Filter Design Toolbox™ Release Notes

How to Contact 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.

Filter Design Toolbox™ Release Notes

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

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

Contents

Summary by Version

This table provides quick access to what's new in each version. For clarification, see "Using Release Notes" on page 2 below.

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Latest Version V4.7.1 (R2010b)	No	No	Bug Reports	Printable Release Notes: PDF Current product documentation
V4.7 (R2010a)	Yes Details	Yes Summary	Bug Reports	No
V4.6 (R2009b)	Yes Details	No	Bug Reports	No
V 4.5 (R2009a)	Yes Details	Yes Summary	Bug Reports	No
V4.4 (R2008b)	Yes Details	No	Bug Reports	No
V4.3 (R2008a)	Yes Details	Yes Summary	Bug Reports	No
V4.2 (R2007b)	Yes Details	Yes Summary	Bug Reports	No
V4.1 (R2007a)	Yes Details	No	Bug Reports	No
V4.0 (R2006b)	Yes Details	No	Bug Reports	No
V3.4 (R2006a)	Yes Details	Yes Summary	Bug Reports	No

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
V3.3 (R14SP3)	Yes Details	Yes Summary	Bug Reports	No
V3.2 (R14SP2)	Yes Details	Yes Summary	Bug Reports	No

Using Release Notes

Use release notes when upgrading to a newer version to learn about:

- New features
- Changes
- Potential impact on your existing files and practices

Review the release notes for other MathWorks® products required for this product (for example, MATLAB® or Simulink®). Determine if enhancements, bugs, or compatibility considerations in other products impact you.

If you are upgrading from a software version other than the most recent one, review the current release notes and all interim versions. For example, when you upgrade from V1.0 to V1.2, review the release notes for V1.1 and V1.2.

What Is in the Release Notes

New Features and Changes

- New functionality
- Changes to existing functionality

Version Compatibility Considerations

When a new feature or change introduces a reported incompatibility between versions, the **Compatibility Considerations** subsection explains the impact.

Compatibility issues reported after the product release appear under Bug Reports at the MathWorks Web site. Bug fixes can sometimes result in incompatibilities, so review the fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

MathWorks offers a user-searchable Bug Reports database so you can view Bug Reports. The development team updates this database at release time and as more information becomes available. Bug Reports include provisions for any known workarounds or file replacements. Information is available for bugs existing in or fixed in Release 14SP2 or later. Information is not available for all bugs in earlier releases.

Access Bug Reports using your MathWorks Account.

Version 4.7.1 (R2010b) Filter Design Toolbox

This table summarizes what is new and changed in Version 4.7.1 (R2010b)

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
No	No	Bug Reports	Printable Release Notes: PDF Current product documentation

Version 4.7 (R2010a) Filter Design Toolbox

This table summarizes what is new and changed in Version 4.7 (R2010a)

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes — Details labeled as **Compatibility Considerations** below. See also Summary.	Bug Reports	Printable Release Notes: PDF Current product documentation

- "Audio Weighting Filters" on page 5
- "Functions, Objects, Object Methods, and Object Properties Being Removed" on page 6

New features and changes with compatibility impact introduced in this version:

Audio Weighting Filters

In R2010a, you can design several types of audio weighting filters. Audio weighting filters model the frequency and level-dependent perceptual weights of the human auditory system. The filter specification object `fdesign.audioweighting` provides support for A, C, C-message, ITU-T 0.41, and ITU-R 468-4 weighting filters.

Functions, Objects, Object Methods, and Object Properties Being Removed

Name	What Happens When you Use the Function, Object, Object Method, or Object Property	Use Instead	Compatibility Considerations
`adaptlms` (Object)	Errors	`adaptfilt.lms`	Replace all existing instances of `adaptlms` with `adaptfilt.lms`.
`adaptnlms` (Object)	Errors	`adaptfilt.nlms`	Replace all existing instances of `adaptnlms` with `adaptfilt.nlms`.
`adaptrls` (Object)	Errors	`adaptfilt.rls`	Replace all existing instances of `adaptrls` with `adaptfilt.rls`.
`adaptsd` (Object)	Errors	`adaptfilt.sd`	Replace all existing instances of `adaptsd` with `adaptfilt.sd`.
`adaptse` (Object)	Errors	`adaptfilt.se`	Replace all existing instances of `adaptse` with `adaptfilt.se`.
`adaptss` (Object)	Errors	`adaptfilt.ss`	Replace all existing instances of `adaptss` with `adaptfilt.ss`.
`fdesign.decim` (Object)	Errors	`fdesign.decimator`	Replace all existing instances of `fdesign.decim` with `fdesign.decimator`.

Name	What Happens When you Use the Function, Object, Object Method, or Object Property	Use Instead	Compatibility Considerations
`fdesign.interp` (Object)	Errors	`fdesign.interpolator`	Replace all existing instances of `fdesign.interp` with `fdesign.interpolator`.
`fdesign.src` (Object)	Errors	`fdesign.rsrc`	Replace all existing instances of `fdesign.src` with `fdesign.rsrc`.
`farrow.fd` (Object)	Errors	`dfilt.farrowfd`	Replace all existing instances of `farrow.fd` with `dfilt.farrowfd`.
`farrow.linearfd` (Object)	Errors	`dfilt.farrowfd`	Replace all existing instances of `farrow.linearfd` with `dfilt.farrowfd`.
`filtmsb` (Method)	Errors	`FilterInternals` property of `mfilt.cicdecim` and `mfilt.cicinterp` objects	Replace all existing instances of `filtmsb` by setting the `FilterInternals` property of `mfilt.cicdecim` and `mfilt.cicinterp` objects to `'FullPrecision'`.
`initlms` (Object)	Errors	`adaptfilt.lms`	Replace all existing instances of `initlms` with `adaptfilt.lms`.
`initnlms` (Object)	Errors	`adaptfilt.nlms`	Replace all existing instances of `initnlms` with `adaptfilt.nlms`.

Name	What Happens When you Use the Function, Object, Object Method, or Object Property	Use Instead	Compatibility Considerations
`initrls` (Object)	Errors	`adaptfilt.rls`	Replace all existing instances of `initrls` with `adaptfilt.rls`.
`initsd` (Object)	Errors	`adaptfilt.sd`	Replace all existing instances of `initsd` with `adaptfilt.sd`.
`initse` (Object)	Errors	`adaptfilt.se`	Replace all existing instances of `initse` with `adaptfilt.se`.
`initss` (Object)	Errors	`adaptfilt.ss`	Replace all existing instances of `initss` with `adaptfilt.ss`.
`legacyfixptfir`	Warns		Replace all pre-R14SP2 fixed-point FIR filters with post-R14SP2 fixed-point FIR filters.
`SectionWordLengthMode` (Property of `mfilt.cicdecim` and `mfilt.cicinterp` objects)	Errors	`FilterInternals` property of `mfilt.cicdecim` and `mfilt.cicinterp` objects	Replace all existing instances of `SectionWordLengthMode` by setting the `FilterInternals` property of `mfilt.cicdecim` and `mfilt.cicinterp` objects to `'MinWordLengths'`, or `'SpecifyWordLengths'`.

