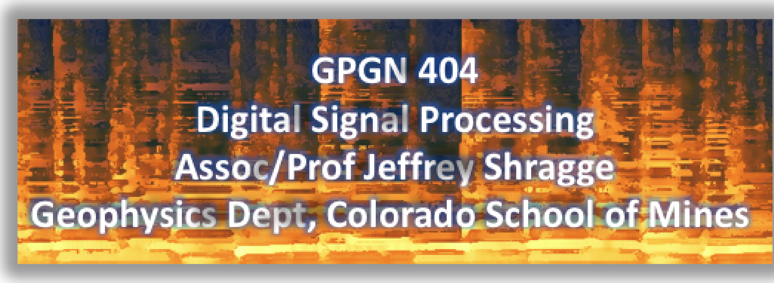
Out[34]: The raw code for this Jupyter notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.



Module 13: Practical Filtering

When confronted with geophysical data there are numerous situations where you seek to alter the frequency content of the signals in some particular fashion. For example, four key operations are:

1. **Low-pass filtering:** Removal of frequencies higher than some cut-off frequency f_C. This is also known as high-cut filtering.
2. **High-pass filtering:** Removal of frequencies lower than some cut-off frequency f_C. This is also known as low-cut filtering.
3. **Bandpass filtering:** Removal of frequencies lower than low-cutoff frequency f_{LC} and higher than high-cutoff frequency f_{HC}.
4. **Band-reject filtering:** Removal of frequncies in the range $[f_{LC}, f_{HC}]$. This is also known as notch filtering.

The notes in this section will explore some issues related to these filtering operations. Let's first look at the effects of introducting zeros and poles on our signals.

Effects of Zeros in the Transfer Function

Let's first examine the effect a zero has in the transfer function by looking at an example

Q: We want a filter that will have a zero gain at 400Hz. Calculate the transfer function $H(z)$ if the sampling rate is $f_s = 2000$ Hz.

A: We need only one zero at $f_c = 400$ Hz in the numerator (where f_c is the frequency to cut). Because we are interested in the frequency domain response, let's look at the effect of this operation on the unit circle (i.e., $r = 1$). This means that we're going to look at scenarios where $z|_{r=1} = e^{i\phi}$. Let's now evaluate where to put this zero on our normalized angular frequency axis:

$$\phi = 2\pi f_c / f_s. \tag{1}$$

Substituting $f_c = 400$Hz and $f_s = 2000$Hz yields the following on the z-plane $z = e^{i2\pi/5}$. Thus, let's construct a transfer function given by:

$$H(z) = z - e^{i2\pi/5} = z(1 - z^{-1} e^{i2\pi/5}) = \frac{1 - z^{-1} e^{i2\pi/5}}{z^{-1}}. \tag{2}$$

where the final two equalities are included because they will be introduced below.

Thus, when evaluated on the unit circle, this transfer function will be identically zero! Let's see how $H(z)$ will affect all other frequencies (note that I'm plotting the x-axis with respect to the expected frequency range of $f = \pm 1000$Hz, which is the Nyquist frequency). Here I plot the magnitude spectra of two functions: (1) an all-pass filter when $H(z) = z$ in black; and (2) a filter with a notch at $f_c = 400$Hz in blue.

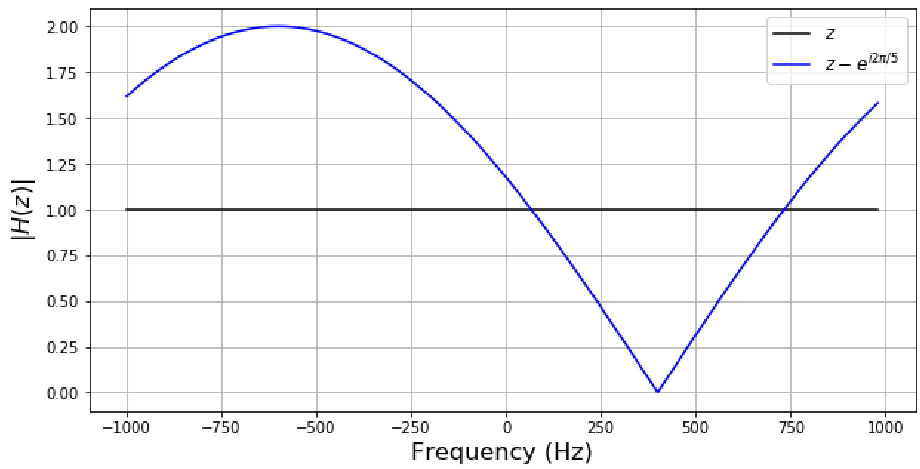


Figure 1. Example showing the result of including one zero at 400 Hz in the transfer function.

