7.0 Filtering of Time Series

7.1 Introduction

In this section we will consider the problem of filtering time or space series so that certain frequencies or wavenumbers are removed and some are retained. This is an oft-used and oft-abused method of accentuating certain frequencies and removing others. The technique can be used to isolate frequencies that are of physical interest from those that are not. It can be used to remove high frequency noise or low frequency trends from time series and leave unaltered the frequencies of interest. These applications are called low-pass and high-pass filtering, respectively. A band-pass filter will remove both high frequencies and low frequencies and leave only frequencies in a band in the middle. Band-pass filters tend to make even noise look periodic, or at least quasi-periodic. We will begin by noting a few important theorems that constitute the fundamental tools of non-recursive filtering.

The Convolution Theorem:

If two functions $f_1(t)$ and $f_2(t)$ have Fourier transforms $F_1(\omega)$ and $F_2(\omega)$ then the Fourier transform of $f_1(t) \cdot f_2(t)$ is

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} F_1(\lambda)F_2(\omega-\lambda)d\lambda$$

And the Fourier transform of

$$\int_{-\infty}^{\infty} f_1(\tau)f_2(t-\tau)d\tau$$

is $F_1(\omega) \cdot F_2(\omega)$. This latter result is the most useful in filtering, since it says that the Fourier transform of the convolution of two functions in time is just the product of the Fourier transforms of the individual functions.

Parseval's Theorem:

$$\int_{-\infty}^{\infty} f_1(t)f_2(t)dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} F_2(\omega)F_1(\omega)^* d\omega$$

for $f_1(t) = f_2(t) = f(t)$.

$$\int_{-\infty}^{\infty} f(t)^2 \, dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left| F(\omega) \right|^2 d\omega$$

Here we see that the variance of a time series integrated over all time is equal to the power spectrum integrated over all frequency.

7.2 Filtering

Suppose we wish to modify oscillations of certain frequencies in a time series while keeping other frequencies the same. *e.g.* remove high frequency oscillations (low-pass filter), remove low frequencies (high-pass filter), or both (band-pass filter). The ozone layer of Earth's atmosphere is a low-pass filter for sunlight in the sense that it absorbs all energy with wavelengths shorter than 300 nm before it reaches the surface. A couple different approaches to filtering can be taken.

7.2.1 Fourier Method

Fourier analyzed time series to compute amplitudes at all frequencies. Modify these amplitudes as desired, then reconstitute the series.

$$f(t) = \sum_i C_{\omega_i} \cos\left(\omega_i t - \phi_i\right) \qquad (7.1)$$

$$f_{filtered}(t) = \sum_{i=1}^{N} C_{\omega_i} \bullet R(\omega) \bullet \cos(\omega_i t - \phi_i) \qquad (7.2)$$

Here the function $R(w)$ is the response function of the desired filtering process and measures the ratio of the amplitude of the filtered to the unfiltered time series as a function of frequency.

$$R(\omega) = \frac{C_{\omega \cdot \text{filtered}}}{C_{\omega \cdot \text{original}}}$$

The problem with this method is that the reconstructed time series may not resemble the original one, particularly near the ends. This is the same general characteristic of functional fits discussed in an earlier chapter. Also, you need the whole record of data before you can produce a single filtered data point, and the most recently acquired values are at the end of the data stream, where the problems with the Fourier method are worst. So if you want to do real-time filtering, Fourier methods are hopeless. For real-time problems the recursive methods are quite good.

7.2.2 Centered, Non-recursive Weighting Method

In the centered non-recursive weighting method, the time series is subjected to a weighted running average, so that the filtered point is a weighted sum of surrounding points.

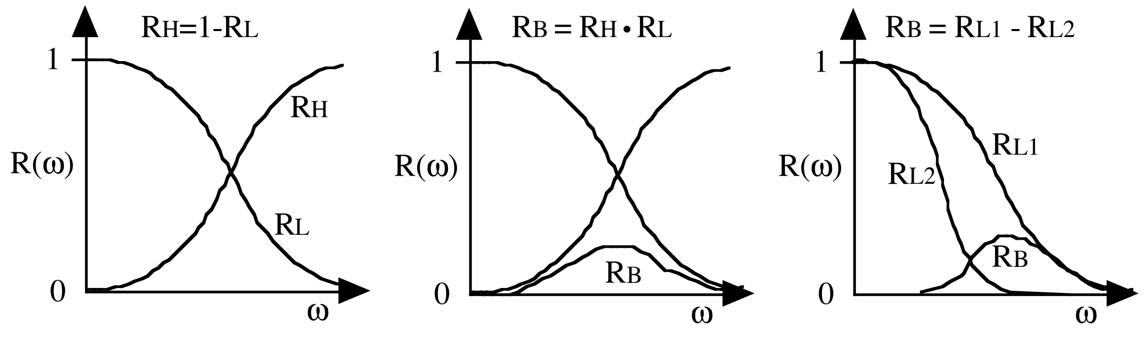
$$f_{\text{filtered}}(t) = \sum_{k=-J}^{J} w_k f(t + k\Delta t) \tag{7.3}$$

Some data points will be lost from each end of the time series since we do not have the values to compute the smoothed series for $i < J$ and $i > N\text{-}J$.

It seems obvious that such an operation can most reasonably produce only smoothed time series and hence constitutes a low-pass filter. However, a high-pass filter can be constructed quite simply by subtracting the low-pass filtered time series from the original time series. The new high-pass response function will then be

$$R_H(\omega) = 1 - R_L(\omega) \tag{7.4}$$

Where the subscripts H and L refer to high- and low-pass filters. One can then design a high-pass filter by first designing a low-pass filter that removes just those frequencies one wishes to retain. You can also make a band-pass filter by applying a low pass filter to a time series that has already been high-passed (or vice versa), in which case the response function is the product of the two response functions (center case below). Or you can subtract a low pass filtered version of the data set from another one with a cutoff at a higher frequency, as illustrated below on the right.



7.3 The Response Function

The response function is the spectrum of amplitude modifications made to all frequencies by the filtering function.

$$\left|R(\omega)\right| \equiv \frac{\text{amplitude of output series at frequency } \omega}{\text{amplitude of input series at frequency } \omega}$$

$$\left|R^2(\omega)\right| \equiv \frac{\Phi(\omega)}{\Phi(\omega)} \equiv \frac{\text{output power at } \omega}{\text{input power at } \omega}$$

$R(\omega)$ can be real or imaginary. Some filtering is done by nature and instruments and these may introduce phase errors. Any filter we would design and apply would have a real response function, unless we desire to introduce a phase shift. Phase shifting filters are not too commonly used in meteorological or oceanographic data analysis or modeling, and so we will not discuss them except in the context of recursive filtering, where a phase shift is often introduced with a single pass of a recursive filter.

