter coefficients to a .coe file. Your current filter must be a quantized direct-form FIR filter with one section; you cannot export nonquantized filters as `.coe` files, nor multiple-section filters.
- `realizemdl`—Create a Simulink model of your quantized filter using blocks from the Fixed-Point Blockset, or DSP Blockset if you do not own Fixed-Point Blockset.

Two other functions provide signal interpolation and decimation capability:

- `cicdecimate`—use a cascaded integrator-comb (CIC) decimation filter to decrease the sampling rate for a signal
- `cicinterpolate`—use a cascaded integrator-comb (CIC) interpolation filter to increase the sampling rate for a signal

New Frequency Transformation Functions Added

The Filter Design Toolbox 2.2 includes new filter transformation functions. Every transformation maintains the overall ripple characteristics of your prototype or original filter while transforming to your target filter specification.

All of the new transformation functions return the coefficients of your target filter and the coefficients of the allpass mapping filter used to transform your prototype filter to the new filter. In the following lists, transformation functions appear grouped by

- Allpass transformation filters
- Format of the prototype filter specification you supply
  - IIR transfer function
  - Zero-pole-gain

Allpass Filter Transformations

Use an allpass filter to map a prototype filter to a new frequency specification:

- `allpasslp2lp`—design an allpass filter to transform a lowpass filter to a lowpass filter
- `allpasslp2hp`—design an allpass filter to transform a lowpass filter to a highpass filter
- `allpasslp2bp`—design an allpass filter to transform a lowpass filter to a bandpass filter
- `allpasslp2bs`—design an allpass filter to transform a lowpass filter to a bandstop filter
- `allpassshift`—design an allpass filter to perform a real frequency shift transformation
- `allpaslp2mb`—design an allpass filter to transform a lowpass filter to an M-band frequency filter
- `allpasslp2xn`—design an allpass filter to transform a lowpass filter to N-point frequency specification
- `allpasslp2bpc`—design an allpass filter for lowpass to complex bandpass frequency transformation

- `allpasslp2bsc`—design an allpass filter for lowpass to complex bandstop frequency transformation
- `allpassshiftc`—design an allpass filter for complex shift frequency transformation
- `allpasslp2mbc`—design an allpass filter for lowpass to complex M-band frequency transformation
- `allpasslp2xc`—design an allpass filter for lowpass to complex N-point frequency transformation
- `allpassbpc2bpc`—design an allpass filter for complex bandpass frequency transformation
- `allpassrateup`—design an allpass filter for integer upsampling frequency transformation

IIR Filter Transformations

Use the allpass filter described in the allpass filter transformations to transform a prototype IIR filter to a new frequency response specification:

- `iirlp2mb`—transform an IIR filter from lowpass to M-band frequency response
- `iirlp2cn`—transform an IIR filter from lowpass to N-point frequency response
- `iirlp2bpc`—transform an IIR filter from lowpass to complex bandpass frequency response
- `iirlp2bsc`—transform an IIR filter from lowpass to complex bandstop frequency response
- `iirshiftc`—perform an IIR complex shift frequency transformation
- `iirlp2mbc`—transform an IIR filter from lowpass to complex M-band frequency response
- `iirlp2xc`—transform an IIR filter from lowpass to complex N-point frequency response
- `iirbpc2bpc`—transform an IIR filter from complex bandpass to complex bandpass frequency response
- `iirrateup`—perform an IIR interger upsample frequency transformation
- `iirftransf`—apply an IIR mapping filter to a prototype filter

Zero-Pole-Gain Filter Transformations

Use the allpass filter described in the allpass filter transformations to transform a prototype filter in zero-pole-gain format to a new frequency response specification:

