PRACTICAL GUIDE TO DATA SMOOTHING AND FILTERING

Ton van den Bogert
October 31, 1996

Summary:
This guide presents an overview of filtering methods and the software which is available in the HPL.

1. What is filtering/smoothing?

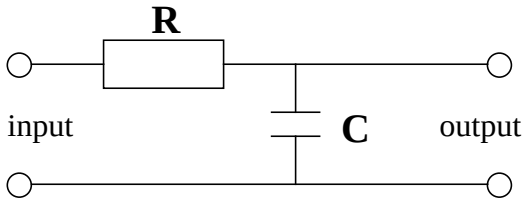
Smoothing is an operation which removes high-frequency fluctuations from a signal. Low-pass filtering is another term for the same thing, but is restricted to methods which are linear: i.e. if you want to filter a signal x(t)+y(t), it does not matter whether you apply the filter before or after adding the two signals. Such linear operations can be described by a frequency response. All methods described here are linear, with the exception of curve fitting.

2. Why (low-pass) filtering?

Random noise can greatly affect the result of certain types of analysis. Two important examples are: quantification of peaks in the signal (noise can shift the location of the peak, and leads to systematic overestimation of the height of the peak), and differentiation (noise is amplified by differentiation). Always consider filtering before these types of data analysis. Since random (or 'white') noise is distributed over all frequencies, and signal is typically limited to low frequencies, a reduction of high frequency components will improve the signal/noise ratio.

3. Methods and software

3.1. Analog (RC, resistor-capacitor) filter



The frequency response is a 1st order Butterworthfilter, and really easy to make (<1 Stano-hour), cut-off frequency $f_0 = \frac{1}{2\pi RC}$. Connect this between your signal source (transducer or amplifier) and the A/D converter if you want to get rid of high frequencies before the signal is sampled. Remember the sampling theorem: if the signal contains frequencies which are higher than half the sampling frequency, you can get erroneous results. An analog filter is the only way to make sure that high frequencies don't reach the A/D converter.

The frequency response (defined as the ratio between input and output amplitude for a sine wave of frequency f) of an nth order Butterworth filter is:

$$\frac{A_{\text{out}}}{A_{in}} \equiv H(f) = \frac{1}{\sqrt{1 + \left(\frac{f}{f_0}\right)^{2n}}}$$

H(f)

f_0=50 Hz

n=1

n=2

n=3

f [Hz}

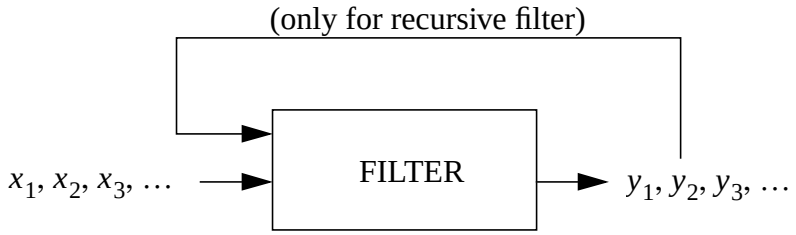
Higher order analog filters are much harder to make (> 1 Stano-day).

3.2. Digital filters

Digital filters are implemented as a computer program that transforms sampled data. E.g. 2nd order recursive filter:

$$y_i = ax_i + bx_{i-1} + cx_{i-2} + dy_{i-1} + ey_{i-2}$$

The term ‘recursive’ means that past y-values are fed back to the input. The relationship between input samples ($x_1 \ldots x_n$) and output samples ($y_1 \ldots y_n$) is illustrated in the following diagram:



KinTrak:
Uses a double recursive 2nd order Butterworth filter. This is a 4th order filter (since the asymptotic response is f^{-4}), but NOT a 4th order *Butterworth* filter because its frequency response is the square of the 2nd order filter:

$$H(f) = \frac{1}{1 + \left(\frac{f}{f_0}\right)^4}$$

Matlab functions:
BUTTER: get coefficients a,b,... for a Butterworth filter with specific order and cut-off frequency
FILTER: filter data
FILTFILT: filter data twice, forward and reverse

Notes:
1. Recursive digital filters always have a time lag between input and output, non-recursive filters (where y is not fed back to the input side) can be made with zero-lag but do not allow a good approximation to a Butterworth filter. Solution for the time lag problem: filter twice, second time in reverse direction.
2. Digital filters require constant sampling interval.
3. It is not (yet) clear to me how the Matlab implementation deals with boundary conditions.

3.3. Curve fitting (or non-linear regression)

Matlab functions:
POLYFIT to get coefficients of best-fitting polynomial
POLYVAL to evaluate polynomial at arbitrary time values
POLYDER to evaluate first derivative at arbitraty time values
LEASTSQ('model', parameters) for other nonlinear regression problems

Other software:
spreadsheet (Lotus, Excel), statistics (SPSS, SAS).