How do we design a weighting with the desired frequency response? Our smoothing operation can be written

$$g(t) = \sum_{k=-J}^{J} f(t + k\Delta t)\, w(k\Delta t) \tag{7.5}$$

where $g(t)$ is the smoothed time series, $f(t)$ is the original time series and $w(k\Delta t)$ is the weighting. In the continuous case we can write this as

$$g(t) = \int_{-\infty}^{\infty} f(\tau)\, w(t-\tau)\, d\tau \tag{7.6}$$

The filtered output $g(t)$ is just the convolution of the unfiltered input series $f(t)$ and the filter weighting function $w(t)$. From the convolution theorem the Fourier transform of

$$\int_{-\infty}^{\infty} f(\tau)\cdot w(t-\tau)\, d\tau \quad \text{is} \quad F(\omega)\cdot W(\omega)$$

$$\text{so that} \qquad G(\omega) = F(\omega)\cdot W(\omega)$$

i.e. to obtain the frequency spectrum or Fourier coefficients of the output we multiply the Fourier transform of the input times the Fourier transform of the weighting function.

$$P_g(\omega) = G(\omega)G^*(\omega) = F(\omega)W(\omega)\cdot\left(F(\omega)W(\omega)\right)^*$$

$$= F(\omega)F^*(\omega)\cdot W(\omega)W^*(\omega)$$

$$= \left|F(\omega)\right|^2 \cdot \left|W(\omega)\right|^2$$

Simple Example:

Suppose our input time series consists of a single cosine wave of amplitude 1.

The output signal is then

$$g(t) = \sum_{k=1}^{K} w_k \cos\left(\omega(t + k\Delta t)\right)$$

And the Fourier Transform

$$G(\omega) = 2\int_0^{\infty} g(t)\cos\omega t\, dt$$

is

$$G(\omega) = 2\int \sum_{\kappa} w_k \cos(\omega t + \omega k\Delta t)\cos\omega t\, dt$$

$$= \sum_k w_k \cos\omega k\Delta t$$

$$= \sum_k w_k \cos\omega t_k$$

Now $F(\omega) = 1$ so that

$$R(\omega) = \frac{G(\omega)}{F(\omega)} = \sum_{k=-\infty}^{+\infty} w_k \cos\omega t_k = W(\omega) \text{ as } k \to \infty.$$

The Response function *R(ω)* is just the Fourier transform of the weighting function *w(t)*. If we assume that the response function and the weighting function are symmetric,

$$R(\omega) = W(\omega) = 2\int_0^\infty w(t)\cos\omega\tau\, d\tau \qquad \tau = k\Delta t$$

and conversely the weighting function can be obtained from a specified (desired) Response function

$$w(\tau) = 2\int_0^\infty R(\omega)\cos\omega\tau\, d\omega$$

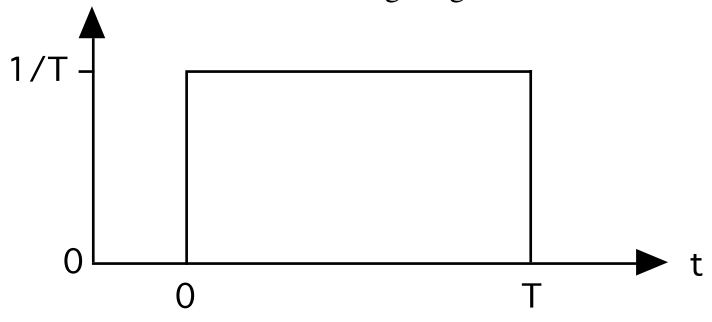
$\therefore$ *w(τ)* and *R(ω)*constitute a Fourier Transform Pair.

Some examples:

A Bad Example: The rectangular "Boxcar" weighting function or "running mean" smoother.

$$w(\tau) = \frac{1}{T} \text{ on the interval } 0 < \tau < T$$

Boxcar Weighting Function

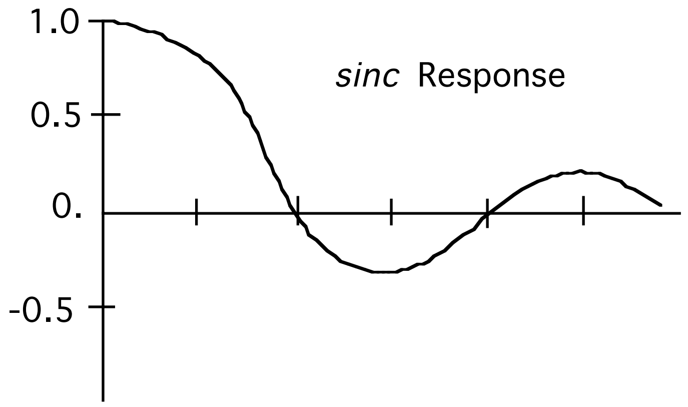


As we recall, the Fourier transform of the boxcar is the *sinc* function

$$R(\omega) = \frac{\sin\left(\frac{\omega T}{2}\right)}{\frac{\omega T}{2}}$$

This response function approaches one as $\omega T/2$ approaches zero. Hence it has no effect on frequencies with periods that are very long compared to the averaging interval *T*.

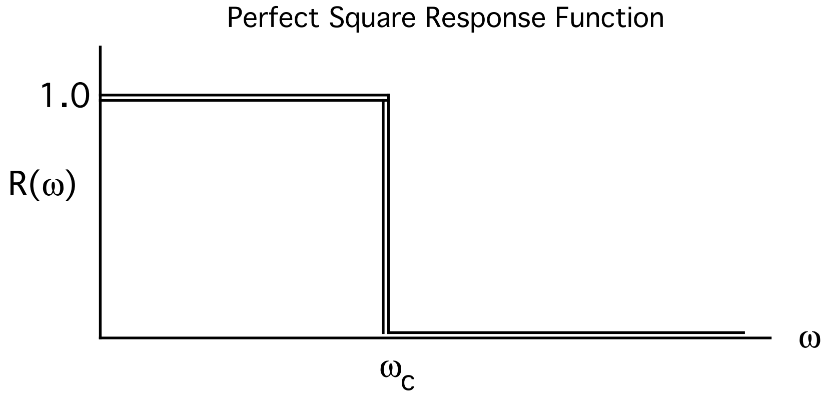
When $\omega T = 2n\,\pi$, n=1,2,3..., $R(\omega) \rightarrow 0$. The running mean smoother exactly removes wavelengths of which the length of the running mean is an exact multiple. For example, a 12 month running mean removes the annual cycle and all its higher harmonics.



