Keyword	Type	Meaning
DATE	Number	The date/time when start was issued via CtrlFIFO.
DELTAT	Number	The FIFO's deltaT value as set by CtrlFIFO.
DISKTOT	Number	Current number of chunks written to the FIFO's file.
FILENUM	Number	The output file refNum or review file refNum as set by CtrlFIFO. This will be zero if the FIFO is connected to no file.
NOTE	String	The FIFO's note string as set by CtrlFIFO.
VALID	Number	Zero if FIFO is not valid.
DATATYPE	Number	Channel's data type as if set by NewFIFOCHAN/Y=(*numType*) where *numType* is a value as returned by the **WaveType** function.

In the following table, the channel's number is represented by "#".

Keyword	Type	Meaning
FSMINUS#	Number	Channel's minus full scale value as set by NewFIFOChan.
FSPLUS#	Number	Channel's plus full scale value as set by NewFIFOChan.
GAIN#	Number	Channel's gain value as set by NewFIFOChan.
NAME#	String	Name of channel.
OFFSET#	Number	Channel's offset value as set by NewFIFOChan.
UNITS#	String	Channel's units as set by NewFIFOChan.

See Also

The **NewFIFO**, **CtrlFIFO**, and **NewFIFOChan** operations, **FIFOs and Charts** on page IV-313 for more information on FIFOs and data acquisition.

The **NumberByKey** and **StringByKey** functions for parsing keyword-value strings.

FilterFIR

FilterFIR [***flags***] ***waveName*** [**,** ***waveName***]…

The FilterFIR operation convolves each *waveName* with automatically-designed filter coefficients or with *coefsWaveName* using time-domain methods.

The automatically-designed filter coefficients are simple lowpass and highpass window-based filters or a maximally-flat notch filter. Multiple filter designs are combined into a composite filter. The filter can be optionally placed into the first *waveName* or just used to filter the data in *waveName*.

FilterFIR filters data faster than **Convolve** when there are many fewer filter coefficient values than data points in *waveName*.

Note: FilterFIR replaces the obsolete **SmoothCustom** operation.

Parameters

waveName is a destination wave that is overwritten by the convolution of itself and the filter.

waveName may be multidimensional, but only one dimension selected by /DIM is filtered (for two-dimensional filtering, see **MatrixFilter**).

If *waveName* is complex, the real and imaginary parts are filtered independently.

Flags

/COEF [=*coefsWaveName*]

Replaces the first output *waveName* by the filter coefficients instead of the filtered results or, when *coefsWaveName* is specified, replaces the output wave(s) by the result of convolving *waveName* with coefficients in *coefsWaveName*.

coefsWaveName must not be one of the destination *waveName*s. It must be single- or double-precision numeric and one-dimensional.

To avoid shifting the output with respect to the input, *coefsWaveName* must have an odd length with the "center" coefficient in the middle of the wave.

The coefficients are usually symmetrical about the middle point, but FilterFIR does not enforce this.

/DIM=*d*

Specifies the wave dimension to filter.

d=-1: Treats entire wave as 1D (default).

d=0: Operates along rows.

d=1: Operates along columns.

d=2: Operates along layers.

d=3: Operates along chunks.

Use /DIM=0 to apply the filter to each individual column (each one a channel, say left and right) in a multidimensional *waveName* where each row comprises all of the sound samples at a particular time.

/E=*endEffect*

Determines how the ends of the wave (*w*) are handled when fabricating missing neighbor values. *endEffect* has values:

0: Bounce method (default). Uses w[*i*] in place of the missing w[-*i*] and w[*n*-*i*] in place of the missing w[*n*+*i*].

1: Wrap method. Uses w[*n*-*i*] in place of the missing w[-*i*] and vice versa.

2: Fill with 0. Same as /ENDV={0}.

3: Fill method. Uses w[0] in place of the missing w[-*i*] and w[*n*] in place of the missing w[*n*+*i*].

/ENDV={*sv* [, *ev*]}

When fabricating missing neighbor values for each filtered wave, missing values before the start of data are replaced with *sv*.

