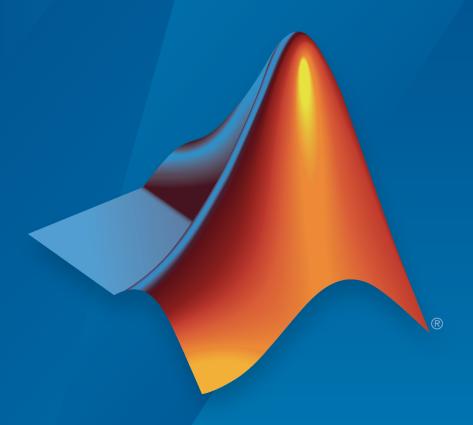Signal Processing Toolbox™ Release Notes

MATLAB®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

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

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

R2014b

R2014a

R2013b

R2011b

R2011a

R2010b

R2010a

R2009b

R2009a

R2006b

R2006a

R14SP3

R14SP2

R2015b

Version: 7.1

New Features

Bug Fixes

Reassigned Spectrogram: Sharpen the time-frequency localization of spectral estimates

The `spectrogram` function now computes reassigned spectra. The reassignment technique sharpens the time and frequency localization of spectral estimates and produces spectrograms that are easier to read and interpret. See the `spectrogram` reference page for examples. The enhanced function also enables you to specify a threshold such that spectrum values with less power are set to zero.

Hampel Filter and Improved Median Filtering: Detect outliers and remove them from data

The new `hampel` function detects and removes outliers from data. The function computes a moving median and considers as outliers those data points that deviate from the median by more than an adjustable number of standard deviations.

The `medfilt1` function now uses a faster algorithm and enables you to choose how to treat signals with `NaNs` and how to filter signal edges.

Order Analysis: Analyze vibrations in rotational machinery with order and frequency maps

The new `rpmordermap` and `rpmfreqmap` functions perform order analysis on rotational machinery. `rpmordermap` computes order maps of rotating systems, helping you visualize vibrations that vary linearly with rotational speed. `rpmfreqmap` computes frequency maps. You can also use these functions to plot the maps on interactive figures.

Envelope Detection: Extract analytic, peak, and RMS envelopes

The new `envelope` function computes the upper and lower envelopes of a waveform. You can compute the envelopes using an FFT-based analytic function, an FIR Hilbert filter, or the function peaks or RMS values.

Signal Alignment: Measure delay and align signals in time

The new `finddelay` and `alignsignals` functions compute the delay between two signals and align signals in time.

R2015a

Version: 7.0

New Features

Bug Fixes

Frequency Measurements: Compute mean and median frequencies using spectral estimates

This release introduces two functions that use spectral estimates to measure signal frequency:

- meanfreq estimates the mean frequency of the spectrum of a signal.
- medfreq estimates the median frequency of the spectrum of a signal.

These functions accept a time-domain signal, a power spectral density, or a power spectrum as input. They have visualization functionality and support multichannel signals.

Bandwidth Measurements: Compute occupied bandwidth and bandwidth at specified power levels

This release introduces two functions that measure signal bandwidth:

- obw returns the occupied bandwidth of a signal at a specified percentage.
- powerbw returns the power bandwidth of a signal at a specified level.

These functions accept a time-domain signal, a power spectral density, or a power spectrum as input. They have visualization functionality and support multichannel signals.

`resample` function accepts signals with nonuniform sampling or missing data

This release enhances the resample function to accept nonuniformly sampled signals and reconstruct them on a regular grid. The function also accepts NaNs in the input signal and treats them as missing data.

`pwelch` function computes maximum-hold and minimum-hold spectra

The pwelch function now computes the maximum-hold and minimum-hold power spectral density (PSD) estimates of a signal.

`spectrogram` function computes two-sided, centered, power, and PSD spectra

The spectrogram function now computes two-sided or centered spectra and can output the mean-square spectrum instead of the power spectral density. The enhanced function also plots a color bar with units.

Fast chirp Z-transform support for spectral analysis functions

All the spectral analysis functions offered by the Signal Processing Toolbox™ product let you input a vector with the frequencies at which to compute PSD estimates. The functions traditionally have used the Goertzel algorithm in those cases. Starting with this release, the functions might switch to a faster method based on the chirp Z-transform when the input frequency vector has many points and the points are uniformly spaced. The change applies to the following functions:

- Nonparametric methods: cpsd, mscohere, periodogram, pmtm, pwelch, spectrogram, tfestimate
- Parametric methods: pburg, pcov, pmcov, pyulear

R2014b

Version: 6.22

New Features

Bug Fixes

Spectral estimation of signals with nonuniform sampling or missing data

This release introduces a new spectral estimation function, plomb. The function can compute spectra of nonuniformly sampled signals or signals with missing samples by using the Lomb-Scargle algorithm. `plomb` also provides visualization functionality and support for multichannel data.

Multichannel support for spectral analysis functions

This release enhances several parametric and nonparametric spectral analysis functions by offering support for multichannel data. The functions continue to process vector data as single channels. For matrix input, the functions process the data column by column and return a matrix of spectra.

- Nonparametric methods: bandpower, cpsd, mscohere, periodogram, pmtm, pwelch, tfestimate.
- Parametric methods: arburg, arcov, armcov, aryule, pburg, pcov, pmcov, pyulear.

Peak finding with visualization and enhanced peak selection

The function findpeaks now lets you estimate the width of any peak and limit the results of peak searches by width or prominence. When called with no output arguments, the function plots the signal and annotates the value, width, and prominence of every peak.

R2014a

Version: 6.21

New Features

Bug Fixes

Compatibility Considerations

Simplified workflow for specification-based filter design

This release introduces a new filter design function, designfilt. A single command allows you to design lowpass, highpass, bandpass, bandstop, differentiator, and Hilbert filters, including minimum-order designs, in one step. The designfilt interface leads to readable and self-documenting code that is easy to maintain.

Use filter to filter signals with the digitalFilter objects generated by designfilt.

Use fvtool to visualize digital filters created using designfilt.

All the filter analysis and data filtering functions offered by the Signal Processing Toolbox product can be used with digital filters designed with designfilt.

designfilt also features a Filter Design Assistant—a smart run-time error-recovery mechanism that helps you correct faulty syntax and generates MATLAB® code at the Command Window or in the Editor.