Note that the response function, $R(\omega)$, of the running mean smoother is negative in the intervals 2π, -4π, 6π, -8π, etc. This means that frequencies in these intervals will be phase shifted by 180° because of the negative side lobes and the slow, $1/(\omega T)$, drop off of the amplitude of these side lobes the "running mean" filter is not very good. It may be useful if we are sure that the variance for $\omega T > \pi$ is small.

7.4 Inverse Problem

Suppose we wish to design a filter with a very sharp cutoff



From
$$w(\tau) = 2\int_0^\infty R(\omega)\cos\omega\tau\, d\omega$$

we get, $$w(\tau) = \frac{\sin\left(\frac{\omega_c \tau}{2}\right)}{\frac{\omega_c \tau}{2}}$$

This is a damped sine wave (sinc function) again.

The first zero crossing occurs at

$$\frac{\omega_c \tau}{2} = \pi \Rightarrow \tau = \frac{2\pi}{\omega_c} = \frac{1}{f_c}$$

The fact that the weight extends across several of these periods before dropping to zero causes severe problems at the beginning and end of a finite time series. In practice it is desirable to settle for a less sharp cutoff of $R(\omega)$ for which more practical weighting functions are available.

"Practical" Filters

(1) "Gaussian bell"

The function $$e^{-x^2}$$

forms a Fourier Transform pair with itself. Hence a Gaussian bell weighting function produces a Gaussian bell frequency response. Although, of course, when you consider the response function the bell is peaked at zero frequency and we usually ignore the negative frequencies.

(2) The Triangular One-Two-One filtering function

$$g(t) = \frac{1}{4} f(t-\tau) + \frac{1}{2} f(t) + \frac{1}{4} f(t+\tau)$$

$$R(\omega) = \cos^{2v}\left(\frac{\omega}{2}\Delta t\right)$$

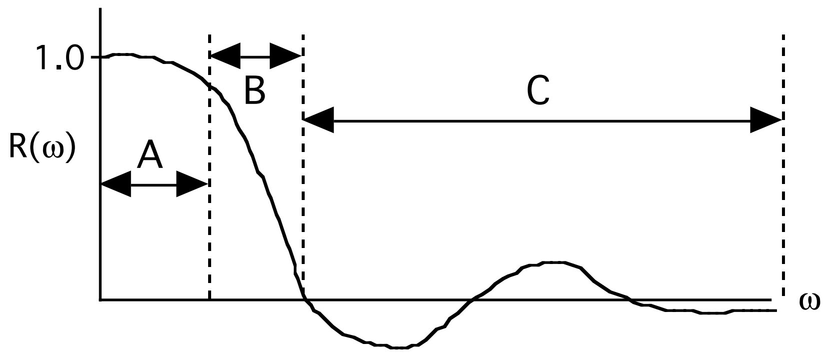
where v is the number of times the filter is applied.

The response function for the 1-2-1 smoother approaches a Gaussian bell shape for $v > 5$, $R(\omega) > 0$.

(3) Compromise Square Response:

You can construct a filter which (1) has relatively few weights, (2) Has a "sort of" square response and (3) has fairly small negative side lobes.

Compromise Response Function



The larger the ratio *A*/*B* the larger side lobes one must accept. Some weights are negative.
You can find a number of these in the literature and in numerous software packages. In the next section we will show how to construct centered, non-recursive filter weights to order.

7.5 Construction of Symmetric Nonrecursive Filters

7.5.1 Fourier Construction of filter weights

In this section we will describe methods for the construction of simple non-recursive filters. Suppose we consider a simple symmetric non-recursive filter

$$y_n = \sum_{k=-N}^{N} C_k x_{n-k} \quad \text{where} \quad C_k = C_{-k} \tag{7.7}$$

Time-Shifting Theorem:

To perform a Fourier transform of (7.7), it is useful to first consider the time-shifting theorem. Suppose we wish to calculate the Fourier transform of a time series *f(t)*, which has been shifted by a time interval *Δt=a*. Begin by substituting into the Fourier integral representation of *f(t)*.

$$f(t \pm a) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\, e^{i\omega(t \pm a)}\, d\omega$$

$$f(t \pm a) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{\pm i\omega a} F(\omega)\ e^{i\omega t}\, d\omega \tag{7.8}$$

From (7.10) we infer that the Fourier transform of a time series shifted by a time interval is equal to the Fourier transform of the unshifted time series multiplied by the factor,

$$z = e^{i\omega \Delta t} \tag{7.9}$$

We can Fourier transform (7.7) and use the time shifting theorem (7.8) to obtain

$$Y(\omega) = \left[\sum_{k=-N}^{N} C_k e^{-i\omega k \Delta t}\right] X(\omega) \tag{7.10}$$

Where $Y(\omega)$ and $X(\omega)$ are the Fourier transforms of $y(t)$ and x(t). Because $C_k = C_{-k}$ and

$$\cos x = \frac{e^{ix} + e^{-ix}}{2} \tag{7.11}$$

we can write (7.10) as

$$R(\omega) = \frac{Y(\omega)}{X(\omega)} = C_0 + 2\sum_{k=1}^{N} C_k \cos(\omega k \Delta t) \qquad (7.12)$$

We can find the weights that would give a desired response function by transforming (7.12) as follows. Multiply both sides of (7.12) by

$$\cos(j\omega\Delta t) \qquad\qquad j = 0,1,\ldots,N$$

and then integrate frequency, ω, over the Nyquist interval 0 — $\pi/\Delta t$

$$\int_0^{\pi/\Delta t} \cos(j\omega\Delta t) H(\omega)\, d\omega = 2C_j \int_0^{\pi/\Delta t} \cos(j\omega\Delta t)\cos(k\omega\Delta t)\, d\omega \qquad (7.13)$$

(7.13) becomes

$$C_k = \frac{1}{\pi}\int_0^{\pi} \cos(k\omega') H(\omega')\, d\omega' \qquad (7.14)$$

$\omega' = \Delta t \omega$, so that $0 < \omega' < \pi$ is the Nyquist interval. From (7.14) we can derive the appropriate weighting coefficients from any arbitrary desired response $H(\omega)$.

7.5.2 Computation of Response function for Symmetric Non-Recursive Weights

If we have a set of symmetric non-recursive weights, we can compute the response function easily using (7.12). Let's do a few examples:

The running mean smoother:

The running mean smoother replaces the central value on an interval with the average of the values surrounding that point. The running mean can be taken over an arbitrary number of points, e.g. 2, 3, 5, 7. Starting with (7.12) again,

$$R(\omega) = C_0 + 2\sum_{k=1}^{N} C_k \cos(\omega k \Delta t) \qquad (7.15)$$

a running mean smoother has C_k = 1/(2N+1), where –N < k < N. The length of the running mean smoother is 2N+1.