Note that we have successfully used a zero to eliminate frequency at $f = f_c/f_s = 0.2$ (here 400Hz). However, this has also significantly affected all of the other frequencies in spectrum. (Ideally, the value of $H(z)$ at all other values of f_c/f_s should be one.)

Finally, to look at how this would apply to an input $x[n]$ we can write the following:

$$Y(z) = H(z)X(z) = \left(z - e^{i2\pi/5}\right) X(z) = zX(z) - e^{i2\pi/5} X(z). \tag{3a}$$

Because z represents a unit **advance** (i.e., z^{-1} is a unit delay), this would correpond to the following filtering operation in the discrete time domain (i.e., n):

$$y[n] = x[n+1] - e^{i2\pi/5} x[n]. \tag{3b}$$

This can be easily solved by considering shifting everything by one index (see the section below).

Causality

Let's first examine whether this filter is realizable. First we note that the expression in equation 3b is not causal because it depends the output at n (i.e., $y[n]$) depends on a future $x[n+1]$. Thus, let's rewrite the expression in equation 3a in the Z-transform domain

$$Y(z) = zX(z) - e^{i2\pi/5} X(z) \tag{4a}$$

and then multiply both sides of this equation by z^{-1} to yield:

$$Y(z)z^{-1} = X(z) - e^{i2\pi/5} z^{-1} X(z). \tag{4b}$$

By applying the unit delay operator, the resulting difference equation is now given by:

$$y[n-1] = x[n] - e^{i2\pi/5} x[n-1]. \tag{5}$$

which is causal. The transfer function $H(z)$ related to this is the following causal expression:

$$H(z) = \frac{1 - e^{i2\pi/5} z^{-1}}{z^{-1}}. \tag{6a}$$

Real coefficients

Let's now get rid of the complex coefficients. We can do this by recognizing that the Fourier domain has both positive and negative frequency components. Thus, let's look at removing $f_c = -400$ Hz as well, which should follow the exact same approach as in equation 6 but with $f_c \to -f_c$.

$$H(z) = \frac{1 - e^{-i2\pi/5} z^{-1}}{z^{-1}}. \tag{6b}$$

Multiplying the expressions in equations 6a and 6b together generates the combined filter response:

$$H(z) = \frac{(1 - e^{i2\pi/5} z^{-1})(1 - e^{-i2\pi/5} z^{-1})}{z^{-2}} \tag{7a}$$

$$= \frac{1 - 2\cos(2\pi/5) z^{-1} + z^{-2}}{z^{-2}} \tag{7b}$$

Let's look at the filter response now:

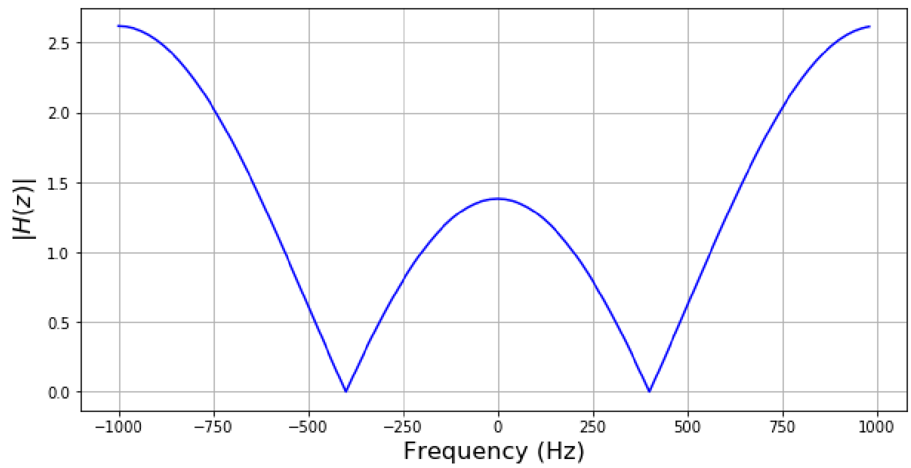


Figure 2. Example showing the result of including two symmetric zeros at $\pm$400 Hz in the transfer function.