Visualization of harmonic distortion measurements

This release enhances SNR, distortion, intermodulation, and spurious free dynamic range measurement functions to provide visualization functionality.

The function sfdr plots the spectrum of the input signal and labels its fundamental component and largest spur. The plot shades the spurious free dynamic range and displays its value; it shows the fundamental, the DC value, and the rest of the signal in different colors. sfdr now uses a Kaiser window by default.

The function sinad plots the spectrum of the input signal, labels its fundamental component, and displays its signal to noise and distortion ratio. The plot shows the fundamental component, the DC value, and the noise in different colors.

The function snr plots the spectrum of the input signal, labels its fundamental component and higher harmonics, and displays its signal-to-noise ratio. The plot shows the fundamental, the noise, and the DC value and harmonics using different colors.

The function thd plots the spectrum of the input signal, labels its fundamental component and harmonics, and displays its total harmonic distortion. The plot shows the fundamental, the harmonics, and the DC level and noise using different colors.

The function toi plots the spectrum of the input signal, annotates its lower and upper fundamentals and intermodulation products, and displays its third-order intercept point.

Changes in name-value pair arguments of measurement functions

The R2014a release changes the name-value pair arguments `'MidPct'` and `'PctRefLevels'` to `'MidPercentReferenceLevel'` and `'PercentReferenceLevels'` for the following signal measurement functions: dutycycle, falltime, midcross, pulseperiod, pulsesep, pulsewidth, risetime, settlingtime, slewrate, overshoot, and undershoot.

Compatibility Considerations

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`'MidPct'` name-value pair of measurement functions	Still runs	`'MidPercentRefere`	Replace all instances of `'MidPct'` with `'MidPercentReferenceLevel'`
`'PctRefLevels'` name-value pair of measurement functions	Still runs	`'PercentReference`	Replace all instances of `'PctRefLevels'` with `'PercentReferenceLevels'`

R2013b

Version: 6.20

New Features

Bug Fixes

Compatibility Considerations

Distortion, intermodulation, and SNR measurement functions

This release introduces four frequency-domain measurement functions that allow you to characterize the nonlinearity of a system. Quantitative measures of system linearity are important in a number of applications including audio system analysis, power electronics, and radio-frequency (RF) network analysis.

Use thd to measure the total harmonic distortion (THD) of a sinusoidal signal. THD is appropriate for signals with discrete spectra consisting of a fundamental frequency and one or more harmonics.

Use sinad to measure the signal to noise and distortion ratio (SINAD) of a signal. SINAD is appropriate for signals with mixed spectra consisting of a fundamental frequency with one or more harmonics and additive noise.

Use toi to measure the third-order intercept (TOI) point. TOI is used to quantify the intermodulation distortion of a system in response to a two-tone input.

Use snr to measure the signal-to-noise ratio (SNR) of a signal. SNR is appropriate when you want to measure the power of the fundamental frequency in comparison to the variance of additive noise.

Design raised cosine and Gaussian pulse-shaping filters

This release introduces two new functions for raised cosine and Gaussian pulse-shaping filter design. The new functions provide a uniform interface for transmit and receive pulse-shaping filters that is designed specifically for communication engineers.

Raised cosine filters are used to minimize intersymbol interference (ISI) by shaping pulses with a filter that satisfies the Nyquist ISI criterion. Use rcosdesign to design a raised cosine or square-root raised cosine finite impulse response pulse-shaping filter.

Gaussian pulse-shaping filters are used in modulation schemes such as Gaussian minimum shift keying (GMSK) to limit the bandwidth of pulse waveforms. Use gaussdesign to design a Gaussian finite impulse response pulse-shaping filter.

Functionality Being Removed or Changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`firrcos`	Still runs	rcosdesign	Replace all instances of `firrcos` with rcosdesign. See Migrate from `firrcos` to `rcosdesign`.
`gaussfir`	Still runs	gaussdesign	Replace all instances of `gaussfir` with gaussdesign. See Migrate from `gaussfir` to `gaussdesign`.
`fdesign.pulseshaping`	Still runs	rcosdesign or gaussdesign	Replace all instances of `fdesign.pulseshaping` with rcosdesign or gaussdesign, as appropriate. See Migrate from `fdesign.pulseshaping` to `rcosdesign` or `gaussdesign`.

Migrate from `firrcos` to `rcosdesign`

When you update legacy code using `firrcos` to use `rcosdesign` instead, keep in mind that `rcosdesign` designs a square-root raised cosine filter by default. If you want a normal raised cosine filter you must set the optional parameter, `shape`, to `'normal'`. `firrcos` has the opposite default behavior.

The number of samples per symbol must be an integer. Equivalently, the ratio `Fs/(2*Fc)`, where `Fs` is the sampling frequency and `Fc` is the cutoff frequency of the filter, must be an integer.

`firrcos` and `rcosdesign` use different normalizations for the filter coefficients. `firrcos` normalizes the coefficient so that the nominal passband gain is 1. `rcosdesign` sets the filter energy to 1.

firrcos	rcosdesign
| ```
Fs = 7000;
N = 16;
Fc = 500;
df = 375;

ag = firrcos(N,Fc,df,Fs)
``` | ```
R = df/2/Fc;
sps = Fs/2/Fc;


ga = rcosdesign(R,N/sps,sps,'normal');
ga = ga/max(ga)/sps
``` |
| ```
ug = firrcos(N,Fc,df,Fs,'sqrt')
``` | ```
gu = rcosdesign(R,N/sps,sps);
gu = gu/max(gu)*(1+R*(4/pi-1))/sps
``` |
| ```
N = 16;
Fc = 1000;
R = 0.25;
Fs = 8000;

b1 = firrcos(N,Fc,R,Fs, ...
     'rolloff','normal')
``` | ```
beta = R;
sps = Fs/(2*Fc);
span = N / sps;

b1n = rcosdesign(beta,span,sps,'normal');
b1n = b1n / max(b1n) / sps
``` |
| ```
N = 16;
Fc = 1000;
R = 0.25;
Fs = 8000;

b2 = firrcos(N,Fc,R,Fs, ...
     'rolloff','sqrt')
``` | ```
beta = R;
sps = Fs/(2*Fc);
span = N / sps;

b2n = rcosdesign(beta,span,sps,'sqrt');
b2n = b2n / max(b2n) /(pi*sps) ...
          * (pi*(1-beta) + 4*beta)
``` |