- `zpklp2lp`—transform a filter specified by its zero-pole-gain form from lowpass to lowpass frequency response
- `zpklp2hp`—transform a filter specified by its zero-pole-gain form from lowpass to highpass frequency response
- `zpklp2bp`—transform a filter specified by its zero-pole-gain form from lowpass to bandpass frequency response
- `zpklp2bs`—transform a filter specified by its zero-pole-gain form from lowpass to bandstop frequency response
- `zpkshift`—use a filter in zero-pole-gain format to perform a real shift frequency transformation
- `zpklp2mb`—transform a filter specified by its zero-pole-gain form from lowpass to M-band frequency response
- `zpklp2xn`—transform a filter specified by its zero-pole-gain form from lowpass to N-point frequency response
- `zpklp2bpc`—transform a filter specified by its zero-pole-gain form from lowpass to complex bandpass frequency response
- `zpklp2bsc`—transform a filter specified by its zero-pole-gain form from lowpass to complex bandstop frequency response
- `zpkshiftc`—use a filter in zero-pole-gain format to perform a complex shift frequency transformation
- `zpklp2mbc`—transform a filter specified by its zero-pole-gain form from lowpass to complex M-band frequency response
- `zpklp2xc`—transform a filter specified by its zero-pole-gain form from lowpass to complex N-point frequency response
- `zpkbpc2bpc`—transform a filter specified by its zero-pole-gain form from complex bandpass to complex bandpass frequency response
- `zpkrateup`—use a filter in zero-pole-gain format to perform a complex bandpass frequency transformation
- `zpkftransf`—use a filter specified in zero-pole-gain format to transform an IIR lowpass filter to a new IIR lowpass filter

You use these from the MATLAB command line or from the new **Transformations** panel provided in FDATool. To read about using these new transformations, refer to “Digital Frequency Transformations” in the online help.

New Transformations Pane Added to FDATool

A new **Transformations** pane in FDATool provides access to the new transformation functions. All filter transformations work from the command line as well. In addition, Version 2.2 modifies the existing filter transformations in FDATool—they use the same input and output arguments but with modified transformation techniques.

Replaces the **Transformations** menu option in earlier toolbox versions and expands the capability to transform filters.

nlm Filter Analysis Method in FDATool

The new release of Filter Design Toolbox adds the noise loading method (`nlm`) to FDATool. While this analytical method was in the toolbox as a command line function, `nlm` is now available in the GUI.

To run the noise loading method in FDATool:

1. Design or import a filter into FDATool.
2. Quantize the filter, remembering to set the quantization parameters as required.
3. To use the noise loading method to estimate the frequency response of your quantized filter, select **Analysis -> Noise Loading Method** on the FDATool menu bar.

   FDATool runs the noise loading method Monte Carlo trials on the filter and displays the result in the Analysis area. For more information about nlm and using FDATool to perform the analysis, refer to “Analyzing Filters with the Noise Loading Method” in the Filter Design Toolbox online help.

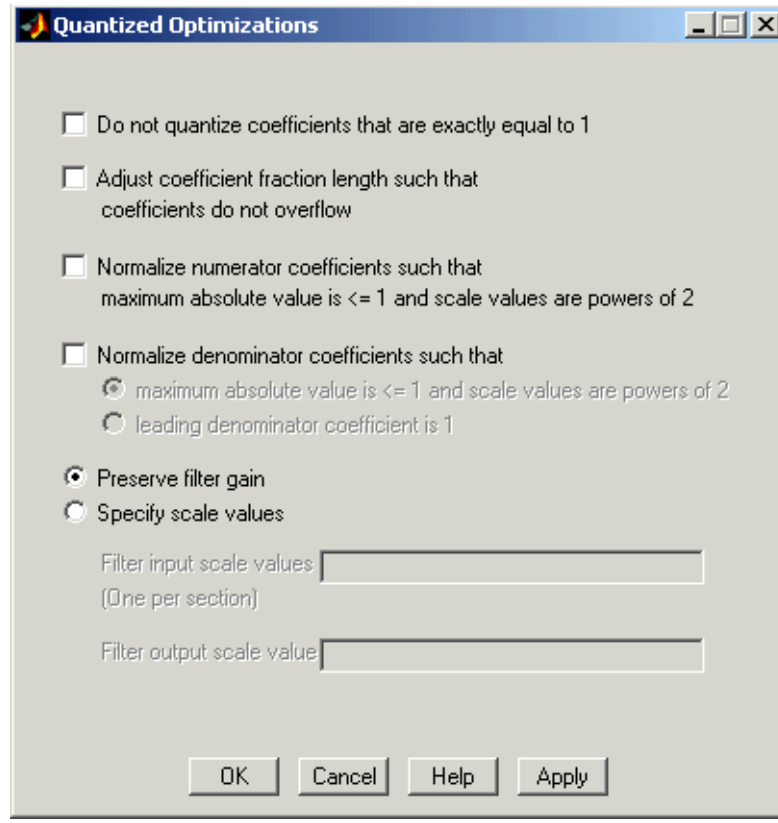
New Functions for Calculating Quantizer Noise Statistics