Interesting! We now have real coefficients and have introduced a second zero at $f_c/f_s = 0.2$, which is here -400 Hz. Thus, the final difference equation is given by:

$$Y(z) = H(z)X(z) = \left(\frac{1 - 2\cos(2\pi/5)z^{-1} + z^{-2}}{z^{-2}}\right) X(z) \tag{8a}$$

Let's multiply through by z^{-2} to yield:

$$z^{-2}Y(z) = \left(1 - 2\cos(2\pi/5)z^{-1} + z^{-2}\right) X(z) \tag{8b}$$

Reverting back to the discrete time domain n yields the following operation:

$$y[n-2] = x[n] - 2\cos(2\pi/5)x[n-1] + x[n-2]. \tag{8c}$$

The impulse response of this filtering operator is given by the following stem plot.

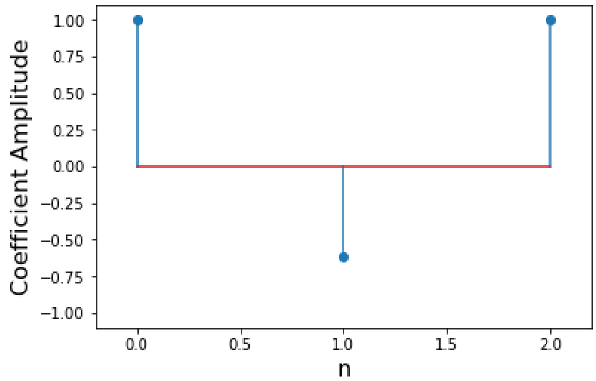


Figure 3. The equivalent filter to the above filtering operation.

Note that the notching frequency is controlled by the value of the coefficient at $n = 1$. Make this value larger (up to a maximum of 2) will increase the value of cut frequency f_c. Interestingly, when $f_c = 0$ then we recover the second-derivative finite-difference operator approximation.

Thus, if we wanted to apply this filtering operator in the time domain, we could **convolve** the expression $[1, -2\cos(2\pi/5), 1]$ with our time-domain signal $x[n]$.

Filtering out at Nyquist frequency

Now let's say that we also want to include a third zero at the Nyquist frequency, $f_c/f_s = 0.5$ which is here ± 1000Hz. This corresponds to introducing an additional multiplicative filter:

$$H(z) = \frac{1 - e^{-i\pi}z^{-1}}{z^{-1}} = \frac{1 + z^{-1}}{z^{-1}}. \tag{9}$$

Note that this equation applies to both $f_c = \pm 1000$Hz. Multiplying these through yields

$$H(z) = \left(\frac{1 - 2\cos(2\pi/5)z^{-1} + z^{-2}}{z^{-2}}\right)\left(\frac{1 + z^{-1}}{z^{-1}}\right) \tag{10}$$

Let's look at how this has affected the filter's magnitude spectrum:

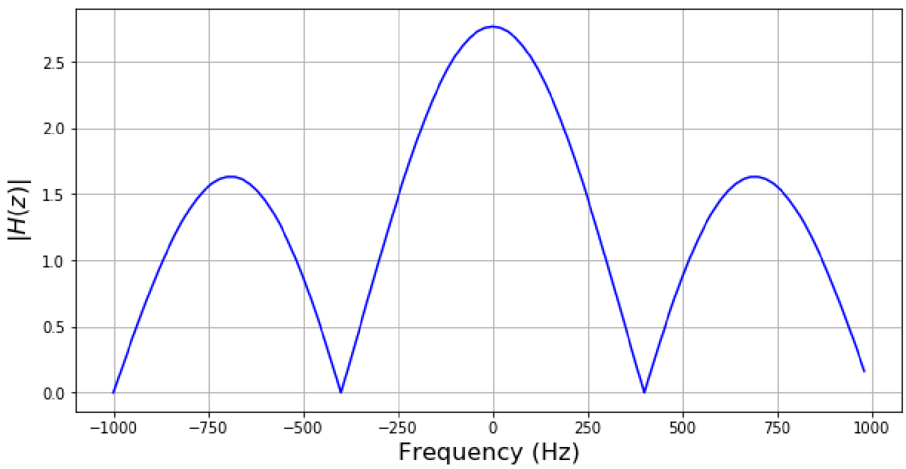


Figure 4. The filtering operation in Figure 2, but now with zeros at Nyquist frequency.

We can now see that we have also (nearly) removed the frequency at $f_c/f_s = 0.5$ (here ± 1000 Hz). Let's look a the impulse response with the three zeros.