2N+1=3 $$R(\omega) = \frac{1}{3} + \frac{2}{3}\cos(\omega\Delta t) \qquad 0 < \omega < \frac{\pi}{\Delta t}$$

2N+1=5 $$R(\omega) = \frac{1}{5} + \frac{2}{5}\cos(\omega\Delta t) + \frac{2}{5}\cos(2\omega\Delta t) \qquad 0 < \omega < \frac{\pi}{\Delta t}$$

2N+1=7 $$R(\omega) = \frac{1}{7} + \frac{2}{7}\cos(\omega\Delta t) + \frac{2}{7}\cos(2\omega\Delta t) + \frac{2}{7}\cos(3\omega\Delta t)$$

These square weighting functions give damped sine wave response functions, which are generally undesirable. A slightly tapered weighting function, such as the 1-2-1 filter gives a much nicer response function.

1-2-1 Filter $$R(\omega) = \frac{1}{2} + \frac{1}{2}\cos(\omega\Delta t) \qquad 0 < \omega < \frac{\pi}{\Delta t}$$

We have to alter (7.15) a bit to compute the response function for a 1-1 Filter, a running mean that just averages adjacent values. The result is:

1 – 1 Filter $$R(\omega) = \cos\left(\frac{1}{2}\omega\Delta t\right) \qquad 0 < \omega < \frac{\pi}{\Delta t}$$

All these results are plotted in Figure 7.5.1. Note how the 1-2-1 filter cuts off more sharply than the 1-1 filter (running mean 2), but does not have the ugly negative side lobe of the 1-1-1 filter (running mean 3).

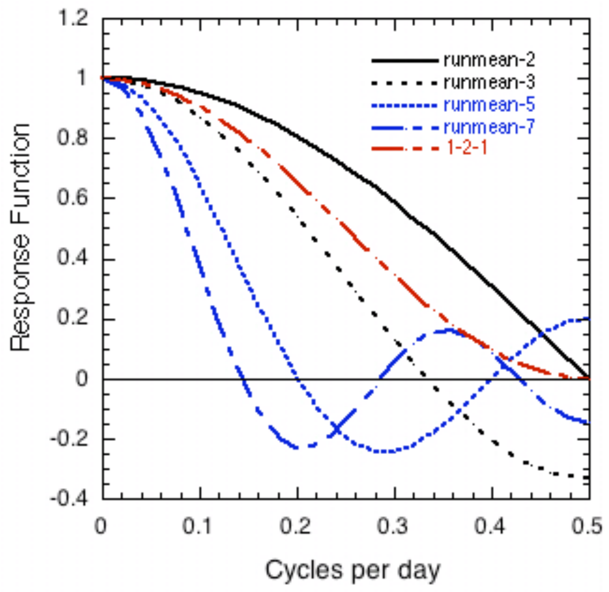


Figure 7.5.1. Response functions for running mean filters of length 2,3,5, and 7, plus the response function of the 1-2-1 filter. Note the nasty negative side lobes of the running mean filters, which have sinch functions shapes. The frequency interval runs from zero to the Nyquist frequency, which is 0.5 cycles per time step, or π radians per time step.

7.5.3 Computation of General Low-Pass Symmetric Non-Recursive Filter

Suppose we wish to derive the coefficients of a filter whose response cuts off sharply at a frequency $\alpha\pi$, where $0.0 < \alpha < 1.0$, as follows.

$$H(\omega) = \begin{cases} 1 & \omega < \alpha\pi \\ 0 & \omega > \alpha\pi \end{cases} \tag{7.16}$$

Then from (7.14) and (7.16) we have

$$C_k = \frac{1}{\pi}\int_0^{\alpha\pi} \cos(k\omega)d\omega \tag{7.17}$$

$$C_k = \frac{1}{k\pi}\sin(\alpha k\pi) \tag{7.18}$$

Note that the amplitude of the coefficients drops off as k^{-1}, which is rather slow. The coefficients, or weights, C_k, are a sinc function in k, as shown previously in Section 7.4. To get a really sharp cutoff we need to use a large number of weights. Usually we want to keep the number of points to a minimum, because we lose N-1 data off each end of the time series and because the computations take time. The computation time problem can be alleviated with the use of recursive filters.

If we truncate (7.18) at some arbitrary value of N, then the response function will be less sharp than we would like and will have wiggles associated with Gibb's phenomenon. This is shown in Figure 7.4.1 which shows the response function (7.12) for the weights (7.18) for truncation of N=10, N=20, and N=50. The value of $\alpha = 0.5$ was chosen to cut the Nyquist interval in the center.

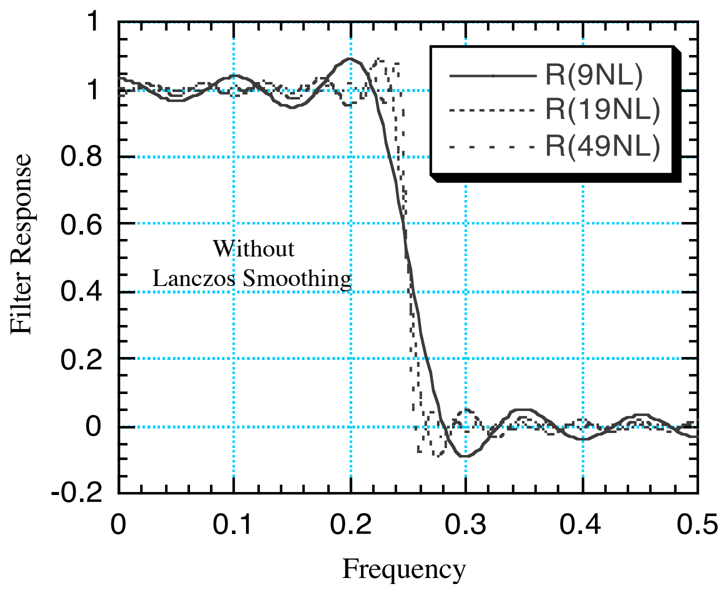


Figure 7.5.2: Response function for the filter weights given by (7.15) for three values of *N*; *N*=9, *N*=19 and *N*=49. α=0.5, so that the cutoff is in the middle of the Nyquist interval, 0 < *f* < *0.5.*, at 0.25.