Three new functions for determining the noise statistics of a quantizer now appear in the toolbox:

- `errmean`—return the mean of the quantization error resulting from quantizing a signal
- `errpdf`—calculate the probability density function (pdf) of the quantization error
- `errvar`—return the variance of the quantization error resulting from quantizing a signal

Enhanced Quantizer Optimization

For this release, a new **Optimization...** option replaces the scaling functions on the quantization panel. Optimization enables you to specify a larger range of optimization options for your quantized filters. When you click **Optimization...** on the quantizer panel, you see the following **Quantized Optimizations** dialog, shown with the default settings:



For more information about using the new optimization features, refer to "Optimizing the Quantization Process For Your Filter" in the online help.

Enable and Disable Quantizers in a Filter

In the revised **Quantized Filter** pane in FDATool, you find a new option—six check boxes that enable or disable the quantizers in a filter. By clearing or selecting a check box, you determine whether to apply the associated quantizer to the filter during the quantization process. For example, you can elect not to apply the Input quantizer by clearing the check box **Convert input to**. During quantization, the inputs will be passed on without modification or changes to their values.

Quantizer Status Functions Added

To help you determine various characteristics of quantizers, the toolbox includes three new functions:

- `isfixed`—test and return whether a quantizer is fixed point
- `isfloat`—test and return whether a quantizer is floating point
- `isnone`—determine and return whether a quantizer has quantization mode equal to none

Enhanced Quantized Filter Structures

To improve the scaling performance and reliability of quantized filters in the toolbox, the quantized filter structures no longer include the input and output scale values s(1) and s(2) in the default structures. This eliminates two quantization error sources when you quantize your filter without input and output scaling.

Also, the df1, df1t, df2, and df2t structures do not include the leading denominator coefficient, a(1), when the coefficient is equal to one. When $a(1) \neq 1$, the structures include the coefficients. Thus, when you quantize a filter with $a(1) = 1$, the quantization skips the multiplication operation and does not quantize a(1).

To see both these changes, and for more information, refer to “Quantized Filter Property Reference” in your Filter Design Toolbox documentation or the Help browser. For an example, the next figures show the original df2t structure and the newer version.