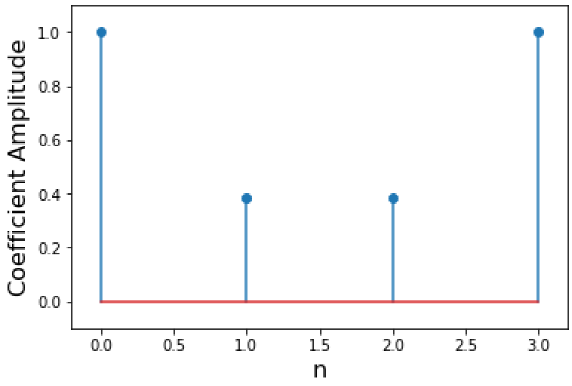


Figure 5. The filter as in Figure 3, but now designed to remove at Nyquist frequency as well.

Applying this filter in the time-domain would again be the equivalent of performing a **convolution** with the filter coefficients defined in the plot above.

Summary - Effects of adding zeros

1. By adding zeros at a specific frequency, we obtain zero gain at that frequency.
2. We have to use conjugate symmetric zeros to obtain real coefficients.
3. We have to introduce delays to obtain a causal form of the filter.
4. This filter will always be FIR (finite impulse response) because there is no dependence on multiple output values $y[n]$ to form a recursion relationship.

Effect of Poles on Frequency Response

Let's examine a filtering desire related to the above that we can address with adding **poles** to the denominator of the expression.

Q: We would like to apply a large 20x gain at a boosting frequency of $f_b = 200$Hz. What would the transfer function $H(z)$ be if the sampling rate is $f_s = 2000$Hz.

A: Let's examine a solution using only one **pole**. However, if the pole were located on the unit circle then we would have an infinite gain due to zero division. Thus, let's look at a solution where the pole is at $f_c = 200$Hz is in the demoninator close to (and inside) of the unit circle at r=0.95. (The choice of 0.95 will be clear below.)

$$\phi = 2\pi f_b/f_s. \tag{11}$$

Substituting $f_b = 200$Hz and $f_s = 2000$Hz yields the following on the z-plane $z = e^{i2\pi 200/2000} = e^{i\pi/5}$. Thus, let's construct a transfer function given by:

$$H(z) = \frac{1}{z - 0.95e^{i\pi/5}} = \frac{1}{z(1 - 0.95e^{i\pi/5}z^{-1})} = \frac{z^{-1}}{1 - 0.95e^{i\pi/5}z^{-1}}, \tag{12}$$

where the final two equalities are again included because they will be used below.

We'll see that the magnitude $|H(z)| = \frac{1}{|1-0.95|} = \frac{1}{0.05} = 20$. Thus, we will observe a 20$\times$ magnification at the desired frequency. What about the other frequencies?

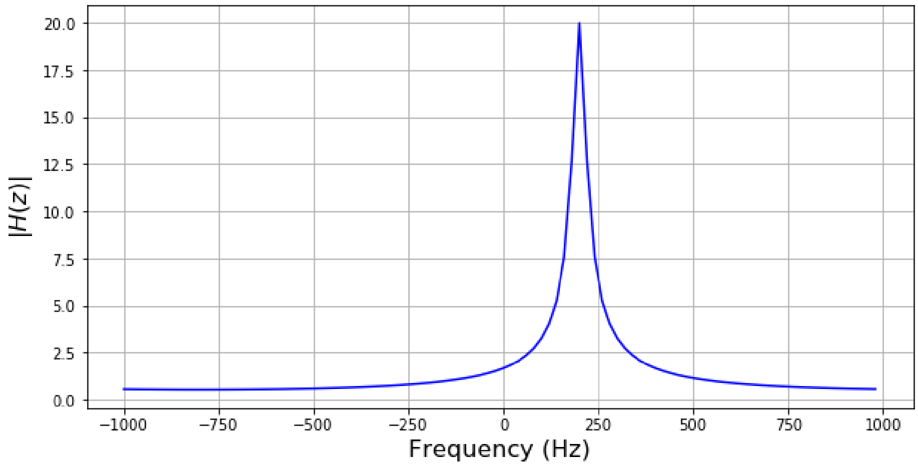


Figure 6. Effect of introducing a pole at 200 Hz at $= 0.95$ which results in a 20x amplification.

We see that there is a 20x increase at $f_b/f_s = 0.1$, which is here $f_b = 200$Hz. However, the frequencies about the f_b are also significantly affected.