Migrate from `gaussfir` to `gaussdesign`

When you update legacy code using `gaussfir` to use `gaussdesign` instead, keep in mind the differences between the arguments in each function. In `gaussfir`, the second argument, `nt`, is the number of symbol periods between the start of the impulse response of the filter and its peak. It is thus equal to *half* the second argument, `span`, of `gaussdesign`. Moreover, `gaussdesign` has no default values for its second and third arguments. `gaussfir` respectively has 3 and 2.

gaussfir	gaussdesign
| ```
bt = 0.3;
h = gaussfir(bt)
``` | ```
hn = gaussdesign(bt,6,2)
``` |
| ```
nt = 5;
h = gaussfir(bt,nt)
``` | ```
span = 2*nt;
hn = gaussdesign(bt,span,2)
``` |
| ```
of = 3;
h = gaussfir(bt,nt,of)
``` | ```
sps = of;
hn = gaussdesign(bt,span,sps)
``` |

Migrate from `fdesign.pulseshaping` to `rcosdesign` or `gaussdesign`

The use of `fdesign.pulseshaping` is not recommended. Use rcosdesign or gaussdesign, as appropriate.

fdesign.pulseshaping	rcosdesign, gaussdesign

```sps = 6;```
 ```span = 4;```
 ```Beta = 0.25;```
 ```f1 = fdesign.pulseshaping(sps, ...```
 ```   'Square Root Raised Cosine', ...```
 ```   'Nsym,Beta',span,Beta);```
 ```d1 = design(f1);```
 ```n1 = d1.Numerator```
	```n1n = rcosdesign(Beta,span,sps);```
	 ```n1n = n1n / max(n1n) * (-1/(pi*sps) ...```
	 ``` * (pi*(Beta-1) - 4*Beta))```

```g1 = fdesign.pulseshaping(sps, ...```
 ```   'Square Root Raised Cosine', ...```
 ```   'N,Beta',sps*span,Beta);```
 ```h1 = design(g1);```
 ```k1 = h1.Numerator```
	```k1n = rcosdesign(Beta,span,sps);```
	 ```k1n = k1n / max(k1n) * (-1/(pi*sps) ...```
	 ``` * (pi*(Beta-1) - 4*Beta))```

```f2 = fdesign.pulseshaping(sps, ...```
 ```   'Raised Cosine', ...```
 ```   'Nsym,Beta',span,Beta);```
 ```d2 = design(f2);```
 ```n2 = d2.Numerator```
	```n2n = rcosdesign(Beta,span,sps,'normal');```
	 ```n2n = n2n/max(abs(n2n))/sps```

```g2 = fdesign.pulseshaping(sps, ...```
 ```   'Raised Cosine', ...```
 ```   'N,Beta',sps*span,Beta);```
 ```h2 = design(g2);```
 ```k2 = h2.Numerator```
	```k2n = rcosdesign(Beta,span,sps,'normal');```
	 ```k2n = k2n/max(abs(k2n))/sps```

```BT = 0.3;```
 ```f3 = fdesign.pulseshaping(sps, ...```
 ```   'Gaussian', ...```
 ```   'Nsym,BT',span,BT);```
 ```d3 = design(f3);```
 ```n3 = d3.Numerator```
	```n3n = gaussdesign(BT,span,sps)```

R2013a

Version: 6.19

New Features

Bug Fixes

Compatibility Considerations

Functions to measure equivalent noise bandwidth, band power, and spurious-free dynamic range

This release introduces three new functions for power measurements in the frequency domain. To measure the average power contained in a specified frequency interval, use bandpower. You can measure the spurious-free dynamic range using sfdr. The spurious-free dynamic range is the ratio of powers for the fundamental, or carrier frequency, to the next largest component. The function enbw measures the equivalent noise bandwidth of a window. The equivalent noise bandwidth is the width of an ideal rectangular filter with peak power equal to the peak power of the window. The product of the peak power and the equivalent noise bandwidth is equal to the power obtained by integrating under the magnitude-squared Fourier transform of the window function.

Function interface to compute power spectrum with confidence intervals and DC-centered spectra

This release enhances functions for nonparametric and parametric power spectral density (PSD) function estimation. In R2013a, nonparametric PSD estimators provide confidence intervals based on the chi-square probability distribution. Parametric PSD estimators provide confidence intervals based on the Gaussian probability distribution. All nonparametric and parametric PSD estimators include the ability to obtain a two-sided PSD estimate with 0 frequency (DC) in the center. See periodogram , pwelch, and pmtm for information on obtaining confidence intervals and DC centered spectra for nonparametric PSD estimation. See pburg , pcov, pmcov, and pyulear for information on obtaining confidence intervals and DC centered spectra for parametric PSD estimation.

Function interface for analysis and implementation of single-precision filters

This release enhances a number of filter analysis and implementation functions to accept single-precision floating point inputs. In R2013a, you can use single-precision filter coefficients with filter analysis functions, such as freqz, fvtool, grpdelay, impz, phasez, and zerophase. If you input single-precision filter coefficients, the analysis function outputs are single precision.

In R2013a, the filtering function, sosfilt, accepts single-precision inputs for both the filter coefficients and data. If either the second-order section matrix input or the data is single-precision, `sosfilt` outputs single-precision data.

The filtering functions `fftfilt`, `filtfilt`, and `latcfilt` still require double-precision inputs.

Function interface for analysis of second-order section (biquad) filters

The R2013a release adds SOS matrix input support for filter analysis functions, such as freqz, fvtool, grpdelay, impz, phasez, and zerophase.

Code generation support for Signal Processing Toolbox functions

This release removes the requirement for DSP System Toolbox™ to generate standalone C and C++ code for supported Signal Processing Toolbox functions. You must have MATLAB Coder™ and Signal Processing Toolbox software to generate standalone C and C++ code for supported functions. See Functions Supported for Code Generation for a list of Signal Processing Toolbox functions with code generation support if you have MATLAB Coder software.

