Exercise 5. Butterworth Filters

The Matlab signal processing toolbox has an overwhelming array of options for designing and implementing filters, but for many geo-scientific applications we can use very simple filters. In this exercise we are going to explore the properties and use of a Butterworth IIR digital filter – perhaps the most commonly used filter.

We can create a Butterworth filter with the command

\[ [b, a] = \text{butter}(n, \text{wn}, \text{type}); \]

You should read the documentation but it is important to remember that

- The order of the filter \( n \) is the order of the polynomials defined by \( a \) and \( b \) (i.e., the number of poles or zeros). Usually \( n \) is chosen to be even. The higher the order the sharper the frequency cutoff.
- The frequency cutoff limit(s) of the filter \( \text{wn} \) are specified in units that go from 0 to 1 at the Nyquist frequency (not 0.5 as we used in class)
- \( b \) and \( a \) are as defined in class That is, the response of the filter is

\[
G(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z + b_2 z^2 + \ldots + b_M z^M}{a_0 + a_1 z + a_2 z^2 + \ldots + a_N z^N}
\]

To apply a filter to the sequence \( x \), execute the command

\[ y = \text{filter}(b, a, x); \]

This command calculates

\[ y_j = \sum_{i=1}^{N} b_i x_{j-i} + \sum_{i=1}^{M} a_i y_{j-i} \]

Note that I have adjusted the indexing to Matlab conventions. Now for any given output \( y_j \), the filter requires the \( N \) previous input values of \( x \) and the \( M \) previous output values of \( y \) and so will not be getting all the inputs until \( j = \max(N, M) + 1 \). Furthermore the feedback component will typically require several times \( M \) prior values to fully settle down (if you think about \( y_{M+1} \), it will be getting prior values of \( y \) but they will in turn have been calculated without the necessary prior values) – Unless the time series starts of with zeros, always give the filter a little extra data and delete the first part of the output.

1. For this questions with a 128-sample time vector with unit sample interval and a start time of 0 (\( t = 0:127 \)) and a input time series function comprising a delta function at 0 (\( x = \text{zeros}(1,128); \; x(1) = 1; \)).

(a) Create a 2\(^{nd} \), 4\(^{th} \), 8\(^{th} \) and 16\(^{th} \) order low-pass Butterworth filters with a cutoff at 0.5 of the Nyquist frequency. Apply them individually to the delta function time series and plot the results

(b) For each filter in (a) plot its amplitude and phase. The easiest way to calculate the amplitude and phase is with the following commands

\[ y = \text{filter}(b, a, x); \]

\[ Y = \text{fft}(y); \]

\[ f = \text{linspace}(0,1,129); \; f=f(1:128); \; f=f([1:64 \; 66:\text{end}]); \]

\[ Y = Y([1:64 \; 66:\text{end}]); \]

\[ Y\text{amp} = \text{abs}(Y); \; Y\text{phaselag} = -\text{unwrap}(\text{angle}(Y)); \]
(c) An alternative way to calculate the amplitude and phase is as follows:
You can either evaluate $G(z)$ at evenly spaced locations on the unit circle in the complex plane
and then obtain the amplitude and phase

```matlab
f = linspace(0,1,129); f=f(1:128); f=f([1:64 66:end]);
clear i;
for i=1:129
    z = exp(-i*2*pi*f);
    Y1alt = polyval(fliplr(b1),z) ./ polyval(fliplr(a1),z);
    Yamp1alt = abs(Y1alt); Yphaselag1alt = unwrap(angle(Y1alt));
end
```

Make sure that this gives the same result as you obtained in (b) for one input and explain briefly
how the two approaches shown work.

(d) Comment on the differences in the time domain and frequency-domain responses of the
filters.

(e) Repeat (a) except for a high pass filter with $\omega_n = 0.8$. Comment on the results.

(f) Repeat (a) except for a notch (stop) filter with $\omega_n = [0.4 \ 0.6]$. Comment on the results.

(g) The `filtfilt` command has the same arguments as `filter` but implements a zero-phase
filter by running the filter through the filtered data in both directions. Use a 128-sample time
vector with unit sample interval and a start time of -63 ($t = -63:64$) and an input time series
function comprising a delta function at $t = 0$ ($x = zeros(1,128); x(64) = 1$). Apply
the filters you constructed in part (a) to the delta function and plot the results. Comment on the
results.

2. The file `teleseism_can.mat` contains a vector $x$ which is a seismogram with a sample
intervals of 0.05 s for a teleseismic earthquake and a vector $t$ which is the time relative to the
first sample.

(a) Plot the seismogram.

(b) Construct a 4<sup>th</sup> order Butterworth high-pass filter with a cutoff frequency of 1 Hz and use it to
filter the seismogram. You should now see a 2<sup>nd</sup> smaller
earthquake that was completely obscured by the first in the unfiltered data. Zoom in on the
filtered and unfiltered data around the time of the 2<sup>nd</sup> earthquake and explain what you see.

(c) Zoom in on the first 5 seconds of the filtered and unfiltered time series. What do you see?
Can you explain what is happening?

(d) Construct and apply a 4<sup>th</sup> order Butterworth low-pass filter with a cutoff frequency of 0.1 Hz.
Plot the filtered seismogram on top of the unfiltered data and zoom in the onset of the main
earthquake. What is the effect of filtering on the apparent onset time of the earthquake? Can
you explain what you see in terms of your plots from Question 1d?
(e) Construct a 2\textsuperscript{nd} order Butterworth low-pass filter with a cutoff frequency of 0.1 Hz and use it to filter the data with the command \texttt{filtfilt}. Plot the result on top of the unfiltered data and zoom in on the onset time of the earthquake. What do you see and why?