Name	What Happens When you Use the Function, Object, Object Method, or Object Property	Use Instead	Compatibility Considerations
`StageInputAutoScale` (Property of fixed-point `dfilt.df1tsos`, `dfilt.df2sos`, and `dfilt.df2tsos` objects)	Warns	`SectionInputAutoScale`	Replace all existing instances of `StageInputAutoScale` with `SectionInputAutoScale`.
`StageOutputAutoScale` (Property of fixed-point `dfilt.df1tsos`, `dfilt.df2sos`, and `dfilt.df2tsos` objects)	Warns	`SectionOutputAutoScale`	Replace all existing instances of `StageOutputAutoScale` with `SectionOutputAutoScale`.
`StageInputWordLength` (Property of fixed-point `dfilt.df1tsos`, `dfilt.df2sos`, and `dfilt.df2tsos` objects)	Warns	`SectionInputWordLength`	Replace all existing instances of `StageInputWordLength` with `SectionInputWordLength`.
`StageOutputWordLength` (Property of fixed-point `dfilt.df1tsos`, `dfilt.df2sos`, and `dfilt.df2tsos` objects)	Warns	`SectionOutputWordLength`	Replace all existing instances of `StageOutputWordLength` with `SectionOutputWordLength`.

Version 4.6 (R2009b) Filter Design Toolbox

This table summarizes what is new and changed in Version 4.6 (R2009b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version:

Optimal Finite Wordlength FIR Filter Designs

Release R2009b introduces three new `dfilt` and `mfilt` methods for optimizing finite wordlength FIR filter designs. You can construct fixed-point filters that meet the floating-point specifications while minimizing or constraining the coefficient wordlength, or maximizing the stopband attenuation. For more information, see `constraincoeffwl`, `maximizestopband`, and `minimizecoeffwl`. Construction of fixed-point filters requires the Fixed-Point Toolbox™ software.

Quality and Shelving Factor Support Added for Parametric Equalizer Filters

This release introduces support for the specification of a quality factor or shelving slope in parametric equalizer filters. You can specify parametric equalizer filters using the quality factor or shelving slope parameter in both `fdesign.parameq` and `filterbuilder`. Access the parametric equalizer filter design pane in `filterbuilder` by entering:

```
filterbuilder('parameq')
```

at the MATLAB command prompt. Set `Order mode` to `specify` to enable the quality factor or shelving slope specifications.

Multirate Pulse-Shaping Filter Support Added for Decimator, Interpolator, and Rational Sample-Rate Converter Designs

R2009b introduces support for polyphase pulse-shaping filter designs using `fdesign.decimator`, `fdesign.interpolator`, `fdesign.rsrc`, and `filterbuilder`. The supported pulse-shaping filters include: Gaussian, raised cosine, and square root raised cosine filters. Access the design pane for multirate pulse shaping filters in `filterbuilder` by entering:

```
filterbuilder('pulseshaping')
```

at the MATLAB command prompt. Specify the type of multirate filter with `Filter Type`.

Zero-Phase Option Added for Equiripple Nyquist Filters

Filter Design Toolbox™ Version 4.6 software now supports zero-phase equiripple Nyquist filter designs. The zero-phase option is available in `fdesign.nyquist`, `fdesign.halfband`, and `filterbuilder` using the `'N,TW'` specification string. Setting `'ZeroPhase'` to `true` returns a nonnegative amplitude, or zero phase response. `'ZeroPhase'` defaults to `false`. The following example demonstrates how to design an equiripple Nyquist filter with a nonnegative zero phase response:

```
f=fdesign.nyquist(4,'N,TW');
```

```
d=design(f,'equiripple','ZeroPhase',true);
% Plot the zero-phase response
zerophase(d)
```

In `filterbuilder`, set `Order mode` to `Specify` and then set `Frequency constraints` to `Transition width`. These settings enable the nonnegative zero phase response option under `Design options`.

filterbuilder Support for Peaking and Notching IIR Comb Filters

In R2009b, `filterbuilder` supports peaking and notching IIR comb filters. Access the comb filter design pane by entering:

```
filterbuilder
```

at the MATLAB command prompt and choosing `comb` from filter response menu, or by entering:

```
filterbuilder('comb')
```

For help on using `filterbuilder`, see “Designing a Filter in the Filterbuilder GUI”.

Ability to Export Filter Coefficients Added to realizemdl

If you use Simulink, you can now use the new `MapCoeffstoPorts` property with `realizemdl` to map filter coefficients from `dfilt` and `mfilt` objects to constant blocks. The coefficients also appear in the MATLAB workspace providing tunability to the realized Simulink model. See `dfilt` and `mfilt` for a list of supported filter structures and any restrictions.

Version 4.5 (R2009a) Filter Design Toolbox

This table summarizes what is new and changed in Version 4.5 (R2009a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes Summary	Bug Reports	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are described here.

Fdesign Support for Comb Filters

`fdesign` now supports IIR peaking and notching comb filters. You can use comb filters to enhance (peaking) or eliminate (notching) energy in the input data at a series of harmonically-related frequencies.

To construct comb filters use the `fdesign.comb` function and specify the `Combtype` property as `'notch'` or `'peak'`.

Maximum Phase Option Added for Equiripple Designs

Equiripple filter designs now include a maximum phase option. You can construct maximum-phase equiripple filter using the `fdesign.lowpass`, `fdesign.bandpass`, and `fdesign.bandstop` filter designers.

The impulse response of a maximum-phase filter builds more slowly than any other filter of equal length and magnitude response. All the zeros of a maximum-phase filter lie on or outside the unit circle. Maximum-phase filters are useful in a number of applications including channel equalization. The following example demonstrates how to construct a maximum-phase equiripple filter using the lowpass filter designer:

```
d=fdesign.lowpass; %Construct a lowpass filter object using the defaults.
Hd=design(d,'equiripple','maxphase',true); %Implement the filter.
zplane(Hd) %Note all the zeros are on or outside the unit circle.
fvtool(Hd,'analysis','impulse') %Plot the impulse response.
```

Changes in SOS IIR Design Defaults

Changes have been made to the default specifications invoked by `fdesign` when constructing second-order sections (SOS) IIR filters.

Previous to the R2009a release, lowpass, highpass, bandpass, or bandstop SOS IIR filters used peak magnitude response scaling (`Linf`) by default.

In R2009a no scaling is applied by default to lowpass, highpass, bandpass, or bandstop SOS IIR filters.

Compatibility Considerations

Section-order sections IIR filters generated in R2009a will be scaled differently than in previous releases. Users can recreate their pre-R2009a default designs by setting the `'SOSScaleNorm'` property to `'Linf'`. The following example illustrates a lowpass Butterworth filter design in R2008b and its equivalent in R2009a.

Code in R2008b:

```
d=fdesign.lowpass; %lowpass filter designer
Hd=design(d,'butter'); %Butterworth filter with default (Linf) scaling
```

Equivalent code in R2009a:

```
d=fdesign.lowpass; %lowpass filter designer
Hd=design(d,'butter','sosscalenorm','linf');
```

Note You must consider scaling carefully when you design fixed-point filters to guard against overflow. See the "Floating-Point to Fixed-Point Conversion of IIR Filters" demo for overflow analyses, both with and without scaling.