Lanczos Smoothing of Filter Weights:

The wiggles in the response functions of Figure 7.5.2 have a wavelength of approximately the last included or the first excluded harmonic of (7.12). We can remove this harmonic by smoothing the response function. The running mean smoother exactly removes oscillations with a period equal to that of the length of the running mean smoother. The wavelength of the last harmonic included in (7.12) is $2\pi/N$; so smooth the response function in the following way:

$$\tilde{H}(\omega) = \frac{N}{2\pi} \int_{-\pi/N}^{\pi/N} H(\omega)\, d\omega \tag{7.19}$$

The running mean filter has no effect on the average, so substituting (7.12) into (7.19) we obtain:

$$\tilde{H}(\omega) = C_0 + \frac{N}{2\pi} \int_{\omega-\pi/N}^{\omega+\pi/N} 2\sum_{k=1}^{N} C_k \cos k\omega^* \, d\omega^* \tag{7.20}$$

$$= C_0 + \frac{N}{\pi} \sum_{k=1}^{N} \int_{\omega-\pi/N}^{\omega+\pi/N} C_k \cos k\omega^* \, d\omega^*$$

$$= C_0 + \frac{N}{\pi} \sum_{k=1}^{N} \frac{C_k}{k} \left\{ \sin\left[k(\omega + \pi/N)\right] - \sin\left[k(\omega - \pi/N)\right] \right\}$$

Expanding the sines and collecting terms, we obtain

$$\tilde{H}(\omega) = C_0 + 2\sum_{k=1}^{N} \left(\frac{\sin\left(\frac{\pi k}{N}\right)}{\frac{\pi k}{N}} \right) C_k \cos(k\omega). \tag{7.21}$$

The running mean smoother of the response function is equivalent to multiplication of filter weights by

$$\operatorname{sinc}\left(\frac{\pi k}{N}\right)$$

These factors are sometimes called the sigma factors. Note that the last coefficient, C_N, disappears entirely because the sigma factor is zero ($\sin \pi = 0$).

Figure 7.5.3 shows the response functions for the new set of weights determined by smoothing the response function.

$$\tilde{C}_k = \operatorname{sinc}\left(\frac{\pi k}{N}\right) C_k \quad \text{for} \quad 1 \le k \le N. \tag{7.22}$$

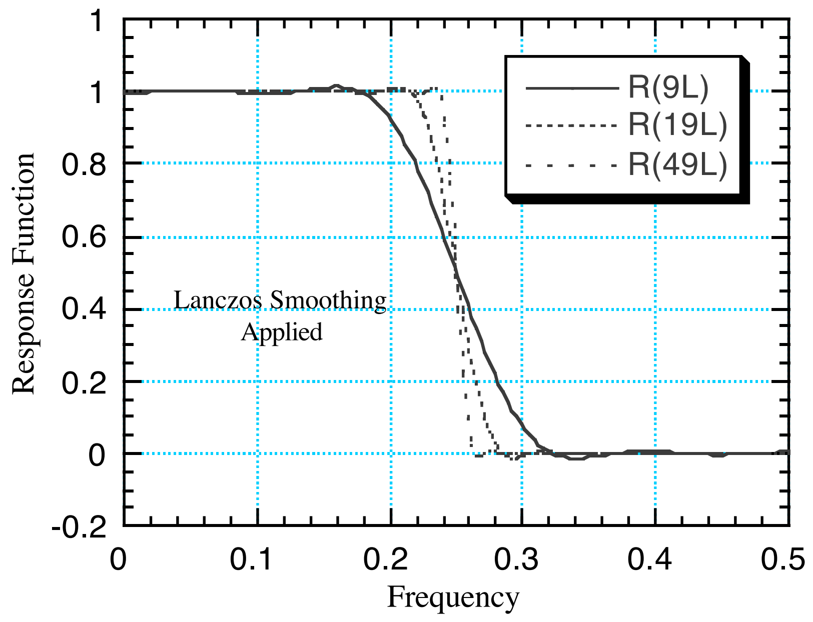


Figure 7.5.3: As in 7.5.2, but with Lanczos smoothing of weights and response function.

The wiggles are reduced by the Lanczos smoothing, but the frequency cutoff is somewhat more gradual. The changes to the weighting functions for the *N*=9 and *N*=19 cases are shown in Fig. 7.5.4. The sigma factors reduce the magnitudes of the weights as *k* approaches *N*.

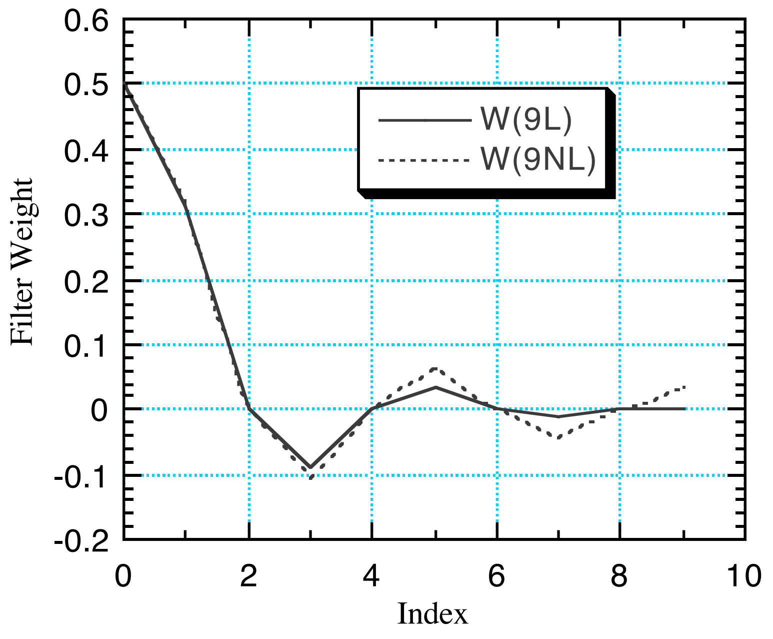


Figure 7.5.4: Filter weights for raw (dashed) and Lanczos smoothed (solid) response functions for case of *N*=9.

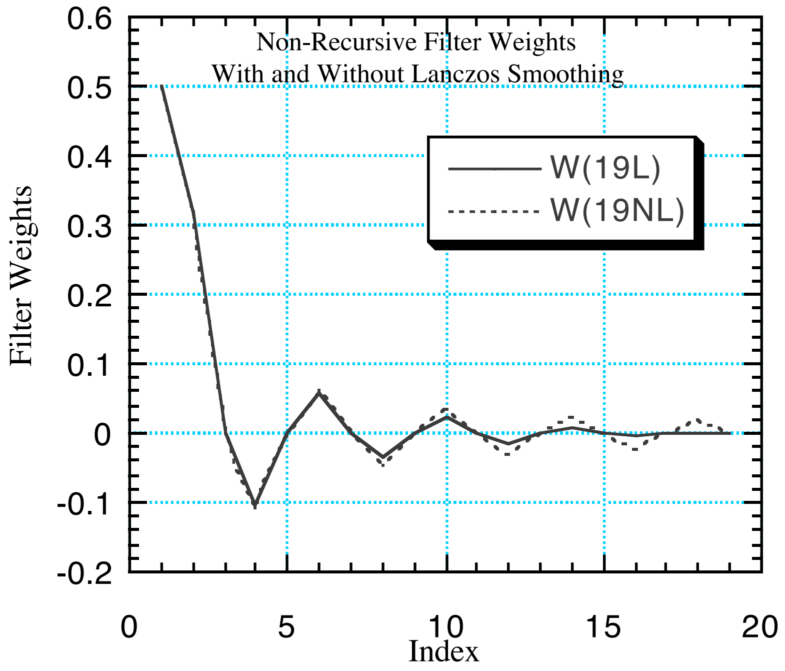


Figure 7.4.3b: Filter weights for raw (dashed) and smoothed (solid) response functions for case of *N*=19.