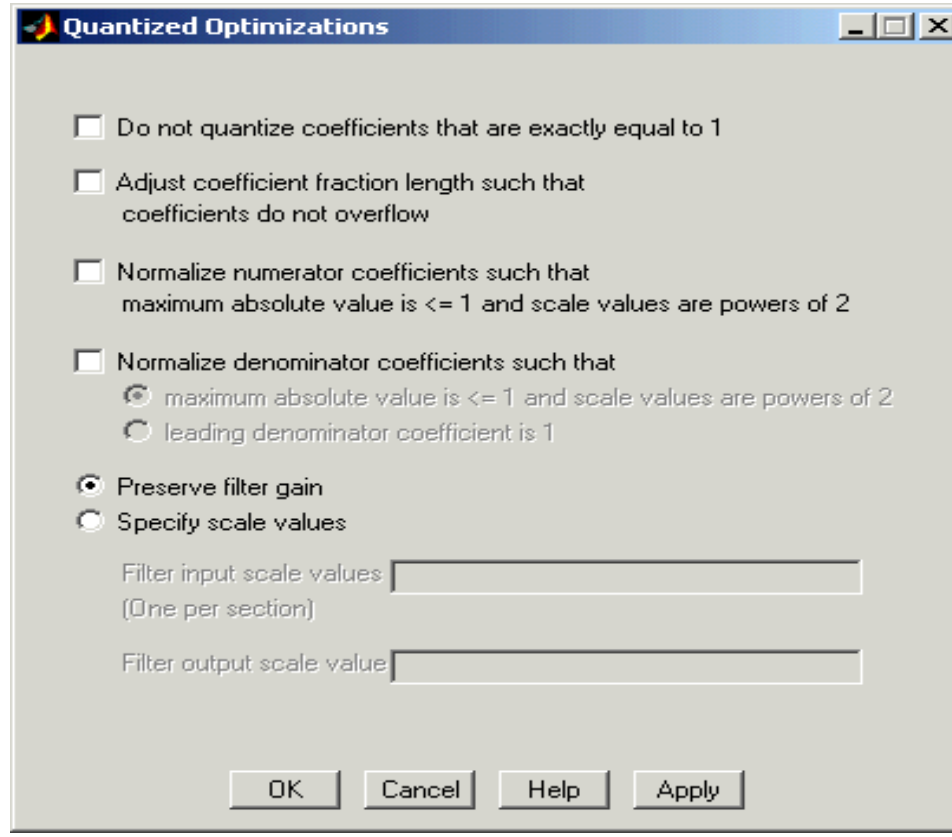


Figure 2-1: Original dft2 Structure Including s(1) and s(2)

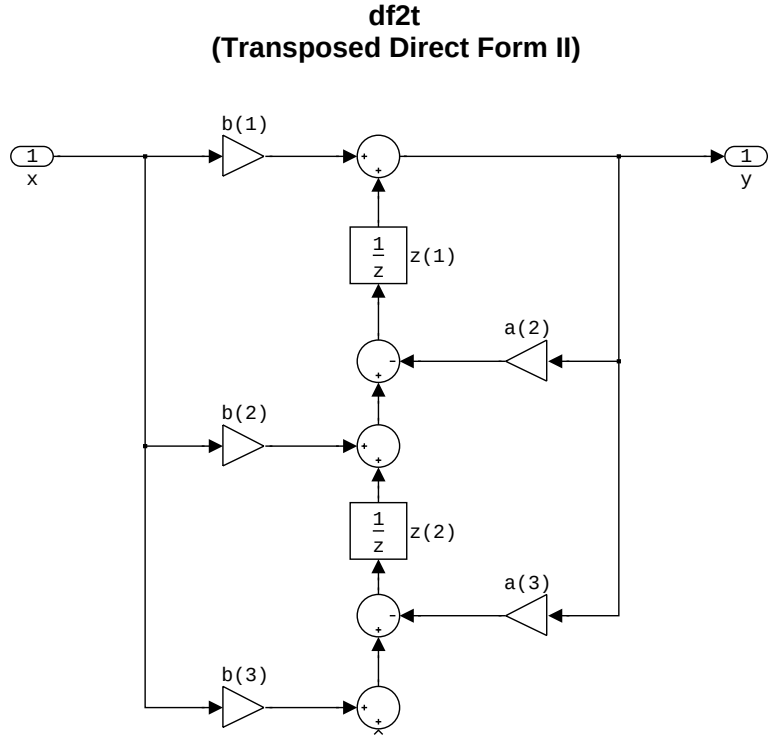


Figure 2-2: New df2t Structure Without Scale Values and with Coefficient a(1) = 1

New Graphic Showing the Quantizers in a Filter

To help you understand where and how the quantizers in a filter structure work, your toolbox documentation provides a figure detailing the quantizers in a df2 quantized filter structure. Refer to “Filter Structure with Quantizers in Place” in your Filter Design Toolbox documentation.

New and Enhanced Filter Demonstration Programs

Version 2.2 adds to and enhances the existing set of demos for the toolbox. We’ve added:

- `firceqripdemo`—demonstrates the filter design function `firceqrip` for designing constrained equiripple FIR filters
- `firtransdemo`—demonstrates frequency transformations for linear-phase FIR filters
- `freqtransdemo`—demonstrates frequency transformations for IIR filters
- `noisepowerdemo`—demonstrates the noise power spectrum for filters
- `numbercircledemo`—demonstrates and defines fixed-point numbers
- `qerrordemo`—demonstrates the new quantizer statistics functions

And we substantially enhanced these existing demos:

- cademo
- firlpnormdemo
- firnyquistdemo
- gremezdemo
- iirgrpdelaydemo
- iirlpnormdemo
- iirlpnormcdemo

Major Bug Fixes

The Filter Design Toolbox 2.2 includes several bug fixes made since Version 2.1. This section describes the particularly important Version 2.2 bug fixes.