Changes in FIR Equiripple Design Defaults

In R2009a, the `firpm` function is used by default to construct the following FIR equiripple filters:

```
fdesign.lowpass('Fp,Fst,Ap,Ast')
fdesign.lowpass('N,Fp,Fst')
fdesign.highpass('Fst,Fp,Ast,Ap')
fdesign.highpass('N,Fst,Fp')
fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2')
fdesign.bandpass('N,Fst1,Fp1,Fp2,Fst2')
fdesign.bandstop('Fp1,Fst1,Fst2,Fp2,Ap1,Ast,Ap2')
fdesign.bandstop('N,Fp1,Fst1,Fst2,Fp2')
fdesign.arbmag('N,F,A')
fdesign.arbmag('N,B,F,A')
fdesign.differentiator(`N')
fdesign.differentiator('N,Fp,Fst')
fdesign.hilbert('N,TW')
```

In pre-R2009a releases, the `firgr` function was used to construct the FIR equiripple filters in the previous code samples.

Beginning with R2009a, users of the Signal Processing Toolbox™ can use `fdesign` and `design` to specify and design filters. However, only concurrent users of the Filter Design Toolbox have access to the enhanced capabilities of `firgr`.

The change in the default algorithm for the specified FIR equiripple filter designs facilitates code sharing between Signal Processing Toolbox users and concurrent users of the Filter Design Toolbox.

Filter Design Toolbox users can access a new `UniformGrid` property, which allows them to design equiripple filters with either the `firpm` or `firgr` functions. Setting the `UniformGrid` property to `false` calls the `firgr` function

to construct the filter. Leaving the `UniformGrid` property unspecified, or setting it to `true`, calls the `firpm` function.

Note In general, both `firpm` and `firgr` filter designs will meet user specifications. However, increasing the number of frequency points near filter transition regions by using a nonuniform grid may improve the equiripple property and frequency response approximation of the filter. Using a nonuniform grid may also produce lower-order filters in minimum-order designs.

Compatibility Considerations

Code used to generate FIR equiripple filters in pre-R2009a releases may produce filters with different orders or coefficients in R2009a. Users can recreate their pre-R2009a default designs in the current release by explicitly setting the `UniformGrid` property to `false`. The following examples illustrate equivalent filter designs in R2008b and R2009a using the `UniformGrid` property.

Code in R2008b:

```
d=fdesign.lowpass;
Hd=design(d,'equiripple');
```

Equivalent code in R2009a:

```
d=fdesign.lowpass;
Hd=design(d,'equiripple','UniformGrid',false);
```

There are two exceptions in equiripple filter design where `fdesign` sets the `UniformGrid` property internally:

- If you create arbitrary magnitude designs that result in complex-valued filter coefficients, the `UniformGrid` property defaults to `true`.
- If you create a design in which you set the `MinPhase`, `MaxPhase`, or `StopbandShape` properties to nondefault values, the `UniformGrid` property defaults to `False`.

If you attempt to override the internal settings to these exceptions, you will get a warning message.

New Quantization Rounding Modes Added for Realizemdl, Block, and Filterbuilder

The rounding modes for the `realizemdl` method, the `block` method, and `filterbuilder` GUI-based wizard now include round and convergent modes.

The rounding mode determines how the filter quantizes numeric values that lie between representable values for the data format (word and fraction lengths).

When you set the rounding mode to `convergent`, values round to the closest representable integer, and ties round to the nearest even stored integer. This approach is the least biased of the methods available in the toolbox.

When the rounding mode is set to `round`, values round to the closest representable integer, negative ties round toward negative infinity, and positive ties round toward positive infinity.

Version 4.4 (R2008b) Filter Design Toolbox

This table summarizes what's new and changed in Version 4.4 (R2008b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are described here.

Fdesign Support for Additional Filter Types

`fdesign` now supports the following filter types:

- Raised-cosine and square-root raised-cosine
- Maxflat FIR and IIR lowpass
- Maxflat FIR highpass
- Constrained least-squares FIR lowpass, highpass, bandpass, bandstop

OptimizeScaleValues Property Added to `dfilt` Second-order Sections Object

OptimizeScaleValues property added to `dfilt` second-order sections object, allows the user to control skipping multiplication-by-1. Previous versions of Filter Design Toolbox software always optimized scale values equal to 1. Starting in R2008b, users can configure the software so it does not skip multiplication-by-1 between sections.

Version 4.3 (R2008a) Filter Design Toolbox

This table summarizes what's new and changed in Version 4.3 (R2008a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes Summary	Bug Reports	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are described here.

Realizemdl Support for Polyphase and Farrow Sample Rate Converters

You can now create Simulink models of Polyphase and Farrow Sample Rate Converters from the command line using the realizemdl command with mfilt.firsrc and mfilt.farrowsrc.

Fdesign Support for Farrow Sample Rate Converters

Multirate Farrow filters can efficiently implement arbitrary (including irrational) rate change factors. You can design Farrow sample rate converters using the new fdesign function fdesign.polysrc.

Expanded Demo: Efficient Sample Rate Conversion Between Arbitrary Factors

The expanded Efficient Sample Rate Conversion Between Arbitrary Factors demo now shows you how to design Farrow sample rate converters using the new fdesign.polysrc function.

Fdesign Support for Highpass Halfband Filters

The fdesign design class, `fdesign.halfband`, has a new property, `type`, that you can use to create a `lowpass` or `highpass` halfband filter.

Input Quantization Added to Models Generated by the Block Method

Models generated by the block method now include a new input quantization feature.

Compatibility Considerations

The new input quantization feature does not work with models generated using previous versions of the software. To restore the previous behavior, open the generated subsystem block, remove the Convert block, and reconnect the remaining block to the subsystem input.

Version 4.2 (R2007b) Filter Design Toolbox

This table summarizes what's new and changed in Version 4.2(R2007b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes	Bug Reports	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are described here.

Farrow Classes Moved into DFILT package

The two classes `farrow.fd` and `farrow.linearfd` have been moved into `dfilt.farrowfd` and `dfilt.farrowlinearfd`, respectively.

All Multirate Filter Structures Support Complex Coefficients

All structures for `mfilt` multirate filters can use complex coefficients.

New Multirate Farrow Filter Capable of Sample Rate Conversion

A new multirate filter, `mfilt.farrowsrc`, has been added to `mfilt`s, allowing for Farrow decimators and interpolators, and fractional decimators and fractional interpolators where the resulting interpolation or decimation factor is not an integer.

Updated Getting Started and User's Guide

Updates to structure of User's Guide and Getting started include sections about using `filterbuilder` and a new chapter about using integers with FIR filters

Compatibility Considerations

- The `farrow.fd` and `farrow.linearfd` functions have been removed. For more information about the removal of these functions, see "Functions Being Removed" on page 22.

Functions Being Removed

Function Being Removed	What Happens When You Run the Function?	Use this Function Instead	Compatibility Considerations
`farrow.fd`	Still runs	`dfilt.farrowfd`	Farrow filters are now constructed using the dfilt (discrete-time filter) structure. Instead of using farrow.fd, use dfilt.farrowfd. See the reference page for dfilt for more information.
`farrow.linearfd`	Still runs	`dfilt.farrowlinearfd`	As noted above Farrow filters are now created using the dfilt constructor.

Version 4.1 (R2007a) Filter Design Toolbox

This table summarizes what's new and changed in Version 4.1(R2007a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are described here.

Fdesign Support Added for Octave Band and Fractional Octave Band Filters

Octave-band and fractional-octave-band filters are commonly used in acoustics, for example, in noise control to perform spectral analysis. You can design octave band and fractional octave band filters using the fdesign function `fdesign.octave`. If you prefer to design these filters using a graphical approach, try the `filterbuilder` function.