7.6 Recursive Filters

The filters we have discussed so far are obtained by convolving the input series $x(n\Delta t) = x_n$ with a weighting function w_k, in the following way.

$$y_n = \sum_{k=-K}^{K} w_k \; x_{n+k} \tag{7.23}$$

Such filtering schemes will always be stable, but it can require a large number of weights to achieve a desired response function. If greater efficiency of computation is desired, then it may be attractive to consider a recursive filter of the general form,

$$y_n = \sum_{k=0}^{K} a_k \, x_{n-k} + \sum_{j=1}^{J} b_j \; y_{n-j} \tag{7.24}$$

In this case the filtered value depends not only on the unfiltered input series, but also on previous values of the filtered time series. In general, sharper response functions can be obtained with fewer weights and thereby fewer computations than with non-recursive filters like (7.23). The method of constructing the weights for a recursive filter from a desired response function is not as easy as with convolution filters, and the filtering process is not necessarily stable.

7.6.2 Response Function for General Linear Filters:

Let's rearrange (7.24),

$$y_n - \sum_{j=1}^{J} b_j \, y_{n-j} = \sum_{k=0}^{K} a_k \, x_{n-k} \tag{7.25}$$

then Fourier transform (7.25), and finally utilize (7.8) and (7.11) to yield.

$$Y(\omega)\left\{1 - \sum_{j=1}^{J} b_j \, z^{-j}\right\} = X(\omega)\left\{\sum_{k=0}^{K} a_k \, z^{-k}\right\}$$

From which we obtain,

$$H(\omega) = \frac{Y(\omega)}{X(\omega)} = \frac{\left\{\sum_{k=0}^{K} a_k \, z^{-k}\right\}}{\left\{1 - \sum_{j=1}^{J} b_j \, z^{-j}\right\}} \tag{7.26}$$

Here H(ω) is the system function of the general recursive filter (7.24) and measures the ratio of the Fourier transform of the output function to the input function. In general H(ω) will be complex for recursive filters, which means that recursive filters will introduce a phase shift in the frequencies that they modify. This is because the filters are not symmetric, in general. Physically realizable filters, as might be employed to real-time data or in electric circuits, cannot be symmetric, since the future data are not know at the time the filtration of the present value must be produced. The real amplitude response function can be obtained from

$$|R(\omega)|^2 = H(\omega)H(\omega)^* \tag{7.27}$$

where the asterisk indicates the complex conjugate.

7.6.3 A Simple Recursive Filter

We can illustrate some important facts about recursive filters by considering the simple example of a recursive filter given by,

$$y_n = x_n + 0.95 y_{n-1} \tag{7.28}$$

The response function for this filter can be gotten from the general formula (7.26).

$$H(\omega) = \frac{1.0}{1.0 - 0.95 z^{-1}} \tag{7.29}$$

We can find the equivalent non-recursive filter by dividing out the rational factor in (7.29) to obtain a polynomial in z. The result is

$$H(\omega) = \frac{1.0}{1.0 - 0.95 z^{-1}} = 1.0 + 0.95 z^{-1} + 0.9025 z^{-2} + 0.8574 z^{-3} + 0.8145 z^{-4} + \ldots \tag{7.30}$$

Notice how slowly the coefficients of the polynomial decay. These coefficients are also the weights of the equivalent non-recursive filter. Thus many, many points are necessary to replicate the effect of the recursive filter (7.28) with a non-recursive filter.

7.6.4 Impulse Response of a Recursive Filter:

It is important to know how many data points a recursive filter must pass over before its response begins to settle out. This will indicate how many points must be disregarded off the end of a time series that has been filtered recursively. We can address this question by asking how the filter responds to a unit impulse time series of the form.

$$x_n = \begin{cases} 1.0 & n = 0 \\ 0.0 & n \neq 0 \end{cases} \tag{7.31}$$

The time series that results from filtering the time series (7.31) can be called the ***impulse response*** of the filter. One can verify that the filter (7.28) acting on the input time series (7.31) will produce the following filtered time series.

$$y_0=1.0,\ y_1=0.95,\ y_2=0.9025,\ y_3=0.8574,\ y_4=0.8145,\ \ldots \tag{7.32}$$

The impulse response(7.32) of the recursive filter (7.28) decays just like the coefficients of the equivalent nonrecursive filter, very slowly. So we conclude that we lose about the same number of endpoints with both types of filter. The only apparent advantage of the recursive filter is that it requires far fewer computations to achieve the same effect.

Unless a large amount of data must be processed, the easiest filter to implement that provides the desired functionality should be preferred.

The phase errors introduced by recursive filters can be reduced or eliminated by passing over the time series twice, once in the forward direction and once backward in time. The resulting amplitude response function is the square of the response function for a single application.

7.7 Construction of Recursive Filters

The construction of appropriate weights from a system function, or response function of desired shape is not quite as straightforward for recursive filters as for non-recursive filters, and requires different mathematics. How do we find the appropriate polynomial in *z* that produces the desired system function as described in (7.24)? Recall that z maps into the unit circle in the complex plane, as the frequency varies from zero to the Nyquist frequency. For a recursive filter to be stable, all zeros of the polynomial in the system function must be within the unit circle. It is also useful to realize that the z transform is linear, so that the system function of the sum of two filters is the sum of the system functions for each filter. Also, if two filters are applied successively, the system function of the result is the product of the system function for the two filters.

The construction of recursive filters, that is finding the appropriate weights, can be simplified by transforming the *z* variable to a *w* variable defined in the following way.

$$z = e^{i\omega\Delta t} = \frac{1+iw}{1-iw} \tag{7.33}$$

or

$$w = i\left(\frac{1-z}{1+z}\right) \tag{7.34}$$

This maps the unit circle in *z* space onto the real *w* axis, $-\infty < w < \infty$, and the stable zone inside the unit circle in *z* into the upper half of the complex *w* plane. The idea is to choose $R(\omega)$ as a simple function of w that produces the desired response. Choose the roots of this function that form a stable filter, i.e. $\mathrm{Im}(w) \geq 0$, to form the filter in w, then use the transformation to convert to a stable, rational polynomial in z. Then the coefficients are simply read from this polynomial, suitably factored.