If you are viewing these Release Notes in PDF form, please refer to the HTML form of the Release Notes, using either the Help browser or the MathWorks Web site and use the link provided.

Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Filter Design Toolbox 2.1 to Version 2.2.

Moved zerophase Function

The filter analysis function `zerophase`, which was included in Filter Design Toolbox, has been relocated to the Signal Processing Toolbox. The syntax has changed to take a numerator and a denominator as its first two input arguments. Recall that you must have the Signal Processing Toolbox to use the Filter Design Toolbox, so `zerophase` remains available to all Filter Design Toolbox customers.

Note The following upgrade information is the same as appeared in the Release Notes for Release 12.

Changes to Quantizer set Function

With this release, and moving forward, you can not use property values alone to set property values when you use `set`. You must supply the property name and property value in the call. In your existing code, be sure that calls to `set` include both property names and property values. If you do not include the property name with the property value, MATLAB returns a warning similar to

```
??? There is no 'propertyvalue' property in the 'quantizer' class.
```

indicating that you should add the property name to define the required property in the syntax. For example:

```
q=quantizer

q =

            Mode = fixed
       RoundMode = floor
    OverflowMode = saturate
          Format = [16  15]
```

```
set(q,'wrap',[16 15])
??? There is no 'wrap' property in the 'quantizer' class.
set(q,'overflowmode','wrap','format',[10 8])
get(q)
              mode: 'fixed'
         roundmode: 'floor'
      overflowmode: 'wrap'
            format: [10 8]
```

Obsolete Functions in Version 2.0

Filter Design Toolbox 2.0 makes obsolete the following functions that were part of Quantized Filter Design Toolbox.

Obsolete Function	Suggested Replacement
`propinfo`	Use `help constructor/propertyname` to get help about a function. Or use the Help browser.
`qfiltlog`	Use `qreport` to get information about quantized filters, quantized FFTs, and quantizers.
`qhelp`	Use `help constructor/propertyname` to get help about a function. Or use the Help browser.

Known Software and Documentation Problems

For a list of bugs reported in the previous release that remain open, see "Known Software and Documentation Problems" on page 3-6.

3

Filter Design Toolbox 2.1 Release Notes

New Features

This section introduces the new features and enhancements added in the Filter Design Toolbox 2.1 since Filter Design Toolbox 2.0 (Release 12).

For information about Filter Design Toolbox features that are incorporated from recent releases, see “New Features” on page 4-2 in the Filter Design Toolbox 2.0 Release Notes.

New Adaptive Filtering Functions

The Filter Design Toolbox 2.1 includes these new filter design functions:

- `adaptkalman` — Use a Kalman filtering algorithm in an adaptive filter role
- `adaptlms` – Use an LMS-based adaptive algorithm in an adaptive filter role
- `adaptnlms` – Use a normalized LMS-based adaptive algorithm in an adaptive filter role
- `adaptrls` – Use an RLS-based adaptive algorithm in an adaptive filter role
- `adaptsd` – Use the sign-data variant of the LMS-based adaptive algorithm in an adaptive filter role
- `adaptse` – Use the sign-error variant of the LMS-based adaptive algorithm in an adaptive filter role
- `adaptss` — Use the sign-sign variant of the LMS-based adaptive algorithm in an adaptive filter role

To support you when you use these new functions, the Toolbox includes corresponding functions for initializing the associated adaptive filter algorithm.

- `initkalman` — Initialize the input argument `s` for the Kalman-based algorithm adaptive filter function
- `initlms` — Initialize the input argument `s` for the LMS-based adaptive filter function
- `initnlms` — Initialize the input argument `s` for the normalized LMS-based adaptive filter function
- `initrls` — Initialize the input argument `s` for the RLS-based adaptive filter function

- `initsd` — Initialize the input argument `s` for the sign-data variant of the LMS-based adaptive filter function
- `initse` — Prepare the input arguments for the sign-error variant of the LMS-based adaptive filter function
- `initss` — Prepare the input arguments for the sign-sign variant of the LMS-based adaptive filter function

New FIR Filter Design Functions

In addition to the new adaptive filtering capabilities, this version of the toolbox has four new filter design functions.