Fdesign Support Added for Parametric Equalizer Filters

Parametric equalizers are digital filters used in audio for adjusting the frequency content of a sound signal. The Filter Design Toolbox provides the capability to design high-order IIR parametric equalizers. Such high-order designs provide much more control over the shape of each filter. You can design parametric equalizer filters using the fdesign function `fdesign.parameq`. The same functionality can be achieved through the GUI function `filterbuilder`.

Fdesign Support Added for Notch and Peak Filters

Filters that peak or notch at a certain frequency are useful to retain or eliminate a particular frequency component of a signal. You can design these peak or notch filters using the new functions `fdesign.peak` and `fdesign.notch`. The same functionality can be achieved through the GUI function `filterbuilder`.

Arbitrary Magnitude and Phase Added to Multirate Filters

When designing a multirate filter, you can specify the response of the resulting filter. Arbitrary magnitude as well as arbitrary magnitude and phase are two responses that have been added to an already extensive list. To understand how to use these responses when designing multirate filters, see `fdesign.rsrc`, `fdesign.decimator`, and `fdesign.interpolator`.

Support for Fixed-point Inputs and Tunable Parameters in Filter Design Toolbox Blocks

The blocks in the Filter Design Toolbox library of Signal Processing Blockset™ now support fixed-point and integer data types on their input and output ports. In addition, parameters of these blocks that do not change filter order or structure are now tunable. This feature is fully described in the "Signal Processing Blockset User's Guide".

Version 4.0 (R2006b) Filter Design Toolbox

This table summarizes what's new and changed in Version 4.0 (R2006b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are described here.

Blocks for Designing Filters Added to Signal Processing Blockset (Filter Design Toolbox Required)

This release adds a block library to the Signal Processing Blockset product. The new library contains blocks that design single- and multirate filters using the new `filterbuilder` filter design dialog boxes. If you have a license for Filter Design Toolbox, you can use the new blocks to design and implement filters in simulations. Users who do not have Fixed-Point Toolbox licenses can run models that contain the new blocks, but they cannot change the filter designs in the blocks.

Support for Automatically Converting Floating-Point Filters to Fixed-Point Based on Input Data

Two new aspects of analysis allow you to convert filters from floating-point to fixed-point format automatically:

- Scaled doubles logging

  `mfilt` and `dfilt` objects now support the scaled double data type. Scaled doubles data types act like fixed-point data types, allowing you to work with both fixed-point and scaled doubles in the same calculation. Allowing this overcomes the limitation that math is permitted only between the same data types. The value `ScaledDouble` has been added to the `DataType` property of the `numerictype` object. The following values have also been added to the `DataTypeMode` property of the `numerictype` object:

  - `Scaled double:  binary point scaling`
  - `Scaled double:  slope and bias scaling`
  - `Scaled double:  unspecified scaling`

- A new method, `autoscale`, automatically converts an input filter from floating-point format to fixed-point format based on the results of filtering a set of data. autoscale attempts to scale the filter to prevent overflows in all filter computations.

The combination of these new capabilities provides the conversion support.

New filterbuilder Function for Interactive Filter Design

A new function, filterbuilder, provides a graphical tool for designing single- and multirate filters in both floating-point and fixed-point forms. For more information, refer to `filterbuilder`.

Fixed-Point Farrow Filter Support

We upgraded the Farrow filters to provide fixed-point support. Similar to the fixed point for all other `dfilt` objects.

Fractional Delay Filter Design with fdesign.fracdelay

Add the new method `fdesign.fracdelay` for designing filters that offer fractional delays. To support designing filters with `fdesign.fracdelay`, we added a `lagrange` design method.

New freqrespest Function for Using Filtering to Estimate the Filter Frequency Response from Measured Data

Use this new method to estimate the frequency response of a `dfilt` or `mfilt` object. `freqrespest` uses filtering to estimate the filter response. To support this new method, we added `freqrespopts`, an object that contains the parameters for `freqrespest`.

Parallel Filter Objects Support Multirate filters

With the addition of this feature, you can create parallel structures composed of multirate filters. This extends the parallel filter capability using `dfilt.parallel` by letting you use multirate filters as the input filters. To create parallel filter objects from two or more multirate filters, the individual filters must have the same rate change factors.

Coupled-Allpass Designs Available for All fdesign Objects

All specification objects now support coupled-allpass filter designs as structures.

Version 3.4 (R2006a) Filter Design Toolbox

This table summarizes what's new and changed in Version 3.4 (R2006a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes Summary	Bug Reports	Printable Release Notes: PDF

New features and changes introduced in this version are described here.

Farrow Filters

The toolbox now provides Farrow filter capability with farrow. Using `farrow` you create filters based on the structure and a few options. After you create

your filter, various analysis functions, like `cost` and `fvtool`, help you determine your filter's fitness. `realizemdl` works with Farrow filters to produce blocks for Simulink models as well.

IIR Polyphase Decimators and Interpolators

Now the toolbox provides design tools for IIR polyphase decimators and interpolators using `fdesign.decimator` and `fdesign.interpolator`.

Single-Rate Allpass Discrete-time and Multirate Filters

Eight new filter function enable you to design both single-rate and multirate allpass filters, including wave digital filters.

- `dfilt.allpass`
- `dfilt.wdfallpass`
- `dfilt.cascadeallpass`
- `dfilt.cascadewdfallpass`
- `mfilt.iirdecim`
- `mfilt.iirwdfdecim`
- `mfilt.iirinterp`
- `mfilt.iirwdfinterp`

iirlinphase Method for Designing Linear Phase IIR Filters

The new iirlinphase method added in this release designs quasi-linear phase IIR filters from a halfband filter specification objects. Use the form

```
hd = design(d,'iirlinphase');
```

when `d` is a halfband specification object. Returned filter object `hd` is an IIR filter with linear phase in the passband.

Arbitrary Magnitude and Phase Filter Specification Object

The new `arbmagnphase` specification object added in this release designs filters where you define the filter magnitude response and the phase response explicitly. Use the form

```
d = fdesign.arbmagnphase();
```

`d` is a filter specification object where the magnitude and phase responses are specified as a complex frequency response you provide.

iirlinphase/elliptic Design for Hilbert Transformers

When you use `fdesign.hilbert` to create a Hilbert transformer specification object, the toolbox provides new `ellip` and `iirlinphase` design methods to implement the filter from the specification object as an elliptic filter or as a quasilinear phase IIR filter.

CIC Filters Provide Full Precision and Specify All Options

CIC filters, such as those created by `fdesign.decimator` and `fdesign.interpolator`, now supports full precision and three word and fraction length modes for the property `FilterInternals`.

- `FullPrecision` mode automatically sets the CIC filter word lengths and fraction lengths to maintain the maximum precision in the filtering process. (new)
- `MinWordLengths` mode lets you set the output word length for the filter.
- `SpecifyWordLengths` mode lets you specify the word lengths for all sections of the filter and for the output. But you cannot set the fraction lengths.
- `SpecifyPrecision` mode lets you set all fraction lengths and word lengths for the filter sections and for the output. (new)

For more information, refer to the reference pages for `fdesign.decimator` and `fdesign.interpolator` in the Filter Design Toolbox documentation.