Functionality Being Removed or Changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`dspdata.avgpower`	Still runs	bandpower	Replace all instances of `dspdata.avgpower` with `bandpower`
`dspdata.msspectrum`	Still runs	periodogram, pwelch	Replace all instances of `dspdata.msspectrum` with the appropriate function interface
`dspdata.psd`	Still runs	pburg, pcov, periodogram, pmcov, pmtm, pwelch, pyulear	Replace all instances of `dspdata.psd` with the appropriate function interface
`dspdata.pseudospectru`	Still runs	peig, pmusic	Replace all instances of `dspdata.pseudospectrum` with the appropriate function interface

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`dspdata.sfdr`	Still runs	sfdr	Replace all instances of `dspdata.sfdr` with `sfdr`
`sigwin.barthannwin`	Still runs	barthannwin	Replace all instances of `sigwin.barthannwin` with `barthannwin`
`sigwin.bartlett`	Still runs	bartlett	Replace all instances of `sigwin.bartlett` with `bartlett`
`sigwin.blackman`	Still runs	blackman	Replace all instances of `sigwin.blackman` with `blackman`
`sigwin.blackmanharris`	Still runs	blackmanharris	Replace all instances of `sigwin.blackmanharris` with `blackmanharris`
`sigwin.bohmanwin`	Still runs	bohmanwin	Replace all instances of `sigwin.bohmanwin` with `bohmanwin`
`sigwin.chebwin`	Still runs	chebwin	Replace all instances of `sigwin.chebwin` with `chebwin`
`sigwin.flattopwin`	Still runs	flattopwin	Replace all instances of `sigwin.flattopwin` with `flattopwin`
`sigwin.gausswin`	Still runs	gausswin	Replace all instances of `sigwin.gausswin` with `gausswin`
`sigwin.hamming`	Still runs	hamming	Replace all instances of `sigwin.hamming` with `hamming`
`sigwin.hann`	Still runs	hann	Replace all instances of `sigwin.hann` with `hann`

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`sigwin.kaiser`	Still runs	kaiser	Replace all instances of `sigwin.kaiser` with `kaiser`
`sigwin.nuttallwin`	Still runs	nuttallwin	Replace all instances of `sigwin.nuttallwin` with `nuttallwin`
`sigwin.parzenwin`	Still runs	parzenwin	Replace all instances of `sigwin.parzenwin` with `parzenwin`
`sigwin.rectwin`	Still runs	rectwin	Replace all instances of `sigwin.rectwin` with `rectwin`
`sigwin.taylorwin`	Still runs	taylorwin	Replace all instances of `sigwin.taylorwin` with `taylorwin`
`sigwin.triang`	Still runs	triang	Replace all instances of `sigwin.triang` with `triang`
`sigwin.tukeywin`	Still runs	tukeywin	Replace all instances of `sigwin.tukeywin` with `tukeywin`
`spectrum.burg`	Still runs	pburg	Replace all instances of `spectrum.burg` with `pburg`
`spectrum.cov`	Still runs	pcov	Replace all instances of `spectrum.cov` with `pcov`
`spectrum.eig`	Still runs	peig	Replace all instances of `spectrum.eig` with `peig`
`spectrum.mcov`	Still runs	pmcov	Replace all instances of `spectrum.mcov` with `pmcov`

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`spectrum.mtm`	Still runs	pmtm	Replace all instances of `spectrum.mtm` with `pmtm`
`spectrum.music`	Still runs	pmusic	Replace all instances of `spectrum.music` with `pmusic`
`spectrum.periodogram`	Still runs	periodogram	Replace all instances of `spectrum.periodogram` with `periodogram`
`spectrum.welch`	Still runs	pwelch	Replace all instances of `spectrum.welch` with `pwelch`

R2012b

Version: 6.18

New Features

Bug Fixes

Compatibility Considerations

Signal Browser in SPTool

The R2012b release introduces a revamped Signal Browser GUI. Whenever you select one or more signals in the **Signals** list box and click the **View** button, The Signal Browser GUI launches from the SPTool GUI. The Signal Browser allows you to perform the following additional operations:

- Time Domain Measurements Panels — The Signal Browser includes new side panels labeled **Trace Selection**, **Cursor Measurements**, **Signal Statistics**, **Bilevel Measurements**, and **Peak Finder**.

  - The **Trace Selection** panel allows you to choose which signal to make the active signal when you select multiple input signals in the **Signals** list box in SPTool.

    This panel replaces the **Select Trace** toolbar button ( ) and **Select a trace** dialog box that were used in previous releases to choose an active signal. In the Signal Browser menu, select **Tools** > **Measurements** > **Trace Selection**.

  - The **Cursor Measurements** panel displays screen cursors. This panel replaces the **Markers** menu and all the corresponding toolbar buttons that were used in previous releases to show and configure markers. In the Signal Browser toolbar, click the Cursor Measurements button ( ). Alternatively, in the Signal Browser menu, select **Tools** > **Measurements** > **Cursor Measurements**.

  - The **Signal Statistics** panel displays the maximum, minimum, peak-to-peak difference, mean, median, and RMS values of a selected signal. It also displays the times at which the maximum and minimum values occur. In the Signal Browser toolbar, click the Signal Statistics button ( ). Alternatively, in the Signal Browser menu, select **Tools** > **Measurements** > **Signal Statistics**.

  - The **Bilevel Measurements** panel displays information about a selected signal's transitions, overshoots or undershoots, and cycles. In the Signal Browser toolbar, click the Bilevel Measurements button ( ). Alternatively, in the Signal Browser menu, select **Tools** > **Measurements** > **Bilevel Measurements**.

  - The **Peak Finder** panel displays maxima and the times at which they occur. These displays allow you to modify the settings for peak threshold, maximum number of peaks, and peak excursion. In the Signal Browser toolbar, click the Peak Finder button ( ). Alternatively, in the Signal Browser menu, select **Tools** > **Measurements** > **Peak Finder**.

- Multiple Display Support — R2012b enhances the Signal Browser by allowing you to choose to have multiple displays. This feature allows you to tile your screen into a number of separate displays, up to a grid of 4 rows and 4 columns. You may find multiple displays useful when you select multiple input signals in the **Signals** list box in SPTool. To set the number of displays, in the Signal Browser toolbar, click the Layout button (⊞). Alternatively, in the Signal Browser menu, select **View > Layout**.