- `firhalfband` — Design Lth-band filters where L=2. About half of the filter coefficients are zero so the filters are very efficient to calculate
- `firminphase` — Calculate the minimum-phase FIR spectral factor of a linear-phase FIR filter
- `firnyquist` — Design lowpass filters with certain specified coefficients in the transfer functions set to zero by design
- `ifir` — Design an interpolated FIR filter

FDATool includes `Halfband` and `Nyquist` entries under **Filter Type** so you can design and analyze these kinds of filters within FDATool.

New Filter Transformation Functions

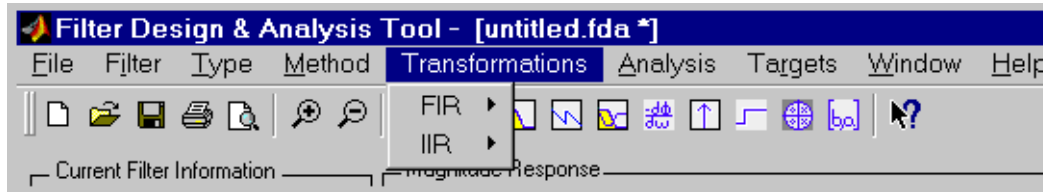
The Filter Design Toolbox 2.1 includes several new filter transformation functions. Each transformation maintains the ripple characteristics and passband/stopband shapes of the source filter while transforming to the new specification.

- `firlp2lp` — Transforms lowpass FIR filters to lowpass FIR filters with different passband width or cutoff specifications
- `firlp2hp` — Transforms lowpass FIR filters to highpass FIR filters
- `iirlp2bp` — Transforms lowpass IIR filters to bandpass IIR filters
- `iirlp2bs` — Transforms lowpass IIR filters to bandstop IIR filters
- `iirlp2hp` — Transforms lowpass IIR filters to highpass IIR filters
- `iirlp2lp` — Transforms lowpass IIR filters to lowpass IIR filters

The new **Transformations** option (refer to the next entry in this section) in FDATool provides access to these transformations from the GUI.

Transformations Option in FDATool

When you use Filter Design and Analysis Tool (FDATool) with Filter Design Toolbox 2.1, the Toolbox adds a new menu to the FDATool menu bar. Named **Transformations**, the new selection provides direct access to the new FIR and IIR filter transformation functions added to the Toolbox in this release (refer to "New Filter Transformation Functions"). **Transformations** appears as shown in this figure only when you have installed the Filter Design Toolbox.



With this new option, FIR and IIR filters whose characteristics are particularly valuable become the sources for other FIR and IIR filters that share the same desirable features in a different passband configuration. Transforming a filter saves you from repeating the filter design process. For more information, refer to Transforming Filters in your *Filter Design Toolbox User's Guide*.

In the figure, the **Targets** option is available only when you have the Developer's Kit for Texas Instruments DSP installed.

New Analysis Method

Filter Design Toolbox 2.1 includes one new analysis method — `zerophase` — that returns the zero-phase response of a linear-phase FIR filter.

`[hr,w] = zerophase(b)`, and the other available syntax options for `zerophase`, returns the zero-phase response `hr` and the frequency vector `w` (in rad/sample) at which `hr` is computed, given a linear-phase FIR filter `b`. The zero-phase response is evaluated at 512 equally spaced points on the upper half of the unit circle.

Note that the zero-phase response is always real, but it is not equivalent to the magnitude response. In particular, the zero-phase response can be negative; the magnitude response is always positive.

New Context-Sensitive Help for Quantization

FDATool Quantization mode now includes context sensitive or What's This? help. For each option on the **Set Quantization Parameters** page, right-click to use the What's This? option to open a brief text description of the option.

Known Software and Documentation Problems

This section updates the Filter Design Toolbox 2.1 documentation, reflecting known Filter Design Toolbox 2.1 software or documentation problems.