Missing values after the end of data are replaced with *ev* if specified, or with *sv* if *ev* is omitted. /ENDV implies /E=2.

/ENDV was added in Igor Pro 9.00.

/HI={*f1*, *f2*, *n*}

Creates a high-pass filter based on the windowing method, using the Hanning window unless another window is specified by /WINF.

f1 and *f2* are filter design frequencies measured in fractions of the sampling frequency, and may not exceed 0.5 (the normalized Nyquist frequency).

f1 is the end of the reject band, and *f2* is the start of the pass band:

$0 < f1 < f2 < 0.5$

n is the number of FIR filter coefficients to generate. A larger number gives better stop-band rejection. A good number to start with is 101.

Use both /HI and /LO to create a bandpass filter.

/LO={*f1*, *f2*, *n*}

Creates a low-pass filter. *f1* is the end of the pass band, *f2* is the start of the reject band, and *n* is the number of FIR filter coefficients. See /HI for more details.

/NMF={*fc*, *fw* [, *eps*, *nMult*]}

Creates a maximally-flat notch filter centered at *fc* with a -3dB width of *fw*. *fc* and *fw* are filter design frequencies measured in fractions of the sampling frequency, and may not exceed 0.5 (the normalized Nyquist frequency).

The longest filter length allowed is 400,001 points, which requires fw >= 0.000789 (0.0789% of the sampling frequency).

Prior to Igor Pro 8.03 the longest filter length was 4,001 points, with fw >= 0.00789 (0.789% of the sampling frequency).

Coefficients at the ends that are smaller than the optional *eps* parameter are removed, making the filter shorter (and faster), though less accurate. The default is 2^{-40}. Use 0 to retain all coefficients, no matter how small, even zero coefficients. Retaining all coefficients will substantially increase the execution time as *fw* is made smaller.

nMult specifies how much longer the filter may be to obtain the most accurate notch frequency. The default is 2 (potentially twice as many coefficients). Set *nMult* <= 1 to generate the shortest possible filter.

The maximally flat notch filter design is based on Zahradník and Vlcek, and uses arbitrary precision math (see **APMath**) to compute the coefficients.

/WINF=*windowKind*

Applies the named "window" to the filter coefficients. Windows alter the frequency response of the filter in obvious and subtle ways, enhancing the stop-band rejection or steepening the transition region between passed and rejected frequencies. They matter less when many filter coefficients are used.

If /WINF is not specified, the Hanning window is used. For no coefficient filtering, use /WINF=None.

Choices for *windowKind* are:

Bartlett, Blackman367, Blackman361, Blackman492, Blackman474, Cos1, Cos2, Cos3, Cos4, Hamming, Hanning, KaiserBessel20, KaiserBessel25, KaiserBessel30, Parzen, Poisson2, Poisson3, Poisson4, and Riemann.

See **FFT** for window equations and details.

Details

If *coefsWaveName* is specified, then /HI, /LO, and /NMF are ignored.

If more than one of /HI, /LO, and /NMF are specified, the filters are combined using linear convolution. The length of the combined filter is slightly less than the sum of the individual filter lengths.

A band pass or band reject filter results when both /LO and /HI are specified. A band pass filter results from /LO frequencies greater than the /HI frequencies (the pass bands of the low pass and high pass filters overlap). Beginning with Igor Pro 9.00, a band reject filter results when /LO frequencies are less than the /HI frequencies (the stop bands of the filters overlap).

The filtering convolution is performed in the time-domain. That is, the FFT is not employed to filter the data. For this reason the coefficients length should be small in comparison to the destination waves.