- Style dialog box — R2012b enhances the Signal Browser by allowing you to customize the style of displays using a Style dialog box. You can change the color of the figure containing the displays, the background and foreground colors of display axes, and properties of lines in a display. The Style dialog box replaces the **Line Properties** toolbar button and **Edit Line** dialog box that were used in previous releases for customizing line properties. To view or modify the line style of the active signal, in the Signal Browser menu, select **View > Style**. You can also right-click on the display and select **Style**. The Signal Browser — Style dialog box opens.

  - Sampled data as stairs — The Signal Browser can display a signal as a *stairstep* graph. A stairstep graph includes only horizontal lines and vertical lines. Each horizontal line represents the signal value for a discrete sample period and is connected to two vertical lines. Each vertical line represents a change in values occurring at a sample. Using this approach is equivalent to using the MATLAB stairs function. Stairstep graphs are useful for drawing time history graphs of digitally sampled data. To display a sampled signal as a Stairstep graph, in the Signal Browser — Style dialog box, set the **Plot type** parameter to Stairs.

- Properties dialog box — R2012b enhances the Signal Browser by providing a central location where you can modify the properties of a display. To change options for a display, in the Signal Browser menu, select **View > Properties**. You can also right-click on the display and select **Properties**. The Signal Browser — Visuals:Time Domain Options dialog box opens.

  - Complex data support — The Signal Browser accepts complex-valued input signals and can visualize them in two distinct fashions. By default, the complex data is displayed in real and imaginary form as different-colored lines on the same axes. Alternately, you can display the magnitude and phase of the signal on separate axes in the same display. To change the complex data options, in the Signal Browser — Visuals:Time Domain Options dialog box, select the **Display** tab. Then, select or clear the **Plot signal(s) as magnitude and phase** check box.

  - Ability to change the time units of the display — The Signal Browser allows you to label the *time*-axis in three different ways. First, you can confirm the default

operation, in which the Signal Browser displays time in metric units. The Signal Browser chooses the appropriate metric units, based on the minimum *time*-axis limit and the maximum *time*-axis limit of the scope window. Second, you can verify that the *time*-axis is always labeled as `Time (seconds)` and that the appropriate power of 10 appears in the bottom-right corner of the Time Scope display. Finally, you remove the units in the *time*-axis label entirely. To change the manner in which the time units are displayed, in the Signal Browser — Visuals:Time Domain Options dialog box, select the **Main** tab. Then, set the **Time Units** parameter to either `Metric (based on Time Span)`, `Seconds`, or `None`, respectively.

See the Signal Browser reference topic for more information.

Compatibility Considerations

The R2012b Signal Browser GUI replaces the tool available in previous releases that was also named Signal Browser. In R2012b, you can still use the Signal Browser from previous releases, which is hereafter referred to as the Legacy Browser. To do so, execute the following steps.

1 In the SPTool menu, select **File** > **Preferences**.
2 In the list at the left side, select **Signal Browser**.
3 Under **Signal Browser**, select the **Use Legacy Browser (to be removed)** check box.
4 Click **OK**.

Now when you select one or more signals from the **Signals** list and click the **View** button, the Legacy Browser opens.

GPU acceleration for `xcorr`, `xcorr2`, `fftfilt`, `xcov`, and `cconv` functions

The R2012b release introduces GPU acceleration for xcorr, xcorr2, fftfilt, xcov, and cconv. GPU acceleration for these functions requires Parallel Computing Toolbox™ software and meeting the GPU computing requirements detailed here:
.

If you have the Parallel Computing Toolbox, you can use gpuArray to create a GPUArray object. The supported Signal Processing Toolbox functions accept GPUArray objects as inputs.

R2012a

Version: 6.17

New Features

Measurements for Bilevel Pulse Waveforms

In R2012a, you can perform a number of basic measurements on bilevel pulse waveforms. These measurements include:

- State levels — You can estimate the state levels of a bilevel waveform using the histogram method with statelevels.

- Transition metrics — You can measure the rise time, fall time, and mid-reference level instants of waveform transitions. See the help for midcross, risetime, falltime, and slewrate for details. Additionally, R2012a introduces pulse metrics to measure bilevel waveform behavior in pretransition and posttransition regions including, overshoot, undershoot, and settlingtime.

- Duration metrics — See the help for pulsewidth, pulseperiod, pulsesep, and dutycycle for details.

Signal Statistics

The R2012a release introduces a number of basic signal statistics to augment the statistical measurements available in base MATLAB. See the function reference pages for rms, rssq, peak2peak, and peak2rms for details.

R2011b

Version: 6.16

New Features

Bug Fixes

Compatibility Considerations

Passband and Stopband Weights for Fixed-Order Unconstrained Partial Band Differentiator Filters

In R2011b, you can specify passband and stopband weights for a fixed-order unconstrained partial band differentiator filter design. You can access this capability through filterbuilder and fdesign.differentiator. With `fdesign.differentiator`, use the specification string `'N,Fp,Fst'`, and set the design method to `'equiripple'`. The following example shows you how to see the passband weight, `Wpass`, and stopband weight, `Wstop`, design options.

```
d = fdesign.differentiator('N,Fp,Fst',30,0.25,0.5);
designopts(d,'equiripple')
```

Specify the passband or stopband weight values when you design your equiripple filter. For example:

```
Hd = design(d,'equiripple','Wstop',4);
```

Numerator and Denominator Order Specifications Added to filterbuilder for Lowpass and Highpass Butterworth Designs

In R2011b, you can specify different numerator and denominator orders in filterbuilder for lowpass and highpass Butterworth (maxflat) designs.

Access this option in `filterbuilder` by setting **Impulse response** under **Filter specifications** to IIR and **Order mode** to Specify. The default is equal order for both the numerator and denominator. Check **Denominator order** to specify a different denominator order.

Conversion of Error and Warning Message Identifiers

For R2011b, Signal Processing Toolbox error and warning message identifiers have changed.

Compatibility Considerations

If you have scripts or functions that use these changed message identifiers, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages. You can also use them in code that uses a `try/catch` statement and performs an action based on a specific error identifier.