The following example uses the `SpecifyPrecision` mode. Use a decimation factor of 5 and differential delay equal to 1.

```
d=fdesign.decimator(5,'cic',1) % M=5, D=1.

d =

          MultirateType: 'Decimator'
       DecimationFactor: 5
               Response: 'CIC'
          Specification: 'Fp,Ast'
            Description: {'Passband Frequency';'Aliasing Attenuation(dB)'}
      DifferentialDelay: 1
    NormalizedFrequency: true
                  Fpass: 0.01
                  Astop: 60

hm=design(d) % Use the default multisection design method.

hm =

          FilterStructure: 'Cascaded Integrator-Comb Decimator'
               Arithmetic: 'fixed'
        DifferentialDelay: 1
         NumberOfSections: 2
         DecimationFactor: 5
         PersistentMemory: false

    InputWordLength: 16
    InputFracLength: 15

    FilterInternals: 'FullPrecision'

hm.FilterInternals='specifyPrecision'

hm =

          FilterStructure: 'Cascaded Integrator-Comb Decimator'
               Arithmetic: 'fixed'
        DifferentialDelay: 1
```

```
      NumberOfSections: 2
      DecimationFactor: 5
      PersistentMemory: false

     InputWordLength: 16
     InputFracLength: 15

     FilterInternals: 'SpecifyPrecision'
  SectionWordLengths: [21 21 21 21]
  SectionFracLengths: [15 15 15 15]
    OutputWordLength: 21
    OutputFracLength: 15
```

Nearest Round Mode for dfilt and mfilt Objects

`dfilt` and `mfilt` objects include an additional mode for rounding the results of calculations —`nearest`. Results round to the nearest representable value in the chosen format. Changing this behavior makes `round` for `dfilt` and `mfilt` objects consistent with `round` in Simulink.

For more information about rounding, refer to fi in the Fixed Point Toolbox documentation, since the new rounding modes derive from the `fi` object used by fixed-point filters.

Compatibility Considerations

The new `round` mode behavior is now matches MATLAB `round` as well.

Cost Method

After you create a filter, you can use `cost` to determine the arithmetic cost when you filter data. `cost` returns estimates of the add, multiplies, and other operations that occur when you use the filter.

New Online Help for fdesign.structure

With the addition of more `fdesign` methods and specification objects, the toolbox changes the way you get help about a specific design method—the command-line help is now adaptive, recognizing the object and the design method in the help syntax.

The command-line help adapts to the filter specification object you have and the design method you intend to use, and provides help specifically for that combination of specification and method. For example, if you are designing a highpass filter and plan to use the `butter` design method, here is the new way to get help:

```
d = fdesign.highpass('fst,fp,ast,ap',0.45,0.55,1,60))

designmethods(d)


Design Methods for class fdesign.highpass (Fst,Fp,Ast,Ap):


butter
cheby1
cheby2
ellip
equiripple
ifir
kaiserwin

help(d,'butter') % New help command syntax with object and method.

DESIGN Design a Butterworth IIR filter.
HD = DESIGN(D, 'butter') designs a Butterworth filter specified by the
FDESIGN object D.

HD = DESIGN(..., 'FilterStructure', STRUCTURE) returns a filter with the
structure STRUCTURE.  STRUCTURE is 'df2sos' by default and can be any of
the following.

'df1sos'
'df2sos'
'df1tsos'
'df2tsos'

HD = DESIGN(..., 'MatchExactly', MATCH) designs a Butterworth filter
and matches the frequency and magnitude specification for the band
MATCH exactly.  The other band will exceed the specification.  MATCH
```

```
can be 'stopband' or 'passband' and is 'stopband' by default.

% Example #1 - Compare passband and stopband MatchExactly.
h     = fdesign.highpass('Fst,Fp,Ast,Ap', .7, .9, 60, 1);
Hd    = design(h, 'butter', 'MatchExactly', 'passband');
Hd(2) = design(h, 'butter', 'MatchExactly', 'stopband');

% Compare the passband edges in FVTool.
fvtool(Hd);
axis([.89 .91 -2 0]);
```

Suppose you decide to use an equiripple design method instead. Again, the `help` command with the specification object `d` and the method `equiripple` provides help for that combination.

```
help (d,'equiripple') % New help command syntax with object and method.

DESIGN Design a Equiripple FIR filter.
HD = DESIGN(D, 'equiripple') designs a Equiripple filter specified by the
FDESIGN object D.

HD = DESIGN(..., 'FilterStructure', STRUCTURE) returns a filter with the
structure STRUCTURE.  STRUCTURE is 'dffir' by default and can be any of
the following.

    'dffir'
    'dffirt'
    'dfsymfir'
    'dfasymfir'
    'fftfir'

HD = DESIGN(..., 'DensityFactor', DENS) specifies the grid density DENS
used in the optimization.  DENS is 16 by default.

HD = DESIGN(..., 'MinPhase', MPHASE) designs a minimum-phase filter
when MPHASE is TRUE.  MPHASE is FALSE by default.

HD = DESIGN(..., 'MinOrder', 'any') designs a minimum-order filter.
The order of the filter can be even or odd. This is the default.
```

```
HD = DESIGN(..., 'MinOrder', 'even') designs an minimum-even-order
filter.

HD = DESIGN(..., 'MinOrder', 'odd') designs an minimum-odd-order filter.

% Example #1 - Design a lowpass Equiripple filter in a transposed
structure.
       h  = fdesign.highpass('Fst,Fp,Ast,Ap');
       Hd = design(h, 'equiripple', 'FilterStructure', 'dffirt');
```

Notice that the content is different for the different methods. This makes it easier for you to know the options that apply to any combination of specification object and design method.

Info Method Updated to Include Filter Measurements

When you request information about a filter, the information now includes measurements of the filter characteristics based on the filter specifications. These are the same results that measure provides.

Measurement Display Changes

`measure` now shows more information and more specific information for any referred object. Now the display provides full text descriptions of the measured values, such as Sampling Frequency (instead of `Fs`) and Stopband Edge instead of `Fstop`. You should find this a more clear presentation of the filter information.

realizemdl Creates Additional Multirate Polyphase Filters

From the command line, you can use `realizemdl` to create realizations for `firdecim`, `firtdecim`, `firinterp`, and `linearinterp` filters. You can also apply `realizemdl` to the new IIR single-rate and multirate filters:

- `dfilt.allpass`
- `dfilt.wdfallpass`
- `dfilt.cascadeallpass`

- `dfilt.cascadewdfallpass`
- `mfilt.iirdecim`
- `mfilt.iirwdfdecim`
- `mfilt.iirinterp`
- `mfilt.iirwdfinterp`

Filter Design Object Now Called Filter Specification Object in the Documentation

When you use `fdesign.`*`response`*, MATLAB returns an object, usually called `d`, that contains the specifications for a filter design. In the documentation, the returned object is now called a *specification object*.

For clarity, we renamed the filter design object to filter specification object, because the object specifies the filter specifications, such as the magnitude response parameters. The specification object is not a filter, but an intermediate step in the filter design process that uses `fdesign.`*`response`* and `design`.

Version 3.3 (R14SP3) Filter Design Toolbox

This table summarizes what's new in Version 3.3 (R14SP3):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes Summary	Bug Reports	Printable Release Notes: PDF

New features and changes introduced in this version are

New Approach and Methods for Designing Filters

To unify and take advantage of the object-based nature of the filters in the toolbox, this release introduces a new design approach for filters using filter design objects and new design methods. In the new process, your filter design tasks begin with identifying the filter response you need for your application.

