1

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 2-2 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.

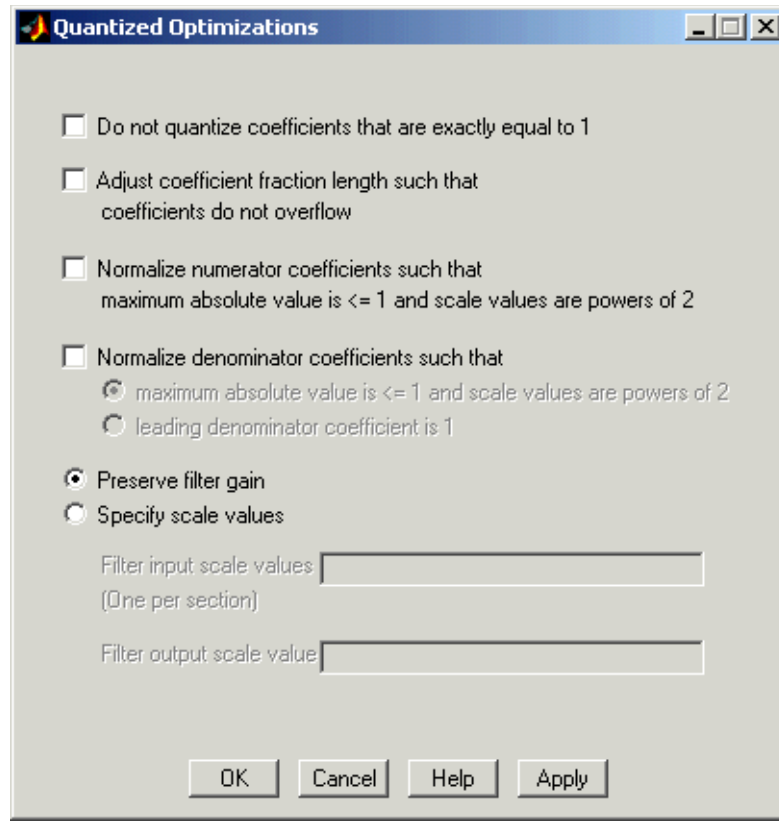
New Functions for Calculating Quantizer Noise Statistics

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.

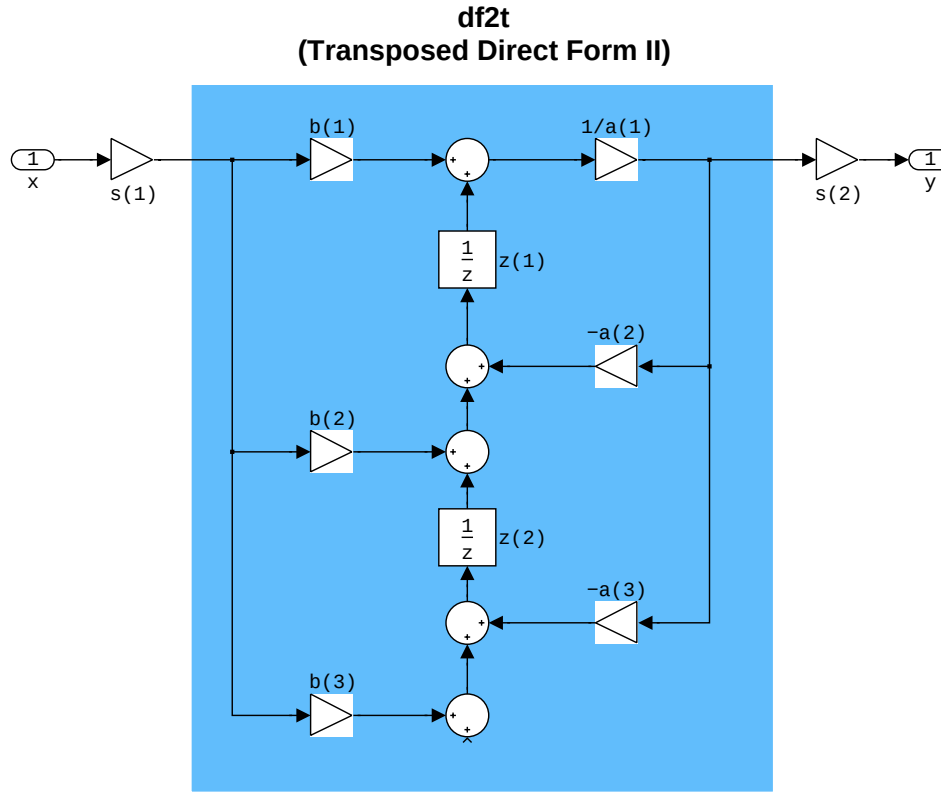


Figure 1-1: Original dft2 Structure Including s(1) and s(2)

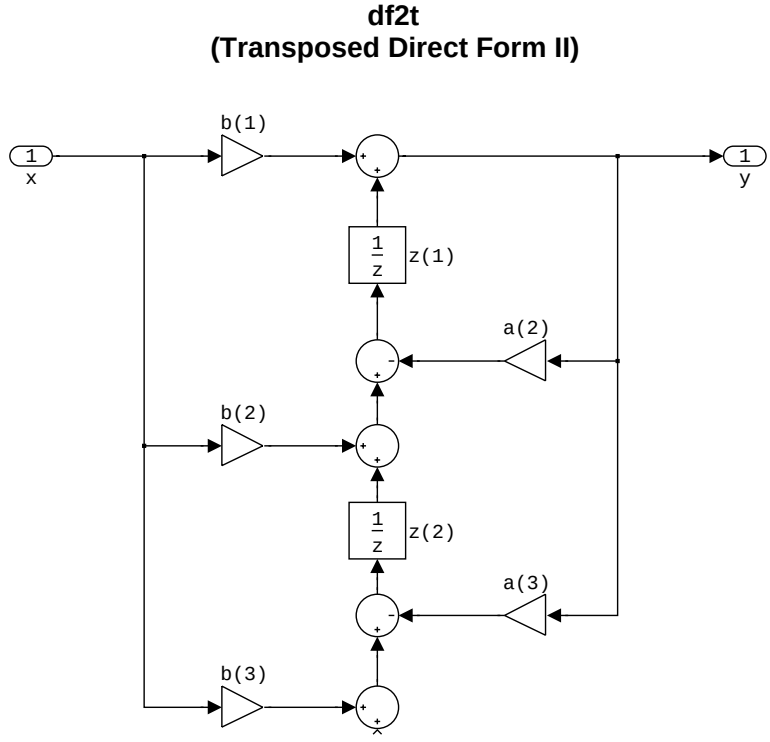


Figure 1-2: New df2t Structure Without Scale Values and with Coefficient a(1) = 1

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

This section includes a link to a description of known software and documentation problems in Version 2.2.

You can see a list of known software and documentation problems in Version 2.2. 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.

For a list of bugs reported in the previous release that remain open, see "Known Software and Documentation Problems" on page 2-6.