If your code checks for a message identifier in a warning or error, you must update it to check for the new warning or error instead. To determine the identifier for a warning, run the following command just after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable `MSGID`.

To determine the identifier for an error, run the following command just after you see the error:

```
exception = MException.last;
MSGID = exception.identifier;
```

Note: Warning messages indicate a potential issue with your code. While you can turn off a warning, a suggested alternative is to change your code so that your code does not generate warnings.

R2011a

Version: 6.15

New Features

Bug Fixes

Compatibility Considerations

Enhancements to filtfilt

In R2011a, there are two major enhancements to filtfilt:

1 filtfilt has been completely rewritten to improve performance. Actual performance improvement depends on your hardware, filter length, signal length, and number of channels.

2 filtfilt now accepts IIR filters in second-order section (biquad) form.

Symmetric Window Option for Blackman-Harris Windows

In R2011a, blackmanharris and nuttallwin have a symmetric window design option. The input argument SFLAG controls the window option and defaults to `'symmetric'`. `'periodic'` returns a N-periodic window. The default symmetric option is preferred in FIR filter design because it results in linear phase. In spectral analysis applications, the periodic option is preferred.

Compatibility Considerations

In releases previous to R2011a, blackmanharris and nuttallwin only return N-periodic windows. To reproduce behavior in R2011a consistent with behavior in previous releases, use:

```
win = blackman(N,'periodic');   % N is the window length
```
or

```
win = nuttallwin(N,'periodic'); % N is the window length
```

rectpuls Returns Double-Precision Vector

In R2011a, rectpuls returns a double-precision vector instead of a logical vector.

Compatibility Considerations

In previous releases, rectpuls returns a logical vector. To produce behavior in R2011a consistent with previous releases, cast the output of `rectpuls` to a logical vector.

```
t = linspace(0,1,0.01);
y = logical(rectpuls(t));
```

Code Generation from MATLAB and Fixed-Point MEX-File Generation

In R2011a, MathWorks® is no longer using the term *Embedded MATLAB* to refer to the language subset that supports code generation from MATLAB algorithms. This nomenclature incorrectly implies that the generated code is used in embedded systems only.

The new term is *code generation from MATLAB*. This terminology better reflects the full extent of the capability for translating MATLAB algorithms into readable, efficient, and compact MEX and C/C++ code for deployment to both desktop and embedded systems.

Signal Processing Toolbox users who have the DSP System Toolbox and MATLAB Coder software can generate deployable C/C++ code and MEX files using supported functions in the Signal Processing Toolbox.

You can find material on using Code Generation from MATLAB with the Signal Processing Toolbox software in Code Generation from MATLAB Support in Signal Processing Toolbox.

Users who have the DSP System Toolbox and Fixed-Point Toolbox™ can accelerate MEX-files for fixed-point applications using fiaccel.

Compatibility Considerations

The functionality associated with C/C++ and MEX code generation from MATLAB has changed in R2011a. These changes include:

- The compiler flag `#codegen` replaces `#eml`.
- codegen replaces `emlc` and `emlmex` for generating deployable C/C++ code and MEX-files from MATLAB algorithms. You must have the DSP System Toolbox and the MATLAB Coder software to use `codegen`.
- fiaccel replaces `emlmex` for generating fixed-point MEX code from MATLAB algorithms. To use `fiaccel`, you must have the DSP System Toolbox and Fixed-Point Toolbox software and your MATLAB code must satisfy the conditions described on the fiaccel reference page.

R2010b

Version: 6.14

New Features

Bug Fixes

Embedded MATLAB Support for Additional Signal Processing Toolbox Functions

In R2010b, Embedded MATLAB® supports upsample and downsample in the Signal Processing Toolbox.

R2010a

Version: 6.13

New Features

Bug Fixes

Compatibility Considerations

Single-Precision Support Added for dfilt Objects

In R2010a, users can construct dfilt objects with single-precision floating point arithmetic. Set the `Arithmetic` property to `'single'` to obtain a single-precision floating point representation of the filter coefficients.

Embedded MATLAB Support for Additional Signal Processing Toolbox Functions

In R2010a, Embedded MATLAB supports additional functions in the Signal Processing Toolbox. You can find a comprehensive list of supported functions in the Function Library Reference. You can find examples of using supported functions with the Signal Processing Toolbox software at Code Generation from MATLAB Support in Signal Processing Toolbox.

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
`addsection` (Method for `dfilt.cascade` and `dfilt.parallel` objects)	Errors	`addstage` method	Replace all instances of the `addsection` method with the `addstage` method for dfilt.cascade and dfilt.parallel objects.
`FFTLength` (Property of `spectrum` objects)	Errors	`'NFFT'` parameter in `psd` or `msspectrum` methods	Replace all instances of the `FFTLength` property in spectrum objects by the `'NFFT'` parameter in the `psd` or `msspectrum` methods.

Name	What Happens When you use the Function, Object, Object Method, or Object Property	Use Instead	Compatibility Considerations
freqzplot	Warns	fvtool	Replace all instances of freqzplot with fvtool.
removesection (Method for dfilt.cascade and dfilt.parallel objects)	Errors	removestage method	Replace all instances of the removesection method with the removestage method for dfilt.cascade and dfilt.parallel objects.
Section (Property of dfilt.cascade and dfilt.parallel objects)	Errors	Stage property	Replace all instances of the Section property with the Stage property for dfilt.cascade and dfilt.parallel objects.
Sidelobe_atten (Property of sigwin.chebwin object)	Errors	SidelobeAtten property	Replace all instances of the Sidelobe_atten property with the SidelobeAtten property for sigwin.chebwin objects.
specplot	Warns	plot method with DSPDATA object	Replace all instances of specplot with the plot method for DSPDATA objects.

Warning for Filter Designer in SPTool

In R2010a, the use of Filter Designer in sptool is not recommended. Use fdatool instead. Under `File —> Preferences-> Filter Designer` in SPTool, you may still select to use Filter Designer, but you will receive a warning that Filter Designer will be removed in a future release. If you select Filter Designer, you will be prompted to change your preferences in the `sigprefs.mat` and `startup.spt` files when you exit SPTool. Changing your preferences to use Filter Designer results in a warning each time SPTool starts. See Setting Preferences and Saving and Loading Sessions for details.

