FILTERING IN 1-D

Filtering of data is typically done in order to either smooth the signal or suppress power at particular frequencies or wave numbers. So far we’ve learned that filtering can be considered an example of a convolution between the data and the filter:

\[ h(t) = s(t) * f(t) \]

Thus, we can immediately take advantage of the convolution theorem and write

\[ H(\omega) = S(\omega) \cdot F(\omega) \]

Hence, we simply take the Fourier Transform of \( s(t) \) and multiply its spectral components by \( H(\omega) \). For example, we learned that a simple MA filter consists of convolving the signal with a rectangle function of width \( W \).

How does this operation look like in the frequency domain?

Thus, the MA filter results in multiplying the Fourier coefficients by the sinc-function. This means that some of the coefficients will have their signs reversed, which we know amounts to a phase change of 180°. Also, while the filter do fall off with increasing \( \omega \), it does so in a very oscillatory way and takes a long time to settle down to zero. This means the sinc function is a poor choice for a filter if you really wanted to cut out, say, high frequency information. Maybe instead we should design \( F(\omega) \) so that it truncates power at frequencies higher than \( \omega_{\text{cut}} \)? This operation would look like
The final result is obtained by transforming $H(\omega)$ back into the time domain:

$$h(t)$$

We find that the output had a lot of “ripples” at about the wavelength corresponding to $\omega_{cut}$. This “ringing” is caused by convolving $s(t)$ with $h(t)$ which is a sinc function. Because all power at higher frequencies are eliminated, the power at $\omega_{cut}$ stands out. [Remember Gibbs’ phenomenon]

It is clear from the two examples of the use of a rectangular-shaped function that the price we pay for having a sharp truncation of filter coefficients in one domain is excessive “ringing” in the other domain. This, of course, is simply the convolution theorem at work. A rectangle function is necessarily a poor choice for a filter because of the slowly decaying oscillatory nature of the sinc-function.

While we do desire to find a filter that rapidly tapers off power, say, beyond a certain frequency, we know that in the limit, when the gradual taper approximates a step-function, our transform will approximate a sinc function. How do we find a good filter? It will depend on the application. Because of the inverse relation between $t$ and $f$ ($f = 1/T$), we saw that the scale theorem became

$$g(at) \leftrightarrow \frac{1}{|a|} G\left(\frac{f}{a}\right)$$

Thus, making a filter broader in one domain narrows it in the other. In the limit ($T \to \infty$) we recover the transform pair

$$1 \leftrightarrow \delta(\hat{f})$$

Clearly, somewhere in the middle there must be a function that looks similar in both domains, i.e.,

$$g(t) \leftrightarrow G(f) = g(f)$$

There is such a function; here we will simply state that the Gaussian normal distribution behaves this way. Let

$$g(t) = e^{-\pi t^2}$$

To find its transform pair we must integrate
\[ G(f) = \int_{-\infty}^{\infty} e^{-\pi^2 t^2} e^{-2\pi i f t} dt = \int_{-\infty}^{\infty} e^{-(\pi^2 t^2 + 2\pi i f t)} dt \]

Notice that the exponent is almost \((a+b)^2\). We complete the square by adding and subtracting the missing term:

\[ G(f) = \int_{-\infty}^{\infty} e^{-\pi^2 t^2} e^{-2\pi i f t} dt = \int_{-\infty}^{\infty} e^{-\pi^2(t+i\pi f)^2} e^{i\pi f^2} dt \]

With \(u = \sqrt{\pi} t + i\sqrt{\pi} f\) and \(du = \sqrt{\pi} dt\) we get

\[ G(f) = \frac{1}{\sqrt{\pi}} e^{-\pi^2} \int_{-\infty}^{\infty} e^{-u^2} du = e^{-\pi^2} = g(f) \]

Filters based on the Gaussian curve are some of the most used filters in data analysis and data processing. Because of its smooth transform properties we know that it will minimize ringing in the other domain.

A well-known set of frequency-domain filters has the functional form

\[ F(s) = \frac{1}{1 + (s/s_0)^2N} \]  \hspace{1cm} (22.3)

These are called Butterworth filters and are often used for low-pass filtering purposes. The frequency \(s_0\) defines the halfway point of the filter: The power of \(F(s_0)\) is always 1/2, regardless of the exponent \((2N)\). The value of \(N\) determines how fast the filter falls off. The higher the value of \(N\), the faster the filter will drop off beyond \(s_0\).

We see the sharper the drop off, the more ringing in the time domain, reflecting Gibbs’ phenomenon caused by truncating the spectrum too rapidly. As always, there will have to be a trade-off between how sharply you want to reduce the spectrum and how much ringing you can tolerate.

**Median Filters**

We have assumed for most of the time that all filtering can be described by a convolution, and in general this is the case. The advantage of being able to write the filtering as a convolution is very important: Thanks to the convolution theorem we can simply transform data and filter into the frequency domain and perform a multiplication and one inverse transform. Why would we give up that advantage? Consider again the MA filter used earlier, say...
When convolved with data, it simply returns the mean value of the points inside the filter width. However, we know the mean is a least squares estimate of "average" value. What happens if we have occasional bad data points? Because it is $L_2$, it returns bad values for the entire filter width:

![Image of a signal with a filter applied, showing the mean value being returned]

Clearly, this is not a desirable situation. However, we can design a more robust filter by using a more robust estimate of the "average" value. A good estimator that is insensitive to the occasional outlier is the median:

$$
\tilde{x} = \begin{cases} 
  x_{n+1}, & n \text{ is odd} \\
  \frac{1}{2} \left( x_n + x_{n+1} \right), & n \text{ is even}
\end{cases}
$$

Unfortunately, the median is not an analytic function and has no transform (consider what its "impulse response" would be). Hence, we are forced to calculate the result median filtering in the time domain only as convolution is not applicable. An $n$-point median filter will, for each lag, return the median value of the current $n$ points. Thus, the filtering of the previous data example would be:

![Image of a signal with a median filter applied, showing the median values being returned]

In addition to being robust, i.e., insensitive to outliers, a median filter also preserves step-functions (provided the step length exceeds 1/2 the filter width.):

![Image of a step function and a median filter applied, showing the median values being returned]

For data that consists of a noisy signal superimposed on a step-like background, median filters are very useful. Consider the task of finding the "regional" depth from bathymetric data. The regional depth is "contaminated" by seamounts and faults, and may also abruptly increase or decrease across fracture zones (which are step functions in

$$
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
5 & 5 & 5 & 5 & 5
\end{bmatrix}
$$
Finding the median of all points inside the filter width at each output point seems like an expensive operation since the data must be sorted at each step. Fortunately a simple iterative scheme may be used to speed up operations. The median satisfies

$$\sum_{i=1}^{n} \frac{x_i - \tilde{x}}{|x_i - \tilde{x}|} = 0.$$ 

We may rewrite this as

$$\sum_{i=1}^{n} \frac{x_i}{|x_i - \tilde{x}|} - \sum_{i=1}^{n} \frac{\tilde{x}}{|x_i - \tilde{x}|} = 0 \Rightarrow \hat{x} = \frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} 1 - \sum_{i=1}^{n} \frac{\tilde{x}}{|x_i - \tilde{x}|}}$$

This scheme converges fairly quickly, especially since we would expect the median output from the previous filter position to be similar to the median for the next step.

Finally, one can design a filter based on the mode rather than the median. Such filters are called “maximum likelihood” filters and return the most frequently occurring value within the filter width. These filters are used to track typical levels through very noisy data. We can implement such a filter using the LMS approximation to the mode as discussed earlier.