Here is the new process.

1. Determine the response type for your filter.
2. Choose the appropriate `fdesign.`*`response`* method to create a filter specifications object.
3. Select the specifications to use to define your filter object. Here you can select minimum order designs, IIR or FIR designs, or designs that specify the filter order as well as the frequency and magnitude specifications, among many choices.
4. Use `designmethods` to find out which design algorithms apply to your specifications object. Select the design method to use.
5. Use `designopts` with your design object to review the input arguments for your specifications object and your selected design method.
6. Now design your filter using your filter design object, the design method you chose, and the input arguments you require.

The result of this process is a filter object that meets your requirements in response shape or form and designed by the method you selected.

Based on three design methods and a new help method, you now design filters starting with the desired response and moving to the final filter. These new methods are:

Method	Description
`design`	Design a filter from the specifications using either a default method or a specified method.
`designmethods`	Find out which design methods apply to your current design object, including dependence on the specifications.
`designopts`	Find out which input arguments apply to your design method and design object.

Here is a short example that demonstrates the new design flow.
This `fdesign.lowpass` syntax uses the default response specification '`Fp,Fst,Ap,Ast`', where `Fp` is the passband edge, `Fst` is the stopband edge, `Ap` specifies the ripple in the passband, and `Ast` defines the desired stopband attenuation.

```
d = fdesign.lowpass % Select the response.
designmethods(d) % Determine the design methods available.
hd = design(d) % Design the filter using the default method (equiripple).

d =

               Response: 'Lowpass'
          Specification: 'Fp,Fst,Ap,Ast'
            Description: {4x1 cell}
    NormalizedFrequency: true
                  Fpass: 0.45
                  Fstop: 0.55
                  Apass: 1
                  Astop: 60
```

```
Design Methods for class fdesign.lowpass (Fp,Fst,Ap,Ast):


butter
cheby1
cheby2
ellip
equiripple
ifir
kaiserwin
multistage


hd =

     FilterStructure: 'Direct-Form FIR'
          Arithmetic: 'double'
           Numerator: [1x43 double]
    PersistentMemory: false
```

For more information about a particular design method, use the new help capability with your specification object and the design method as input arguments to `help`.

This help example gets more information about using the `equiripple` method to design a lowpass filter.

```
help(d,'equiripple') % Get help on using equiripple with your lowpass filter.

 DESIGN Design a Equiripple FIR filter.
    HD = DESIGN(D, 'equiripple') designs a Equiripple filter specified by the
    FDESIGN object H.

    HD = DESIGN(..., 'FilterStructure', STRUCTURE) returns a filter with the
    structure STRUCTURE.  STRUCTURE is 'dffir' by default and can be any of
    the following.

    'dffir'
    'dffirt'
```

```
'dfsymfir'
'fftfir'

HD = DESIGN(..., 'DensityFactor', DENS) specifies the grid density DENS
used in the optimization.  DENS is 16 by default.

HD = DESIGN(..., 'MinPhase', MPHASE) designs a minimum-phase filter
when MPHASE is TRUE.  MPHASE is FALSE by default.

HD = DESIGN(..., 'MinOrder', 'any') designs a minimum-order filter.
The order of the filter can be even or odd. This is the default.

HD = DESIGN(..., 'MinOrder', 'even') designs an minimum-even-order filter.

HD = DESIGN(..., 'MinOrder', 'odd') designs an minimum-odd-order filter.

HD = DESIGN(..., 'StopbandShape', SHAPE) designs a filter whose stopband
has the shape defined by SHAPE.  SHAPE can be 'flat', '1/f', or 'linear'.
SHAPE is 'flat' by default.

HD = DESIGN(..., 'StopbandDecay', DECAY) specifies the decay to use when
'StopbandShape' is not set to 'flat'.  When the shape is '1/f' this
specifies the power that 1/f is raised.  When shaped is 'linear' this
specifies the slope of the stopband in dB/rad/s.

% Example #1 - Design a lowpass Equiripple filter in a transposed structure.
   h  = fdesign.lowpass('Fp,Fst,Ap,Ast');
   Hd = design(h, 'equiripple', 'FilterStructure', 'dffirt');
```

New Way to Get Help for Filter Designs

Getting help about filter design and filter design methods is now dynamic and depends on the design object and method. When you want help about designing a filter, use `help` with both the filter specification object and the method to use to design the filter. Here is an example.

```
d = fdesign.bandpass(0.25,0.35,0.55,0.65,50,0.05,50)
designmethods(d)

d =
```

```
                Response: 'Bandpass'
           Specification: 'Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2'
             Description: {7x1 cell}
    NormalizedFrequency: true
                 Fstop1: 0.25
                 Fpass1: 0.35
                 Fpass2: 0.55
                 Fstop2: 0.65
                 Astop1: 50
                  Apass: 0.05
                 Astop2: 50


Design Methods for class fdesign.bandpass (Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2):

butter
cheby1
cheby2
ellip
equiripple
kaiserwin

help(d,'kaiserwin')
 DESIGN Design a Kaiser Windowed FIR filter.
    HD = DESIGN(D, 'kaiserwin') designs a Kaiser Windowed filter
    specified by the FDESIGN object H.

    HD = DESIGN(..., 'FilterStructure', STRUCTURE) returns a filter with the
    structure STRUCTURE.  STRUCTURE is 'dffir' by default and can be any of
    the following.

    'dffir'
    'dffirt'
    'dfsymfir'
    'dfasymfir'
    'fftfir'

    HD = DESIGN(..., 'ScalePassband', SCALE) scales the first passband so
    that it has a magnitude of 0 dB after windowing when SCALE is TRUE.
```

```
SCALE is TRUE by default.

% Example #1 - Design a bandpass Kaiser Windowed FIR filter.
   h  = fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2');
   Hd = design(h, 'kaiserwin', 'ScalePassband', false);
```

New Demo Programs to Introduce fdesign Filter Design Approach

This release adds many new tutorial demos that introduce you to using `fdesign` for your filter design tasks. To access the new demos, enter

```
demos
```

at the Command prompt. When the Help system opens, select `Filter Design > Tutorial Demos` from the Help Navigator tree in the left pane.

Alternatively, use the `demo` command with input arguments:

```
demo('toolbox','filter design')
```

to open the Demos folder showing the Filter Design Toolbox demos.

Fdesign Now Provides Arbitrary Magnitude Filter Response Design

The designs available for `fdesign` now include arbitrary magnitude response filters. You use `fdesign.arbmag` with input arguments to specify a vector of frequency points and response values at those points to define a custom filter response curve.

Fdesign Now Provides Hilbert and Differentiator Filter Response Design

The designs available for `fdesign` now include differentiator and Hilbert magnitude response filters. You use `fdesign.differentiator` or `fdesign.hilbert` with input arguments to specify a differentiator or Hilbert filter design object.

Fdesign Objects Now Use a Default Design Method When Available

`design` now applies a default design method if you do not provide the design method as an input. Usually the default method is `equiripple` for FIR filters and `ellip` for IIR filters.

butter and ellip Half-Band Design Methods Added for IIR Fdesign Objects

For designing IIR halfband filters with fdesign and design, we added both butter and ellip to the available design methods.