Compatibility Considerations

Because a future release will remove Filter Designer , use fdatool instead. Filters created in Filter Designer are not compatible with FDATool. Under `File —> Preferences-> Filter Designer` in sptool, you can select to use FDATool. You receive a prompt to convert filters created in Filter Designer to a format compatible with FDATool. When you exit sptool after changing your preferences, you receive another prompt, instructing you to update your preferences to use FDATool.

R2009b

Version: 6.12

New Features

Bug Fixes

Embedded MATLAB Support Added to Signal Processing Toolbox Functions

In R2009b, Embedded MATLAB supports the generation of embeddable C code for a subset of Signal Processing Toolbox filter design and window generation functions. You must install both the Signal Processing Toolbox and Signal Processing Blockset™ software to use this feature. Depending on which Embedded MATLAB feature you wish to use, additional products are required. The generated C code meets the strict memory and data type requirements of embedded target environments. See Code Generation from MATLAB Support in Signal Processing Toolbox for a list of supported Signal Processing Toolbox functions and examples.

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 objects to constant blocks. The coefficients also appear in the MATLAB workspace providing tunability to the realized Simulink model. See dfilt for a list of supported filter structures and any restrictions.

R2009a

Version: 6.11

New Features

Bug Fixes

New filter design approach using fdesign and filterbuilder

A new and more robust way to design filters has been added to the toolbox —fdesign objects and the filterbuilder GUI. The following filter responses are supported: lowpass, highpass, bandpass, bandstop, Hilbert, differentiator, pulse-shaping (including FIR Gaussian) and arbitrary magnitude. Advanced design methods and additional responses are available in Filter Design Toolbox™.

New dfilt method to specify filter coefficients at block ports

A new option has been added to the dfilt block method to specify filter coefficients via Simulink block ports.

R2008b

Version: 6.10

New Features

Bug Fixes

New Walsh–Hadamard Transform functions

A new fast Walsh–Hadamard transform fwht function and an inverse fast Walsh–Hadamard transform ifwht have been added to the toolbox. An associated demo has also been added.

R2008a

Version: 6.9

New Features

Bug Fixes

New Marcum Q Function

A new marcumq function, which implements the generalized Marcum Q function, has been added to the toolbox.

Conversion Between Magnitude and dB Added

The new utility functions mag2db and db2mag have been added for converting from magnitude to dB and dB to magnitude, respectively.

PMTM Function Enhanced with Ability to Keep or Drop Last Taper

You can now specify whether to keep or drop the last taper for calculating the pmtm, which returns the PSD using the Thomson multitaper method. By default, the last taper is dropped. If you set the DropLastTaper property false, the last taper is included.

R2007b

Version: 6.8

New Features

Bug Fixes

Confidence Interval Estimation Added

The spectrum object has been enhanced with a new method for calculating confidence intervals for PSDs and mean-squared spectra.

Spurious-Free Dynamic Range (SFDR) Measurement Added

The dspdata object has been enhanced with a new method to measure spurious-free dynamic range (SFDR) for mean-squared spectra.

Local Maxima/Peak Finder Added

A new function findpeaks has been added to identify local maxima in a data vector. You can specify the minimum peak height and distance from its neighbors to limit the results. A `findpeaks` method has also been added to the dspdata object.

Conversions Between Power and dB Added

The new utility functions pow2db and db2pow have been added for converting from power to dB and dB to power, respectively.

R2007a

Version: 6.7

New Features

Bug Fixes

lsf2poly and latcfilt Multi-Channel Input Support Added

lsf2poly and latcfilt now support N-D array input where each column represents a separate input channel.

Circular Convolution (cconv) Function Added

A new function (cconv) that computes circular convolution has been added to the toolbox.

Spectrum Objects Partial Frequency Range Input Support Added

spectrum objects now support computing the spectrum and pseudospectrum on a user-specified vector of frequencies. This vector identifies the frequencies at which the spectrum or pseudospectrum is calculated.

cceps Factorize Algorithm Information Clarified

A more detailed explanation of the factorize algorithm and an example have been added to the cceps reference page.

dfilt.statespace Now Supports realizemdl Method

You can now create a Simulink block from dfilt.statespace objects with the `dfilt realizemdl` method.

ellip and ellipap Functions Enhanced

Both ellip and ellipap have been enhanced so that they are able to handle filters with more stringent requirements.

R2006b

Version: 6.6

New Features

Bug Fixes

Compatibility Considerations

Frequency Vector Input Added to Spectral Analysis Functions

The spectral analysis command line functions (pburg, pcov, peig, periodogram, pmcov, pmtm, pmusic, pwelch, and pyulear) now accept a frequency vector as an input parameter. This vector identifies the frequencies at which the spectral analysis function returns an estimate.

For functions that use the Goertzel algorithm (periodogram, pmtm, pwelch, and spectrogram), the frequency inputs are rounded to match the nearest bin value used by the algorithm.

FFT Length in Spectral Analyses Changed

The `FFTLength` parameter has been removed from all spectrum objects and you now specify the number of FFT points (NFFT) via the `psd`, `msspectrum`, or `pseudospectrum` estimation method. The NFFT value can be an integer or a string (either `'Nextpow2'` or `'Auto'`). `'Nextpow2'` is the default and sets the number of FFT points to the next power of 2 greater than the input signal length (or the segment length for `spectrum.welch` objects). `'Auto'` sets the number of FFT points to be equal to the input signal or segment length.

Compatibility Considerations

You should update any existing code that specifies the `FFTLength` parameter and instead use the NFFT parameter associated with an estimation method.

You should also verify that any `spectrum.welch` objects are using the desired FFT length, since the FFT length is now based on the segment length instead of the input signal length.

sosfilt and dfilt filter Method Support Multidimensional Array Input

You can now input a multidimensional array to sosfilt and to the dfilt `filter` method.

dfilt block Method Supports Target Subsystem Destination and Link Between Command Line and Model

The dfilt `block` method now allows you to specify a target subsystem in your Simulink model where you want to place the block. Two new parameters implement this:

'Destination' and 'Link2Obj'. The 'Destination' specifies where to place the block and 'Link2Obj' creates a link between the block in your model and inputs from the command line.

gaussfir Algorithm Updated

The alpha parameter in the formula used in gaussfir has been updated to match the formula in Rappaport T.S., *Wireless Communications Principles and Practice*, 2nd Edition, Prentice Hall, 2001.