Switching Between Design and Quantization Modes in FDATool

After you scale a quantized filter, or convert a quantized filter to second-order sections, do not switch between quantized mode and filter design mode in FDATool. Selecting and clearing the **Turn quantization on** option (to shift between the quantization and design modes) after you have scaled or converted a quantized filter can corrupt the coefficients for your reference filter and your quantized filter.

To recover your reference filter design and quantized filter design if they have been corrupted, either:

- Import the filter from your workspace again
- Use FDATool to repeat the filter design

Help for Filter Transformations in FDATool

Option **Transformations** in FDATool does not have What's This? help available. Use `helpwin` *`functionname`* for information about the transformation functions or use the Help browser to search the online documentation.

4

Filter Design Toolbox 2.0 Release Notes

New Features

Incorporating functionality from the Quantized Filtering Toolbox, Filter Design Toolbox 2.0 is a new product for digital filter design, implementation, and analysis using fixed-point or arbitrary-precision floating-point arithmetic. This toolbox includes advanced FIR and IIR filter design methods and a new quantization tool for designing and analyzing quantized FIR and IIR filters. This section introduces the features and enhancements added in the Filter Design Toolbox 2.0 since the Quantized Filtering Toolbox 1.0 was released in Release 11.1.

Note The Quantized Filtering Toolbox 1.1 has been integrated into the Filter Design Toolbox 2.0. The information below summarizes the changes to the Quantized Filtering Toolbox since Version 1.0 of that product. These changes are incorporated into the new Filter Design Toolbox 2.0.

You must have the Signal Processing Toolbox installed to use Filter Design Toolbox.

New Quantization Tool for Advanced FIR and IIR Filter Design

Signal Processing Toolbox includes an interactive filter design tool called Filter Design and Analysis Tool (FDATool). You use this visual interface to design, convert, quantize, import, and export FIR and IIR filters. FDATool integrates filter design and analysis functions in a single tool, letting you do all your filter design tasks from within the tool instead of using the MATLAB command line. Filter Design Toolbox adds quantization analysis to FDATool.

When you install Filter Design Toolbox, FDATool operates in three modes:

- Filter Design for designing double-precision FIR and IIR filters.
- Import for importing existing filters from your MATLAB workspace or by entering an expression.
- Quantization for quantizing filters in FDATool. You can quantize any filter in FDATool, whether you designed or imported the filter.

To enable you to convert filters from one structure to another, for example, from direct form to coupled allpass, FDATool offers smart conversion tools. When you convert a filter, FDATool offers you a selection of conversion structures that depends on the current filter type.

To launch FDATool, enter `fdatool` at the MATLAB command prompt or use the Launch Pad. For more information on the new tool, refer to "Filter Design and Analysis Tool" in the *Signal Processing Toolbox User's Guide* or use Help to review the online documentation.

New Objects

The Filter Design Toolbox 2.0 includes these new objects:

- Quantizer object

  Use the quantizer to quantize data using fixed-point or custom floating-point arithmetic. Use the `quantizer` function to construct a quantizer, and use the `quantize` function to quantize the data according to your quantizer specifications. There are also many functions that calculate the characteristics associated with a given quantizer. For example, `eps` determines the precision of the quantizer, and `realmax` determines the maximum number the quantizer can quantize without overflow.

- Quantized FFT object

  Use the new quantized FFT object to create and simulate fixed-point and custom floating-point FFTs. Construct a quantized FFT to specify the quantizing characteristics of the FFT method that you want to apply to data. Use the function `qfft` to construct a quantized FFT, and the function `fft` to apply an FFT to data as specified by your quantized FFT.