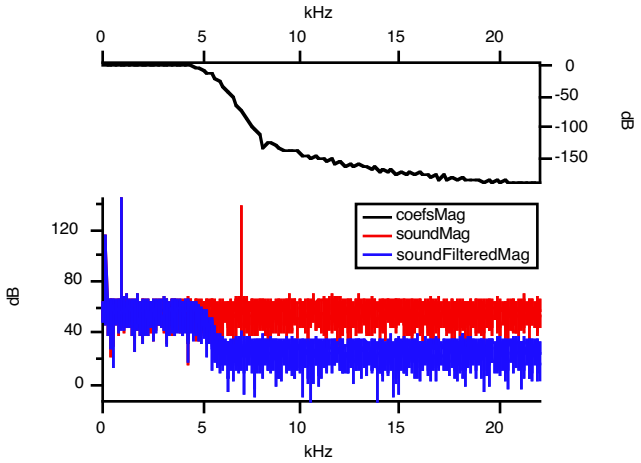
FilterFIR assumes that the middle point of *coefsWaveName* corresponds to the delay = 0 point. The "middle" point number = trunc(numpnts(*coefsWaveName* -1)/2). *coefsWaveName* usually contains the two-sided impulse response of a filter, and usually contains an odd number of points. This is the kind of coefficients data generated by /HI, /LO, and /NMF.

FilterFIR ignores the X scaling of all waves, except when /COEF creates a coefficients wave, which preserves the X scale deltax and alters the leftx value so that the zero-phase (center) coefficient is located at x=0.

Examples

```
// Make test sound from three sine waves
Variable/G fs= 44100                              // Sampling frequency
Variable/G seconds= 0.5                           // Duration
Variable/G n= 2*round(seconds*fs/2)
Make/O/W/N=(n) sound                              // 16-bit integer sound wave
SetScale/p x, 0, 1/fs, "s", sound
```

```
Variable/G f1= 200, f2= 1000, f3= 7000
Variable/G a1=100, a2=3000,a3=1500
sound= a1*sin(2*pi*f1*x)
sound += a2*sin(2*pi*f2*x)
sound += a3*sin(2*pi*f3*x)+gnoise(10)              // Add a noise floor

// Compute the sound's spectrum in dB
FFT/MAG/WINF=Hanning/DEST=soundMag sound
soundMag= 20*log(soundMag)
SetScale d, 0, 0, "dB", soundMag

// Apply a 5 kHz low-pass filter to the sound wave
Duplicate/O sound, soundFiltered
FilterFIR/E=3/LO={4000/fs, 6000/fs, 101} soundFiltered

// Compute the filtered sound's spectrum in dB
FFT/MAG/WINF=Hanning/DEST=soundFilteredMag soundFiltered
soundFilteredMag= 20*log(soundFilteredMag)
SetScale d, 0, 0, "dB", soundFilteredMag

// Compute the filter's frequency response in dB
Make/O/D/N=0 coefs                                 // Double precision is recommended
SetScale/p x, 0, 1/fs, "s", coefs
FilterFIR/COEF/LO={4000/fs, 6000/fs, 101} coefs
FFT/MAG/WINF=Hanning/PAD={(2*numpnts(coefs))}/DEST=coefsMag coefs
coefsMag= 20*log(coefsMag)
SetScale d, 0, 0, "dB", coefsMag

// Graph the frequency responses
Display/R/T coefsMag as "FIR Lowpass Example";DelayUpdate
AppendToGraph soundMag, soundFilteredMag;DelayUpdate
ModifyGraph axisEnab(left)={0,0.6}, axisEnab(right)={0.65,1}
ModifyGraph rgb(soundFilteredMag)=(0,0,65535), rgb(coefsMag)=(0,0,0)
Legend
```




```
// Graph the unfiltered and filtered sound time responses
Display/L=leftSound sound as "FIR Filtered Sound";DelayUpdate
AppendToGraph/L=leftFiltered soundFiltered;DelayUpdate
ModifyGraph axisEnab(leftSound)={0,0.45}, axisEnab(leftFiltered)={0.55,1}
ModifyGraph rgb(soundFiltered)=(0,0,65535)

// Listen to the sounds
PlaySound sound               // This has a very high frequency tone
PlaySound soundFiltered       // This doesn't
```

References

Zahradník, P., and M. Vlcek, Fast Analytical Design Algorithms for FIR Notch Filters, *IEEE Trans. on Circuits and Systems*, *51*, 608 - 623, 2004.

<http://euler.fd.cvut.cz/publikace/files/vlcek/notch.pdf>

See Also

Smoothing on page III-292; the **Smooth**, **Convolve**, **MatrixConvolve**, and **MatrixFilter** operations.