R2006a

Version: 6.5

New Features

Bug Fixes

Compatibility Considerations

Taylor Window Function Added

A new function taylorwin for generating Taylor windows has been added.

SPTool Filter Designer Replaced by FDATool

FDATool has replaced the SPTool Filter Designer as the preferred method for designing filters for use in SPTool. For details, see FDATool in the Signal Processing Toolbox documentation.

Compatibility Considerations

The format in which filters are saved differs between SPTool Filter Designer and FDATool. When you load an SPTool session with saved filters, you are prompted to upgrade your filters to use FDATool format.

sgolay Example Improved

The example for sgolay has been improved and expanded.

zp2sos zeroflag Parameter Added

A new parameter `zeroflag` has been added to zp2sos. This parameter is used for real zeros that are the negatives of each other. For these zeros it specifies whether to keep them together instead of ordering them according to proximity to poles.

Help for Objects Changed

To obtain help for objects, use `help object.constructor` instead of the old `help object/constructor`. Note that to obtain help for methods, you still use `help object/method`.

R14SP3

Version: 6.4

New Features

Bug Fixes

dfilt (Discrete-Time Filters) Delay Structure Added

A new delay structure (dfilt.delay) has been added to `dfilt` objects. This structure adds latency to any signal filtered with it.

WinTool/WVTool Normalize Magnitude Added

A `Normalize magnitude` option has been added to the **Analysis Parameters** of WinTool and WVTool magnitude plots.

FDATool/FVTool Plot Displays Improved

The default plots for filter responses have been improved. The *y*-axis autoscaling includes buffer regions around the data and shows only the significant data. To see all of the data without any buffer regions, select **Full view** from the **View** menu.

FVTool Passband Zoom Added

If you have a filter in FVTool that was created in FDATool or from a Filter Design Toolbox `fdesign` object, you can use **Passband** on the **View** menu to zoom the passband region.

R14SP2

Version: 6.3

New Features

Bug Fixes

Compatibility Considerations

FDATool and FVTool Changes

FDATool Spectral Rejection Masks Added

You can draw lines on your filter response in FDATool to indicate rejection areas.

FDATool Generated C Header File Complex Filter Support

FDATool now supports generating C header files for complex filters.

FDATool Tip of the Day Added

A new Tip of the Day dialog displays when you start FDATool. It contains tips and hints for using FDATool.

FDATool State Space Filters Support Removed

FDATool no longer supports state space filters.

Compatibility Considerations

If you load a saved FDATool session that contains a state space filter, it is converted to a direct-form II transposed filter.

FDATool/FVTool New Analysis Parameters Magnitude Response Options

Three new options have been added to the Analysis Parameters for magnitude response displays.

- **Normalize Magnitude to 1 (0 dB)** — displays the magnitude so that the maximum magnitude value occurs at 0 dB
- **Autoscale axes** — automatically scales the response data y-axis
- **dB Display Range** — If you are not using autoscale and the magnitude display is in dB, this allows you to specify the y-axis limits, .

FVTool SOS Filter Coefficients Display Enhancement

The coefficient view in FVTool now displays each section of a second-order section filter as a separate filter with its own numerator, denominator, and gain.

FVTool Default Phase Units Changed

The default units for the phase response in FVTool have been changed to radians. This is consistent with the phasez function.

Compatibility Considerations

You should verify that the FVTool phase plots generated by existing code display the desired units.

dfilt Changes

dfilt Coefficients Method Changed

The `dfilt coefficients` method has been changed to the `coeffs` method, which returns a structure. See the Methods section of dfilt for information.

Compatibility Considerations

You should update any code that uses the `coefficients` method to use the new `coeffs` method and its returned structure.

dfilt Filter States Changed to Use States Property

You cannot pass filter states (initial and final conditions) via the `dfilt filter` method. You must use the `states` property. See dfilt for more information.

Compatibility Considerations

You should update any code that passed filter states via the `dfilt filter` method to use the new `states` property.

Spectral Analysis Changes

spectrogram Function Replaces specgram

spectrogram has been added to replace the grandfathered `specgram` function. If you use this function with no outputs, a surface plot is displayed, instead of an image.

Compatibility Considerations

You should update any code that references `specgram` to use the new `spectrogram` function. `spectrogram` uses different default values than `specgram` and the order of the inputs has changed.

Spectral Analysis Functions Inputs Changed

pwelch (and the other spectrum analysis functions) no longer accept `'half'` or `'whole'`. You must use `'onesided'` or `'twosided'` to indicate the type of analysis you want.

Compatibility Considerations

You should update any code that uses `'half'` or `'whole'` with spectral analysis functions and instead use `'onesided'` or `'twosided'`, respectively.

PSD Objects and Function Output Plots Changed

The following functions and methods now generate standard MATLAB plots, instead of launching an interactive plot. Refer to the MATLAB documentation for information on plots.

- dspdata `plot` method
- spectrum `psd`, `pseudospectrum`, and `msspectrum` methods
- pburg
- pcov
- periodogram
- pmcov
- pmtm
- pwelch
- pyulear

Other Changes

gaussfir Function Replaces firgauss

gaussfir has been added to replace the grandfathered `firgauss` function. `gaussfir` uses parameters that are common to communications systems.

Compatibility Considerations

You should update any code that references `firgauss` to use the new `gaussfir`.

firpm and cfirpm Inputs Changed

The firpm and cfirpm functions now take function handles as inputs instead of strings.

Compatibility Considerations

You should update any code that uses `firpm` or `cfirpm` so that it will work correctly with function handle inputs instead of string input.

New Demos

Signal Processing Toolbox demos have been reorganized and a new demo on the analysis of a numerically controlled oscillator (NCO) has been added.

Filter Wizard Product Dependency Removed

The Filter Wizard no longer requires Filter Design Toolbox software. You can use the Filter Wizard if you have Signal Processing Toolbox software and Simulink installed. If you have the Filter Design Toolbox software installed, more options are available. See dspfwiz for more information.