Added multistage Filter Design Method

In addition to single-stage filters, you can now design multistage filters from lowpass filter design objects by applying the `multistage` design method to the object.

For example, after you create a lowpass filter object, use `multistage` to create the filter as a multistage filter.

```
d=fdesign.lowpass(0.25,0.35,0.05,50);
hd = design(d,'multistage')

hd =

     FilterStructure: Cascade
            Stage(1): Direct-Form FIR Polyphase Decimator
            Stage(2): Direct-Form FIR Polyphase Decimator
            Stage(3): Direct-Form FIR Polyphase Interpolator
            Stage(4): Direct-Form FIR Polyphase Interpolator
    PersistentMemory: false
```

limitcycle Method Restored to the Toolbox

The function `limitcycle` is now available to test your fixed-point IIR filters for the limit cycle behavior.

normalizefreq Method Added to the Toolbox

To let you convert your filters to use normalized frequency specifications, rather than absolute frequency, the toolbox adds `normalizefreq` for filter objects.

New measure Method for Filters

A new method, `measure`, lets you measure the response of filters after you design them. `measure` returns the response values at a variety of frequencies in the filter magnitude response.

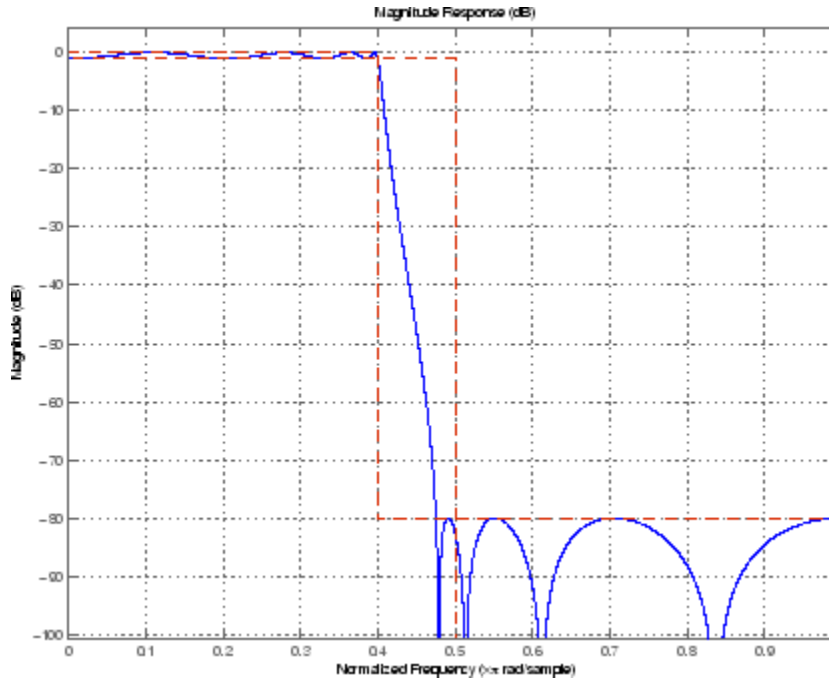
With Fdesign Objects, New Passband Zoom View Option

Selecting the **View > Passband** option from the menu bar automatically zooms the display to focus on the passband for the filter shown. Using an `fdesign` object to design your filter enables the Passband Zoom option in FVTool.

With Fdesign Objects, New Filter Specification Mask View Option

When you use FVTool or FDATool to display a filter response for a filter you design with an `fdesign` object, you see new masks that outline the filter passband, stopband, and transition regions as specified by your filter object.

The following graphic shows the mask for a lowpass filter. To display the specification mask, use a filter design object to construct your filter, and then display the filter in FVTool. Select **View > Specification Mask** from the menu bar in FVTool to see the specification mask.



Fdesign Object Display No Longer Shows Fs When the Design Object Uses Normalized Frequency

In this release, the default filter display no longer shows the sampling frequency `Fs` when you specify the filter to use normalized frequency instead of absolute frequency.

For cicinterp Objects, Changed the Order of the Properties in the Display

Reordered the listing of the filter properties in the default display of CIC filters. The new arrangement better matches the display organization for single rate filters.

For IIR Design Objects, Property Fcutoff is Now Called F3dB

The filter property `Fcutoff` is now called `F3dB` to be more descriptive.

Changes to the Displays in MATLAB for Filters

Some of the displays for filter objects, showing the properties and values, are different in this release. Some property names have changed, and some properties reordered to make the displays more logically grouped and consistent across the various objects. Among the changed displays are the CIC object property arrangements and the names of some properties for bandpass, bandstop, and general IIR filter objects.

Compatibility Considerations

If you depend on the displays in your code or scripts or programs, be sure to modify your work to accommodate the new display names and arrangements.

Obsolete Functions and Methods in This Release

The following methods are now obsolete.

Compatibility Considerations

As you see in the table, new methods replace them, providing the same or expanded design capability.

Obsolete Method	Replacement Method
`fdesign.decim`	`fdesign.decimator`
`fdesign.interp`	`fdesign.interpolator`
`fdesign.src`	`fdesign.rsrc`

The obsolete methods continue to work, but they may be removed in the future.

block Method for mfilt.firfracdecim Filter Objects No Longer Works

Changes in the FIR Sample Rate Change block in the Signal Processing Blockset software required that the `block` method for `firfracdecim` filters be made obsolete. You cannot use `block` to create a Simulink block from an `firfracdecim` filter object. To create a `block` from the `firfracdecim` object, convert the object to an `firsrc` object, and then use `block`.

```
hm = mfilt.firfracdecim(4,7); convert(hm,'firsrc') block(hm)
```

Compatibility Considerations

Programs that use the `block` method for `firfracdecim` require that you convert your `mfilt.firfracdecim` multirate filter to `firsrc` form using the `convert` method.

Version 3.2 (R14SP2) Filter Design Toolbox

This table summarizes what's new in Version 3.2 (R14SP2):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes Summary	Bug Reports	Printable Release Notes: PDF

New features and changes introduced in this version are

Improved Fixed-Point Support for FIR Filters

Four FIR filters now support fixed-point processing using the same properties or attributes and methods (mostly) that the fixed-point multirate filters use.

- `dfilt.dfasymfir`
- `dfilt.dffir`
- `dfilt.dffirt`
- `dfilt.dfsymfir`

With the improved filter objects, the properties for your discrete-time filters now look the same as your multirate filters. Unifying the look and feel makes working with the full range of filters in the toolbox easier and more clear.

Additionally, making the switch from floating-point to fixed-point by setting `Arithmetic` to `fixed` creates a fixed-point version of your floating-point filter that uses full precision arithmetic internally. The result—a fixed-point filter that most closely matches to your floating-point prototype. If your design cannot support the resources for this fixed-point implementation, you can start to adjust the fixed-point properties as you need.

To continue to use your existing fixed-point FIR filters, you have to upgrade them to the new format. The toolbox includes a new utility for making the transition—`legacyfixptfir`. Note that this utility is not covered in the Filter Design Toolbox documentation. You can get help by entering

```
help legacyfixptfir
```

at the MATLAB prompt.

For information about converting your existing fixed-point FIR filters to the new objects, refer to "Upgrading Your Existing Fixed-Point FIR Filters to the New Properties" on page 52.

Fixed-Point Linear and Hold Interpolators

Both `mfilt.holdinterp` and `mfilt.linearinterp` let you use fixed-point arithmetic. After you create the interpolator object, you can switch the setting for the `Arithmetic` property to `fixed` to use fixed-point interpolation.

