Smooth

Smooth [***flags***] ***num, wave0*** [**,** ***wave1***]…

The Smooth operation smooths the named waves using binomial (Gaussian) smoothing, boxcar (sliding average) smoothing, Savitzky-Golay (polynomial) smoothing, or running-median filtering.

Parameters

num is the number of smoothing operations to be applied for binomial smoothing or the integer number of points in the smoothing window for boxcar, Savitzky-Golay, and running-median smoothing, or, if you use /BLPF, the low-pass cutoff frequency.

Each wave, which may be floating point or integer, is smoothed in-place, overwriting the values with the smoothed result.

If an input wave is complex, the real and imaginary parts are smoothed independently.

If an input wave contains NaNs, smoothing results are unsatisfactory unless running-median (/M) filtering is used with a median of 3 or more. The **Loess** operation can fill in NaNs and the **Interpolate2** operation ignores NaNs and INFs.

Flags

/B [=*b*] Invokes boxcar smoothing algorithm. If given, *b* specifies the number of passes to use when smoothing the data with smoothing factor *num* (box width). The number of passes can be any value between 1 and 32767.

/BLPF[=*roundMode*] Calculates the integer number of smoothing operations to be applied for binomial smoothing based on interpreting the *num* parameter as the low-pass cutoff frequency and then performs the smoothing. /BLPF was added in Igor Pro 8.00.

The calculated number of smoothing operations is returned in V_iterations:

$$V\_iterations = \ln(\sqrt{2})\left(\frac{1}{(\pi f_{cn})^2} - \frac{1}{6}\right)$$

where f_{cn} = *num* * DimDelta(wave0, *dim*)

If *roundMode* is 1, V_iterations is rounded down to an integer value. This is the default if you use /BLPF by itself without *roundMode*.

If *roundMode* is 2, V_iterations is rounded up to an integer value.

f_{cn} is the Nyquist-normalized cutoff frequency and should be ≤ 0.182028, which results in V_iterations = 1.

The cutoff frequency, also known as the "half power point", is the frequency at which the magnitude response of the smoothing filter declines to 1/sqrt(2) ≈ 0.707107.

/DIM=*dim* Specifies the wave dimension to smooth.

dim=-1:	Treats entire wave as 1D (default)
dim=0:	Operates along rows
dim=1:	Operates along columns
dim=2:	Operates along layers
dim=3:	Operates along chunks

/E=*endEffect* Determines how to handle the ends of the wave (w) when fabricating missing neighbor values.

endEffect=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*].
endEffect=1:	Wrap method. Uses w[*n*-*i*] in place of the missing w[-*i*] and vice versa.
endEffect=2:	Zero method. Uses 0 for any missing value.
endEffect=3:	Repeat method. Uses w[0] in place of the missing w[-*i*] and w[*n*] in place of the missing w[*n*+*i*].

/EVEN [=*evenAllowed*]

Specifies the smoothing increment for boxcar smoothing (/B). Values are:

0:	Increments even values of *num* to the next odd value. Default when /EVEN omitted.
1:	Uses even values of *num* for boxcar smoothing despite the half-sample shifting this introduces in the smoothed output (prior to version 6, this shift was prevented). Same as/EVEN alone.

/F [=*f*] Selects the boxcar or multipass binomial smoothing method:

f=0:	Slow, but accurate, method (default).
f=1:	Fast method. Same as /F alone.

/M=*threshold* Invokes running-median smoothing and specifies an absolute numeric threshold used to optionally replace "outliers". Points that differ from the central median by an amount exceeding *threshold* are replaced, either with the *replacement* value specified by /R, or otherwise with the median value.

Special *threshold* values are:

0: Replace all values with running-median values or the *replacement* value.

(NaN): Replace only NaN input values with running-median values or the *replacement* value.

The smoothing factor *num* is the number of points in the smoothing window used to compute each median.

/MPCT=*percentile* Used with /M to compute a smoothed value that is a different percentile than the median. /M must be present if /MPCT is used.

percentile is a value from 0 to 100.

Roughly speaking, the smoothed value returned is the smallest value in the smoothing window that is greater than the smallest *percentile* % of the values. See "Median and Percentile Smoothing Details", below.

percentile=0: The smoothed value is the minimum value in the smoothing window.

percentile=50: The smoothed value is the median of the values in the smoothing window. This is the default if /MPCT is omitted.

percentile=100: The smoothed value is the maximum value in the smoothing window.