Applications: e.g. quantification of force-deformation curves.

Notes:
1. Apply only when underlying function is simple; high order polynomials are dangerous when extrapolated and give large errors in derivatives at the endpoints.
2. Does NOT correspond to a low-pass filter! (frequency response depends on data)

3.4. Smoothing splines

A nth degree spline function $f(t)$ is a piecewise nth degree polynomial function. The polynomials are joined at the 'nodes' (t-values in the input data) in such a way that all derivatives up to the $(n-1)$ th are continuous. Within these constraints, the function $f(t)$ is selected which minimizes:

$$\sum (f(t_i) - x_i)^2 + p \cdot \int \left(f^{\left(\frac{n+1}{2}\right)}(t) \right)^2 dt$$

where (t_i, x_i) are the raw data samples, and $f^{(k)}$ denotes the kth derivative of $f(t)$. The weight factor p is the smoothing parameter. With $p = 0$, an interpolating spline will be obtained, with $p = \infty$ a least squares fit of the entire dataset using a single polynomial of degree $(n-1)/2$ (e.g. a straight line for n=3. Intermediate values give a compromise between good fit (the first term in the equation) and the smoothness (the second term). The degree n should be odd: n = 3,5,7 for cubic, quintic, heptic splines.

Matlab function:
CSAPS: 3rd degree (cubic) smoothing spline

Stand-alone program:
GCV: any odd order spline, with many options to select smoothing parameter. See documentation (FrameMaker file) ~bogert/help/gcv.fm.

Fortran library:
GCVSPL: this library by Herman Woltring is used in the GCV program and can also be used in your own Fortran programs. You can interface it with Matlab too. See http://www.kin.ucalgary.ca/isb/software.

Notes:
1. Sampling interval does not have to be constant for a spline method.
2. Differentiation is not an additional finite difference approximation, as in the digital filter, but can be done directly from the equations of the polynomials. The GCVSPL package includes a function SPLDER (spline derivative) that does this for you.
3. Boundary conditions: $f^{\left(\frac{n+1}{2}\right)} = 0$, so use quintic spline if you need acceleration.

3.5. Fourier analysis

Matlab:

FFT (fast fourier transform), followed by reduction of amplitudes of high frequencies, followed by IFFT (inverse fft)

Notes:
1. Simply setting high frequencies to zero can cause ‘ringing’ near sudden transitions in the input signal.
2. Boundary conditions: $f^{(n)}(0) = f^{(n)}(T)$ for function ($n = 0$) and all derivatives ($n = 1, 2, \ldots$). Real movement data (especially non-cyclic data) does not satisfy these conditions.

Both drawbacks are to some extent avoided in the method published by Hatze (1981).

4. Choosing the ‘optimal’ filter

This is typically an interactive trial and error process. The goodness of a filter is best based on visual inspection of the results. Also note that it depends on the subsequent steps in the analysis too. For instance, a good filter for peak detection may be one which reduces noise to 1% of the signal. The same filter may not be good for differentiation, because the noise in the derivative is still to high. Filters for differentiation typically need a lower cut-off frequency.

The GCV software includes two optimization methods to select a p-value objectively. The first requires an estimate of the magnitude of the noise, the second (the generalized cross-validation criterion) tries to make an estimate of the noise. The latter method tends not to work well for choosing an optimal filter for differentiation (it tends to smooth not enough), but your mileage may vary.

5. References

Hatze, H. (1981) The use of optimally regularized Fourier series for estimating higher-order derivatives of noisy biomechanical data. J. Biomech. 14,13-18. (describes a Fourier series method)

Pezzack, J.C., R.W. Norman, and D.A. Winter (1977) An assessment of derivative determining techniques used for motion analysis. J. Biomechanics 10,377-382. (describes digital filtering method and benchmark data)

Woltring, H.J., “Smoothing and differentiation techniques applied to 3-D data,” in Three-Dimensional Analysis of Human Movement, ed. P. Allard, I.A.F. Stokes, J.-P. Blanchi, Human Kinetics Publishers, Urbana-Champaign, Illinois/USA. (good overview and mathematical background)

Woltring, H.J. (1986) A Fortran package for generalized, cross-validatory spline smoothing and differentiation. Adv. Engng. Softw. 8,104-110. (describes spline method, very technical)

Wood, G.A. (1982) Data smoothing and differentiaion procedures in biomechanics. Exerc. Sport Sciences Rev. 10,308-362. (good overview)

NOTE: if you suggestions for additions or revisions to this document, please contact me at bogert@acs.ucalgary.ca