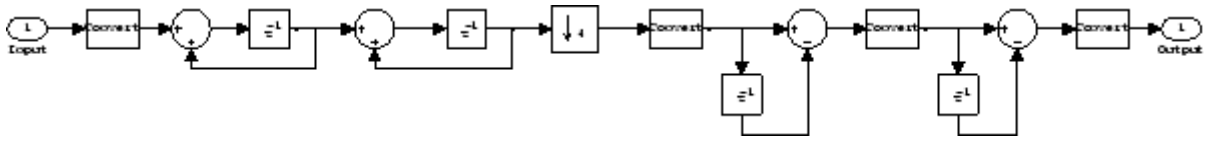
Both also support single-precision floating-point arithmetic.

realizemdl Creates CIC Filters

You can use `realizemdl` to construct CIC filters from basic blocks for processing signals. If you construct a CIC decimator filter, as shown in this example, `realizemdl` can make an atomic subsystem CIC filter block in the Simulink product for you.

```
hm=mfilt.cicdecim(4);   realizemdl(hm)
```

A new Simulink model window opens and you see a filter block. Double-clicking on the new block shows you the CIC filter subsystem.



Note You must have the Signal Processing Blockset to use `realizemdl` to implement CIC filters.

Context-Sensitive Help for FDATool Returns

FDATool now provides help for options on the quantization, multirate filter design, and frequency transformation panels. Access the new help feature either by right-clicking on an option and selecting **What's This** from the context menu, or clicking the **What's This** help icon on the tool bar.

Second-Order Section Filter View Options Available from the Command Line

In Filter Visualization Tool (FVTool), you can view second-order section filters as "individual sections," "cumulative sections," or as sections that you define. Now this functionality is available from the MATLAB command line, by using the `sosViewSettings` property of the FVTool object. In previous releases these view options were available only as options in the SOS View Settings dialog box in FVTool.

Access the FVTool object properties by launching FVTool with a filter object and including a left-hand side output argument:

```
handle = fvtool(hd)
```

`handle` now contains the FVTool properties, similar to the following listing — you use `set` and `get` to manipulate the property values.

```
handle=fvtool(hd)

handle =

     1
```

```
set(handle.sosviewsettings,'view')

ans =

    'Complete'
    'Individual'
    'Cumulative'
    'UserDefined'

set(handle.sosviewsettings,'view','individual')
```

In `SOSViewSettings`, the options are the same, with the same meaning, that you find in **View > SOS View Settings** in FDATool.

For more information about the `fvtool` properties, refer to `fvtool` in the Signal Processing Toolbox documentation or in the online Help system.

Function fdesign Specifies Filter Response with Specified Structure

You can use `fdesign.response` to specify a filter response and specify the filter structure to use during construction.

Upgrading Your Existing Fixed-Point FIR Filters to the New Properties

There is a utility named `legacyfixptfir` to ensure backward compatibility of your existing scripts and a function `update` to help you migrate to the new FIR filters. `legacyfixptfir` switches the preferences for your session between pre- and post-Filter Design Toolbox 3.2 FIR filters.

Here is an example of the process of converting your old FIR filters to the new form in this version of the toolbox.

Begin with an existing direct-form FIR filter `h` that you constructed with

```
h = dfilt.dffir
```

in an earlier version of the toolbox. First, use `legacyfixptfir` to retrieve `h` in the old format. Then convert `h` to the new form.

```
legacyfixptfir(true) % To get the old form of h.
h.Arithmetic='fixed'

h =

     FilterStructure: 'Direct-Form FIR'
          Arithmetic: 'fixed'
           Numerator: 1
    PersistentMemory: false

     CoeffWordLength: 16
      CoeffAutoScale: true
              Signed: true

     InputWordLength: 16
     InputFracLength: 15

    OutputWordLength: 16
          OutputMode: 'AvoidOverflow'

         ProductMode: 'FullPrecision'

           AccumMode: 'KeepMSB'
     AccumWordLength: 40
       CastBeforeSum: true

           RoundMode: 'convergent'
        OverflowMode: 'wrap'

update(h) % Convert h to the new properties.
h

h =

     FilterStructure: 'Direct-Form FIR'
          Arithmetic: 'fixed'
           Numerator: 1
    PersistentMemory: false

     CoeffWordLength: 16
```

```
     CoeffAutoScale: true
             Signed: true

    InputWordLength: 16
    InputFracLength: 15

    FilterInternals: 'SpecifyPrecision'

   OutputWordLength: 16
   OutputFracLength: 13

  ProductWordLength: 32
  ProductFracLength: 29

    AccumWordLength: 40
    AccumFracLength: 29

          RoundMode: 'convergent'
       OverflowMode: 'wrap'
```

Note the changes in properties. The filter performs the same way but the attributes are now updated to the newest form.

Filter Weights Have Been Removed from the Specifications in fdesign

The weights applied to the filter magnitude response are now design options. They are no longer properties of the `fdesign.typeobject`. To set them, pass them as property name/property value (PV) pairs to the appropriate filter design method, as shown in this example.

```
h = fdesign.lowpass('N,Fp,Fst',30) % Was 'N,Fp,Fst,Wp,Wst'.
                                   % Removed Wp and Wst.
hd = equiripple(h, 'Wpass', 3, 'Wstop', 25); % Specify the
                                             % weights here.
hd(2) = equiripple(h, 'Wpass', 3, 'Wstop', 1);
fvtool(hd)
```

Compatibility Summary for the Filter Design Toolbox Product

This table summarizes new features and changes that might cause incompatibilities when you upgrade from an earlier version, or when you use files on multiple versions of the product. Details about the compatibility effects appear with the description of the new feature or change in the New Features and Changes sections for the product.

Version (Release)	New Features and Changes with Version Compatibility Impact
Latest Release V4.7.1 (R2010b)	None
V4.7 (R2010a)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "Functions, Objects, Object Methods, and Object Properties Being Removed" on page 6
V 4.6 (R2009b)	None
V 4.5 (R2009a)	See the **Compatibility Considerations** subheading for each of these changes: • "Changes in SOS IIR Design Defaults" on page 14 • "Changes in FIR Equiripple Design Defaults" on page 15
V4.4 (R2008b)	None
V4.3 (R2008a)	See the **Compatibility Considerations** subheading for "Input Quantization Added to Models Generated by the Block Method" on page 20.
V4.2 (R2007b)	See the **Compatibility Considerations** subheading for "Functions Being Removed" on page 22.
V4.1 (R2007a)	None
V4.0 (R2006b)	None

Version (Release)	New Features and Changes with Version Compatibility Impact
V3.4 (R2006a)	See the **Compatibility Considerations** subheading for "Nearest Round Mode for dfilt and mfilt Objects" on page 32
V3.3 (R14SP3)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "Fdesign Object Display No Longer Shows Fs When the Design Object Uses Normalized Frequency" on page 46 • "Changes to the Displays in MATLAB for Filters" on page 47 • "Obsolete Functions and Methods in This Release" on page 47 • "block Method for mfilt.firfracdecim Filter Objects No Longer Works" on page 48
V3.2 (R14SP2)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "Upgrading Your Existing Fixed-Point FIR Filters to the New Properties" on page 52 • "Filter Weights Have Been Removed from the Specifications in fdesign" on page 54