/R=*replacement* Specifies the value that replaces input values that exceed the central median by *threshold* (requires /M). *replacement* can be any value (including NaN or ±Inf if *waveName* is floating point).

/S=*sgOrder* Invokes Savitzky-Golay smoothing algorithm and specifies the smoothing order. *sgOrder* must be either 2 or 4.

Binomial Smoothing Details

For binomial smoothing, use no flags other than /BLPF, /DIM, /E, and /F.

If you are not using /BLPR, *num* is the number of smoothing operations between 1 and 32767.

If you are using /BLPF, *num* is the low-pass cutoff frequency which must be less than 0.182028 times the sampling frequency of *wave0*. The computed smoothing factor is returned in V_iterations. Very small values of f_{cn} (≈ 0.0007 or less) will fail with errors due to excessive number of smoothing operations or to insufficient range (too few wave points).

In Igor Pro 6 and later, the binomial smooth algorithm automatically switches to a nearly-equivalent but much faster multipass box smooth when *num* >= 50. You can prevent the switch to box smoothing by setting this global variable:

```
Variable/G root:V_doOrigBinomSmooth=1
```

The binomial smoothing algorithm does not detect and ignore NaNs in the input data. Use **Loess** or **Interpolate2** to fill in NaNs.

Boxcar Smoothing Details

For boxcar smoothing, use the /B flag and a *num* value from 1 to 32767.

For *num* < 2, no smoothing is done.

If *num* is even and /EVEN is not specified, *num* is incremented to the next (odd) integer.

If *num* is even and /EVEN is specified, each smoothed output is formed from one more previous value than future values.

The fast boxcar smoothing (/F) algorithm creates small errors when the data has a large offset. For some data sets you may want to subtract the mean of the data before smoothing and add it back in afterwards.

The boxcar smoothing algorithm detects and ignores NaNs in the input data. If *num* is less than the number of NaNs near the output point, then the result is NaN. Otherwise the average of the non-NaN neighboring points is used to compute the smoothed result.

Savitzky-Golay Smoothing Details

For Savitzky-Golay smoothing, use the /S flag and an odd *num* value from 5 to 25. An even value for *num* returns an error. If *sgOrder*=4, then *num*= 5 gives no smoothing at all so *num* should be at least 7.

The Savitzky-Golay smoothing algorithm does not detect and ignore NaNs in the input data.

Median and Percentile Smoothing Details

For running-median smoothing, use the /M flag and a *num* value from 1 to 32767. When *num* is 1, no smoothing is done.

If *num* is even, the median is the average of the two middle values.

For example, the median of 6 values around data[i] is the median of data[i-3], data[i-2], data[i-1], data[i], data[i+1], and data[i+2], and if these values were already sorted, the median would be the average of data[i-1] and data[i].

Use /M=0 to replace all values with the median over the smoothing window or use `/M=`*`threshold`*`/R=(NaN)` to replace outliers with NaNs.

Use `/M=(NaN)` to replace only NaN input values with the running-median values or the *replacement* value.

The running-median smoothing algorithm detects and ignores NaNs in the input data. If *num* is less than the number of NaNs near the output point, then the result is NaN. Otherwise the median of the non-NaN neighboring points is used to compute the smoothed result.

The running-median is a special case of running-percentile, with *percentile*=50.

The /M and /MPCT algorithm uses an interpolated rank to compute the value of percentiles other than 0 and 100.

Using Example 1 from <http://cnx.org/content/m10805/latest/> ("A Third Definition"), the 25th percentile (/MPCT=25) of the 8 values:

```
Make/O sortedData={3,5,7,8,9,11,13,15}// Already sorted, rank 1 to 8
```

The first step is to compute the rank (R) of the 25th percentile. This is done using the following formula: R=(*percentile*/100)*(*num*+1), where *percentile* is 25 and *num* is 8, so here R = 2.25.

If R were an integer, the Pth percentile would be the number with rank R; if R were 2 the result would be the 2nd value = 5.

