16.1 Introduction

Filtering a time series means removal of the spectral power at some chosen frequencies while retaining other frequencies. A high-pass filter retains higher frequencies while removing low frequencies; a low-pass filter does the opposite. A band-pass filter removes all frequencies outside a prespecified band.

Filters are widely used for digital signal processing (DSP) as well as time series analysis. In DSP applications, filters must be very efficient and they often must be causal (rely only on prior data samples to do the filtering in real time). This lecture will focus on simple Matlab-based filtering approaches for analysis of time series or spatial data, where these may be less important considerations.

16.2 Fourier filtering

The most obvious filter just uses the definition. If the time series is very long, truncate or zero-pad it to an efficient length. If needed, detrend or otherwise massage it so there is a minimal endpoint discontinuity. Then take its DFT, zero out the Fourier amplitudes of the frequencies that we don’t want (including corresponding negative frequencies), then take the IDFT to get the filtered time series. This is also a great way of getting rid of discrete frequencies such as a daily or annual cycle, as we have already noted.

A variant on this approach is to use windowed DFTs, but this leads to inconvenient discontinuities in the filtered time series at the edges of the windows.

Fourier filters are great for many types of data analysis, but they are very nonlocal, involving either periodicity assumptions or weighting all elements in the time series. This is problematic if we need to filter a finite non-periodic time series near its end points, in which case more sophisticated approaches are needed.

The script music2 applies Fourier high ($f > 880$ Hz), low ($f < 440$ Hz) and bandpass filters ($440 < f < 880$ Hz) to our musical segment, to show how their results look and sound.
16.3 Filter response

In Fourier space, we can describe a filter by its spectral response function \( R(f) \). If \( \hat{u}_m \) is the DFT of the original time series \( u_j \), and harmonic \( M_m \) and frequency \( f_m = M_m / (N \Delta t) \) correspond to index \( m \), then the DFT of the filtered time series \( v_j \) is

\[
\hat{v}_m = R(f_m) \hat{u}_m.
\]

Hence the filter affects the power spectrum:

\[
\frac{S_f}{S_m} = |R(f)|^2.
\]

For a perfect filter, \( R(f) = 1 \) and the filter power \( |R(f)|^2 = 1 \) for the frequencies being retained and they are zero for the frequencies removed. A Fourier filter (without tapering) has this ideal characteristic. Other filters always have some spectral leakage (spectral power in undesired frequencies) or phase shift (\( R(f) \) is complex, with nonzero phase).

16.4 Convolution theorem and spectral distortion due to tapering

The simplest kind of filter, called FIR (Finite Impulse Response) involves weighted averaging or convolution:

\[
v_j = \sum_{p=P_1}^{P_2} w_p u_{j-p}.
\]

Here the filter is defined by the weights \( w_p \). For instance, for a centered running mean over a window length \( (2P + 1) \Delta t \), we would take \( P_1 = -P \), \( P_2 = P \), and \( w_p = 1/(2P + 1) \), \( p = -P, \ldots, P \). This is a symmetric filter with equal weights at positive and negative lags, so does not create phase shifts. A running mean is often used as a crude low-pass filter, because it is easy to understand.

Unlike a Fourier filter, this filter is localized so the filtered time series uses only elements of the original time series within a finite range of lags \( p \) between \( P_1 \) and \( P_2 \). Given an ‘impulsive’ input which is zero except at a single time \( j = 0 \) at which it is 1, the filtered time series or impulse response function will be \( v_p = w_p, p = 0, \pm 1, \pm 2, \ldots \), which will also be zero except at a finite band of time lags \( P_1 \) to \( P_2 \), hence the name FIR.

The convolution theorem relates the filter response \( R(f) \) to the weights. The DFT version of this assumes that periodic extension of the time series \( u_j, j = 1, \ldots, N \), beyond its endpoints is used to calculate the convolution sums. It states that if we define the weight vector

\[
w = [w_0w_1 \ldots w_{N/2-1}w_{-N/2} \ldots w_{-1}] = \{w_M(m)\}
\]
(here \( w_p = 0 \) outside the range \( -P_1 \leq p \leq P_2 \)) and \( \hat{w} = \text{DFT}(w) \), then
\[
\hat{v}_m = \hat{w}_m \hat{u}_m.
\]
(16.4.3)

That is, the FIR filter obtained by convolution with \( w \) has a response function given by its DFT:
\[
R(f_m) = \hat{w}_m.
\]
(16.4.4)

Hence the filter power is proportional to the power spectrum of \( w \). If the filter is symmetric, it is easy to show \( \hat{w} \) is real, corresponding to \( R \) having zero phase shift at all frequencies.

Matlab script runningmean shows how to use the DFT and the convolution theorem to calculate the response function and power of a centered running mean filter of length \( 2P + 1 = 5 \), given a window length of 100. One sees that the running mean filter does a great job of zeroing out signals with \( f \Delta t = 1/5 \) whose period is \( 5\Delta t \), but is less efficient in removing most other high-frequency signals. Thus the running mean is far from being an ideal low-pass filter.

As also shown in the script, a Gaussian smoother \( w_p \propto \exp(-p^2/2P^2) \) with \( P \geq 1 \) can do a better job of removing high frequencies than can a running-mean smoother, at the expense of needing to extend the time series somewhat more to allow filtering near its end points. Such a filter will mainly pass frequencies \( f < (2\pi P \Delta t)^{-1} \); the case shown has \( P = 1.5 \).