New Filter Design Functions

The Filter Design Toolbox 2.0 includes these new filter design functions:

- `firlpnorm` – Design a least P-norm FIR filter
- `gremez` – Use generalized Remez techniques to design FIR filters
- `iirgrpdelay` – IIR filter design with prescribed group delay value
- `iirlpnorm` – Design a least P-norm IIR filter
- `iirlpnormc` – Design a constrained least P-norm IIR filter

New Filter Conversion Functions

The Filter Design Toolbox 2.0 includes these new filter conversion functions:

- `ca2tf` – Convert coupled allpass transfer function forms to transfer function forms
- `cl2tf` – Convert coupled allpass lattice forms to transfer function forms
- `iirpowcomp` – Compute a power-complementary IIR filter from a given IIR filter
- `sos`: Convert the filter topology of a quantized filter to second-order sections with scaling
- `tf2ca` – Convert transfer function forms to coupled allpass transfer function forms
- `tf2cl` – Convert transfer function forms to coupled allpass lattice forms
- Quantized filter coefficient conversion

  You can now convert quantized filter coefficients to a hex or binary format, using these functions:

  - `num2hex` for hexadecimal conversion
  - `num2bin` for binary conversion

  These functions are overloaded for quantizers and quantized FFTs.

New Filter Structures

Filter Design Toolbox 2.0 includes five new filter structures:

- Direct form FIR transposed
- Direct form antisymmetric FIR
- Lattice coupled-allpass
- Second-order sections
- Lattice coupled-allpass power-complementary

To learn more about these structures, refer to “Filter Structures” in the *Filter Design Toolbox User’s Guide*.

New Analysis Methods

Filter Design Toolbox 2.0 includes two new analysis methods:

- Limit Cycle – The new function `limitcycle` uses Monte Carlo techniques to detect limit cycles in quantized filters.
- Noise Loading Method – Use function `nlm` to use the noise loading method to calculate the frequency response of a quantized filter. Compare the response to the results from `freqz`, which calculates the theoretical frequency response. To generate noise signals that contain complete frequency content across the spectrum, `nlm` uses a series of Monte Carlo trials.

New Functions and Enhancements

Filter Design Toolbox 2.0 includes the following new functions and enhancements:

- Added `copyobj` function for copying quantized filters, quantized FFTs, and quantizers. Copies are independent of the original items, but have the same property values as the original.
- Changed the `CoefficientFormat` property to default to `Quantizer` rather than `Unitquantizer` for quantized filters, quantized FFTs, and quantizers.
- Added `NOperations`, a new data-related read-only property for quantizers that counts the number of data points quantized by a quantizer.
- Updated the quantized filter data display to indicate over- and underflow conditions in filter coefficients, as well as adding the `NOperations` read-only values to the displayed information.
- Changed the default rounding method for the `CoefficientFormat` property for quantized filters and FFTs to `'round'` rather than `'floor'`.
- Included new quantized filtering demos. Use the `demo` function to access the new demos.
- Modified the data format properties.
- You can now set the following properties individually for each data path in a quantizer, quantized filter, or quantized FFT:
  - `Mode`
  - `RoundMode`
  - `OverflowMode`
  - `OptimizeUnityGains`

Upgrading from an Earlier Release

Obsolete Functions in Version 2.0

Filter Design Toolbox 2.0 makes obsolete the following functions that were part of Quantized Filter Design Toolbox:.

Obsolete Function	Suggested Replacement
`propinfo`	Use `help constructor/propertyname` to get help about a function. Or use the Help browser.
`qfiltlog`	Use `qreport` to get information about quantized filters, quantized FFTs, and quantizers.
`qhelp`	Use `help constructor/propertyname` to get help about a function. Or use the Help browser.

Known Software and Documentation Problems

This section updates the Filter Design Toolbox 2.0 documentation, reflecting known Filter Design Toolbox 2.0 software or documentation problems.

Filter Design and Analysis Tool

The quantization dialog in FDATool does not provide context-sensitive help in this release. To get help on a control in the dialog, use the Help browser. Select an option, such as **FDATool Help**, from the **Help** menu in FDATool. Or type `helpdesk` at the MATLAB prompt to open the online help system.

Switching Between Design and Quantization Modes in FDATool

After you scale a quantized filter, or convert a quantized filter to second-order sections, do not switch between quantized mode and filter design mode in FDATool. Checking and clearing the **Turn quantization on** option (to shift between the quantization and design modes) after you have scaled or converted a quantized filter can corrupt the coefficients for your reference filter and your quantized filter.

To recover your reference filter design and quantized filter design if they have been corrupted, either:

- Import the filter from your workspace again
- Use FDATool to repeat the filter design