Since R is not an integer, we compute the Pth percentile by interpolation as follows:

1. Define IR as the integer portion of R (the number to the left of the decimal point). For this example, IR=2.
2. Define FR as the fractional portion of R. For this example, FR=0.25
3. Find the values with Rank IR and with Rank IR+1. For this example, this means the values with Rank 2 and the score with Rank 3. The values are 5 and 7.
4. Interpolate by multiplying the difference between the values by FR and add the result to the lower values. For these data, this is 0.25(7-5)+5=5.5

Therefore, the 25th percentile is 5.5:

```
Smooth/M=0/MPCT=(percentile) 8, sortedData // 8-point smoothing window
Print sortedData[3] // prints 5.5, the 25th percentile of all 8 values
```

Smoothing Window and End Effects Details

These smoothing algorithms compute the output value for a given point using each point's neighbors. Except for running-median smoothing, each algorithm combines neighboring points before and after the point being smoothed. At the start or end of a wave some points will not have enough neighbors so some method for fabricating neighbor values must be implemented. The /E flag specifies the method.

The running-median filter, however, ignores /E. At each end of the data fewer values are included in the median calculation, so that values "beyond" the end of data are not needed.

The first output value is the median of `wave[0, floor((num-1)/2)]`. For example, if *num* = 7, then the first output value is the median of wave[0], wave[1], wave[2], and wave[3]. Because that is an even number of points, the median is the average of the two middle values. Continuing the example, if the values were 3, 1, 7, and 5, the two middle values are 3 and 5. The computed median would be (3+5)/2=4.

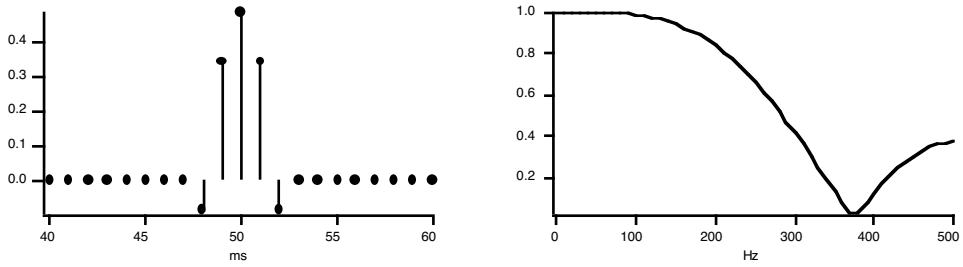
Examples

Box smoothing example:

```
Make/N=100 wv; Display wv
wv=gnoise(1)
Smooth/B/E=3 3,wv    // output[p] = average of wv[p-1], wv[p] and wv[p+1]
                     // /E=3 causes wv[0] = (w[0]+w[0]+w[1])/3
                     // and wv[n-1] = (w[n-2]+w[n-1]+w[n-1])/3
```

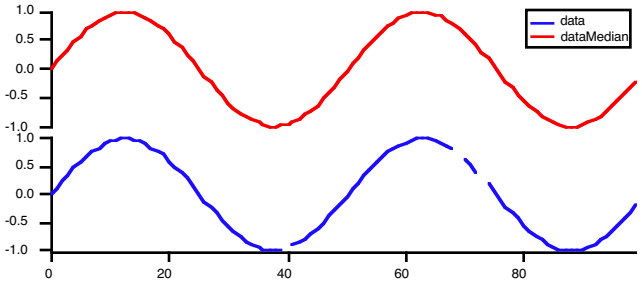
Demonstrate the impulse response of Savitzky-Golay Smoothing:

```
Make/O/N=100 wv
wv= p==50     // 1 at center of wave, 0 elsewhere; an impulse
SetScale/P x, 0, 1/1000, "s", wv           // 1000 Hz sampling rate
Smooth/S=2 5,wv
Display wv
ModifyGraph mode=8,marker=19
FFT/MAG/DEST=fftMag wv
Display fftMag
```



Replace NaN with median:

```
Make/O/N=100 data= enoise(1)>.9 ? NaN : sin(x/8)      // signal with NaNs
Duplicate/O data, dataMedian
Smooth/M=(NaN) 5, dataMedian      // replace (only) NaNs with 5-point median
```



Binomial Smoothing References

Marchand, P., and L. Marmet, *Revues of Scientific Instrumentation 54*, 1034, 1983.

Savitzky-Golay Smoothing References

Savitzky, A., and M.J.E. Golay, *Analytical Chemistry*, *36*, 1627-1639, 1964.

Steiner, J., Y. Termonia, and J. Deltour, *Analytical Chemistry*, *44*, 1906-1909, 1972.

Madden, H., *Analytical Chemistry*, *50*, 1386-1386, 1978.

Percentile References

<http://en.wikipedia.org/wiki/Percentile>

<http://cnx.org/content/m10805/latest/>