7.7.1 Butterworth Filters

As a common example we can consider the Butterworth family of filters with response functions defined as follows.

$$H(\omega)H(\omega)^{*}=\frac{1}{1+\left(\frac{w}{w_c}\right)^{2N}} \tag{7.35}$$

The filter has the desirable property of smoothness and high tangency at the origin and infinity. It contains two design parameters, w_c and N, which can be used to design a filter with a cutoff at the desired frequency and the appropriate amount of sharpness to the cutoff. Using the following figure from Hamming(1989) to define the characteristics of the filter response function, we can derive formulas for the design parameters.

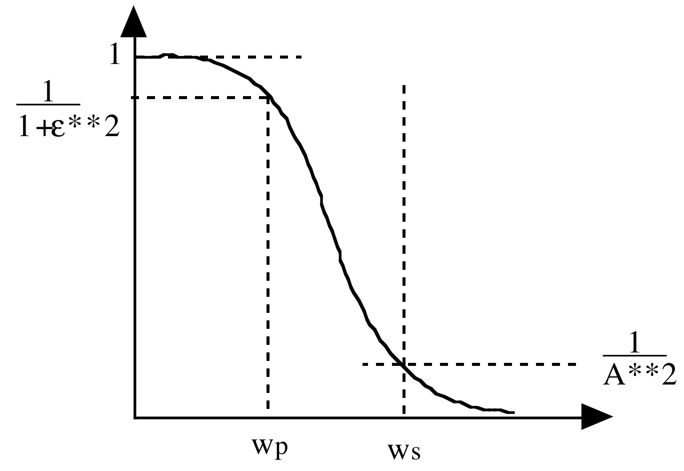


Figure: Diagram of a Butterworth filter response function showing some of the design parameters, wp, ws, ε and A.

$$N=\frac{\ln\left(\varepsilon/\sqrt{A^2-1}\right)}{\ln\left(w_p/w_s\right)} \tag{7.36}$$

$$w_c=\frac{w_p}{\varepsilon^{1/N}} \tag{7.37}$$

We see, as might be expected, that a sharp cutoff requires a Butterworth filter of high N. To find the coefficients of a recursive filter from (7.35) we first factor the denominator. The roots of the equation,

$$1+\left(\frac{w}{w_c}\right)^{2N}=0 \tag{7.38}$$

are

$$\left(\frac{w}{w_c}\right)^{2N}=-1=e^{i\pi(2k+1)} \tag{7.39}$$

or

$$\left(\frac{w}{w_c}\right) = -1 = e^{i\pi(2k+1)/2N} \quad (k = 1, 2, 3, \ldots 2N-1) \tag{7.40}$$

of these zeros, only those corresponding to *(k=1,2,3, N-1)* are in the upper half plane and so constitute a stable filter. They are the complex conjugates of the zeros in the lower half plane. We choose these zeros to form the system function for our filter. The zeros can be paired in such a way that *k* is paired with *N-k-1*. For each pair we have,

$$\left[\frac{w}{w_c} - e^{i\pi(2k+1)/2N}\right]\left[\frac{w}{w_c} - e^{i\pi(2N-2k-1)/2N}\right] \tag{7.41}$$

or

$$= \left[\frac{w}{w_c} - e^{i\pi(2k+1)/2N}\right]\left[\frac{w}{w_c} + e^{-i\pi(2k+1)/2N}\right]$$

$$= \left[\frac{w}{w_c}\right]^2 - 2i\sin\left(\frac{\pi(2k+1)}{2N}\right)\left[\frac{w}{w_c}\right] - 1 \tag{7.42}$$

Substituting the expression (7.33) for *w* in terms of *z*, we obtain,

$$\frac{[1/w_c]^2(1-z)^2 + 2\sin\{\pi(2k+1)/2N\}\left(1-z^2\right)[1/w_c] - (1+z)^2}{(1+z)^2} \tag{7.43}$$

This is one real quadratic factor corresponding to two complex conjugate linear factors derived from the two roots corresponding to k and N-k-1. There are at most N/2 of these quadratics.

If N is odd there will be one additional linear factor left over, corresponding to *k=(N-1)/2*, and therefore

$$\left(\frac{w}{w_c}\right) = e^{i\pi/2} \quad , \tag{7.44}$$

which has the factor,

$$\left[\left(\frac{w}{w_c}\right) - e^{i\pi/2}\right] = \left[\left(\frac{w}{w_c}\right) - i\right] \tag{7.45}$$

Substituting (7.33) gives

$$i\left[\frac{(-1/w_c)(1-z)-(1+z)}{1+z}\right] \tag{7.46}$$

Construction of a polynomial from its zeros leaves an undetermined multiplier. We can set this multiplier by requiring that the system response function be one at zero frequency, where z=1 and w=0. To achieve this we must multiply the quadratic factors (7.43) by -1, and the linear factor (7.46) by -i.

The system function is formed by multiplying together, in the denominator, all the quadratic terms and the final linear term if N is odd.

$$H(\omega) = \prod_{k=0}^{N/2-1} \frac{-(1+z)^2}{Quadratic\,Factors} \quad ; \; if \quad N \;\; even \tag{7.47}$$

and we would multiply (7.47) times the inverse of (7.46) if N is odd. We could carry out the multiplications in (7.43), which would give us polynomials in z of order N in the top and bottom. Alternatively, we could derive the recursive filter corresponding to each factor in the polynomial and apply them in succession to obtain the identical result, since the product of two system response functions corresponds to the system response function of applying the two filters successively. Each quadratic factor corresponds to the partial system response function,

$$H_k(\omega) = \frac{-(1+z)^2}{[1/w_c]^2(1-z)^2 + 2\sin\{\pi(2k+1)/2N\}\left(1-z^2\right)[1/w_c] - (1+z)^2} \tag{7.48}$$

Let's rearrange this a little

$$H_k(\omega) = \frac{w_c^2\left(1 + 2z^{-1} + z^{-2}\right)}{\left(w_c^2 + 2w_c\sin\{\pi(2k+1)/2N\}+1\right) + 2\left(w_c^2-1\right)z^{-1} + \left(w_c^2+1-2w_c\sin\{\pi(2k+1)/2N\}\right)z^{-2}}$$

This is now in the form where we can deduce the coefficients of the corresponding filter simply by reading off the coefficients, and performing the inverse z transform. If we rewrite (7.49) by substituting symbols for the coefficients,

$$H_k(\omega) = \frac{a_0 + a_2 z^{-1} + a_3 z^{-2}}{b_0 + b_1 z^{-1} + b_3 z^{-2}} \tag{7.50}$$

Then we can use (7.21) and (7.25) to infer that the corresponding recursive filter is,

$$\begin{aligned} y_n^k &= a_0\ x_n + a_1\ x_{n-1} + a_2\ x_{n-2} \\ &\quad + b_0\ y_n + b_1\ y_{n-1} + b_2\ y_{n-2} \end{aligned} \qquad (7.51)$$

Thus we get a simple second order recursive filter for each quadratic, which we must apply in sequence to get the full filter of order *2N*. We must also apply the linear part if *N* happens to be odd. Using the design parameters for the Butterworth filter to select *N* and w_c one can construct filters to suit many purposes.

Matlab has some programs for generating the filter weights of Butterworth and other recursive filters. Butterworth filters are smooth and monotonic, which are generally good characteristics for data work. If a sharper cutoff is required and negative side lobes are tolerable, there are other filtering schemes with these characteristics.

From within Matlab, we can enter,

```
>>[b,a]=butter(9,0.5)
```

which gives us the coefficients b and a from (7.21) and (7.25) for a Butterworth filter of order 9, so there are 9 a's and b's, 9 weights for the recursive and nonrecursive parts of the filter process. We have have asked for filter weights that cut the Nyquist interval at the midpoint (0.5). Matlab also provides a function for evaluating the filter response of this low pass filter.

```
>>[h,w]=freqz(b,a,128)
>>p=abs(h)
>>f=w/(pi*2)
>>plot(f,p)
```

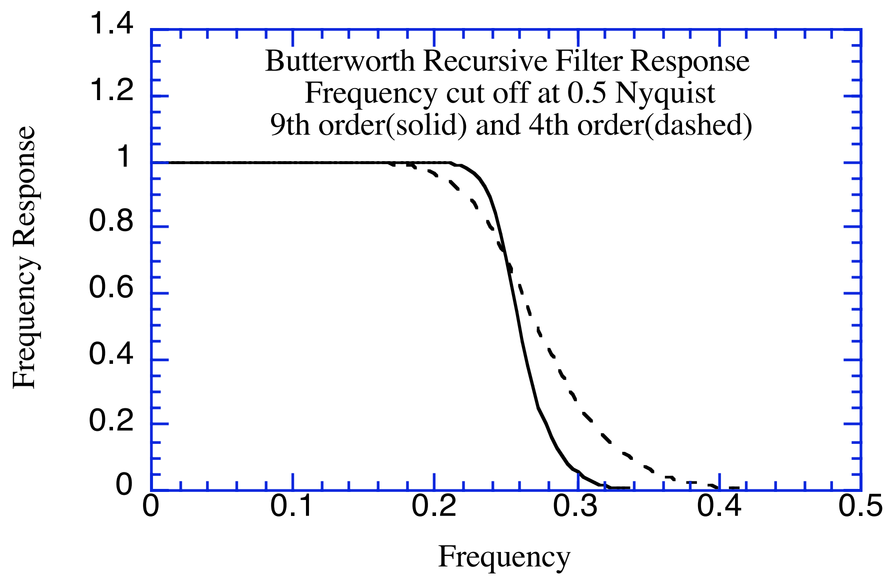
The result of this last command is similar to the following graph, where we compare the filter response for a ninth order Butterworth filter with that of a fourth order Butterworth filter, with the frequency cut taken at the same place halfway across the Nyquist interval of of frequencies ($0 < f < 0.5$).

The coefficients of the fourth order filter are given by the program:

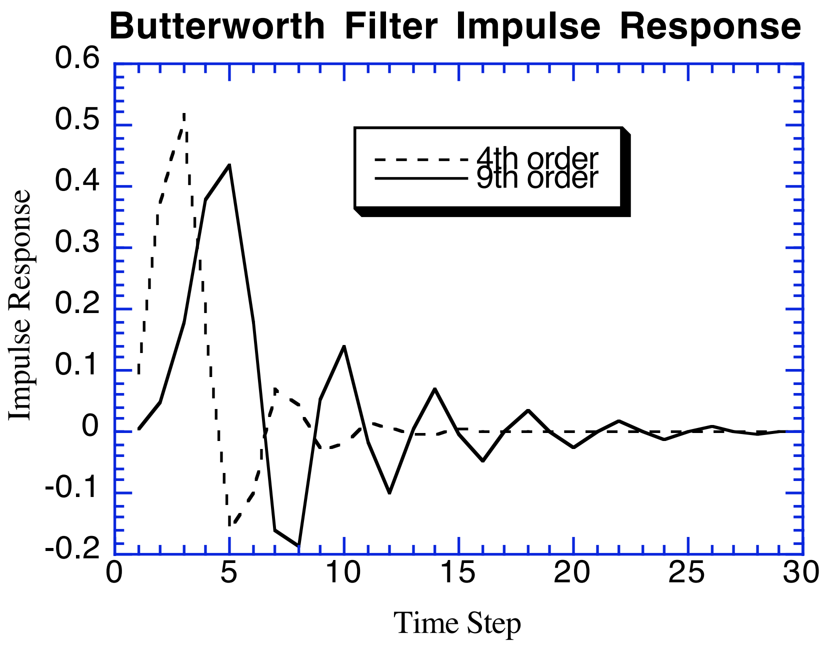
a_n = {1.0, 0.0, 0.486, 0.0, 0.0177}

b_n = {0.0946, 0.3759, 0.5639, 0.3759, 0.0940}

and these can be inserted into (7.21) to make a practical filter in a FORTRAN or C program, or the filtering can be done within Matlab.



The frequency cutoff is pretty good for the ninth order Butterworth filter. It is interesting to look at the impulse response of this filter (what it produces when you filter a time series that has one leading 1.0 followed by all zeros).



The impulse response is the influence function for the first data point of the time series, and tells you how much data at the beginning of the time series will be corrupted by the startup. The data point to the immediate left of the first data point would have had the same influence, shifted one time step to the left. For the 9th order Butterworth filter with a frequency cutoff at 0.5 takes about 20 days to settle down to a negligible level. As you might expect, the lower order filter, with the smoother response function, takes less time to settle out- about 10 days.

References on Filtering

Duchon, C. E., 1979: Lanzcos filtering in one and two dimensions. *J. Appl. Meteor.*, **18,** 1016-1022.

Hamming, R.W., 1989: *Digital Filters*. Prentice Hall, 284pp.

Holloway, J.L., 1958: Smoothing and filtering of time series and space fields. *Adv. Geophys.*, **4**, 351-389.

Jackson, L. B., 1996: *Digital filters and signal processing: with MATLAB exercises*. Kluwer Academic Publishers, 502 pp.

Kuc, R., 1988: *Introduction to Digital Signal Processing*. McGraw Hill, 474pp.

Little, J.N., L. Shure, 1988: *Signal Processing Toolbox for use with MATLAB*. The Math Works.