Let's look again at whether this transfer filter is causal and with real coefficients.

Causality

We note that the difference equation that results from this is not causal and it is **recursive**:

$$y[n+1] = x[n] + 0.95e^{i\pi/5}y[n]. \tag{13}$$

Let's look at the Z-transform of equation 13:

$$Y(z)z = X(z) + 0.95e^{i\pi/5}Y(z) \tag{14}$$

and then get rid of causality issues by multiplying both sides by z^{-1}:

$$Y(z) = X(z)z^{-1} + 0.95e^{i\pi/5}Y(z)z^{-1} \tag{15}$$

or

$$y[n] = x[n-1] + 0.95e^{i\pi/5}y[n-1]. \tag{16}$$

Thus, $H(z)$ is given by the **causal** and **recursive**:

$$H(z) = \frac{z^{-1}}{1 - 0.95e^{i\pi/5}z^{-1}}. \tag{17}$$

Because this is **recursive**, this forms an IIR filter.

Getting rid of complex coefficients

Similar to the previous case, let's include a second pole at the corresponding negative frequency component:

$$H(z) = \frac{z^{-2}}{(1 - 0.95e^{i\pi/5}z^{-1})(1 - 0.95e^{-i\pi/5}z^{-1})} \tag{18}$$

$$= \frac{z^{-2}}{1 - 1.9\cos(\pi/5)z^{-1} + (0.95)^2 z^{-2}} \tag{19}$$

Let's look at how this affects the transfer filter $H(z)$:

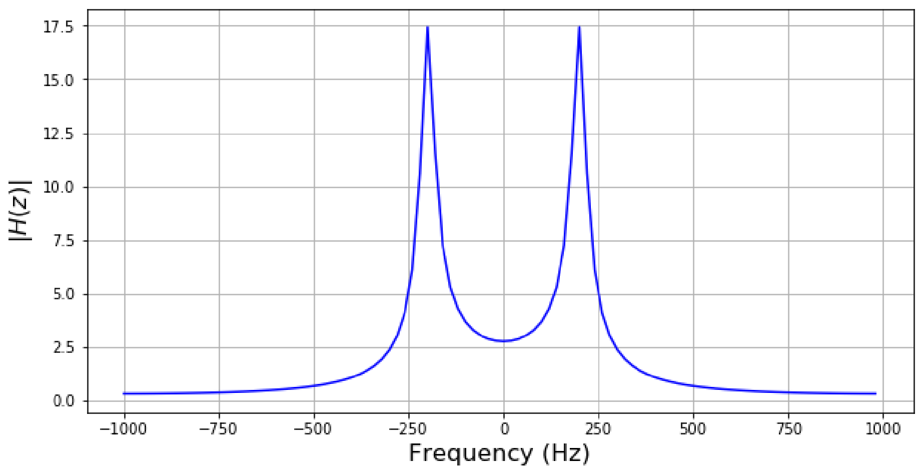


Figure 7. Effect of introducing poles at ±200 Hz at $= 0.95$ which results in a nearly 20x amplification.

Let's look at the response of this filter in the time domain. However, this is not as straightforward as above where we could read off the coefficients through a convolution process. This is because of the recursive nature of the filter! However, we can estimate its response by using numpy's *lfilter* routine.

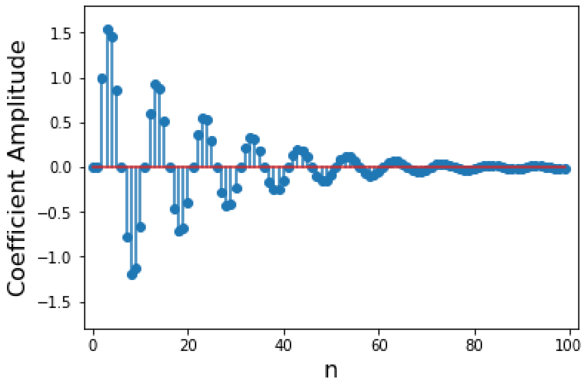


Figure 8. Equivalent time-domain convolutional filter that can be applied (note this was truncated at $n = 100$ for illustration purposes).

We see that this filter keeps on ringing with non-zero values out to a very large number of samples.

Note that applying this filter would require a **convolution** of the above filter impulse reponse with your time-series data.

Effect of pole distance to unit circle

The next question examines what is the effect of choosing a different r than 0.95? Let's do a bit of a test here:

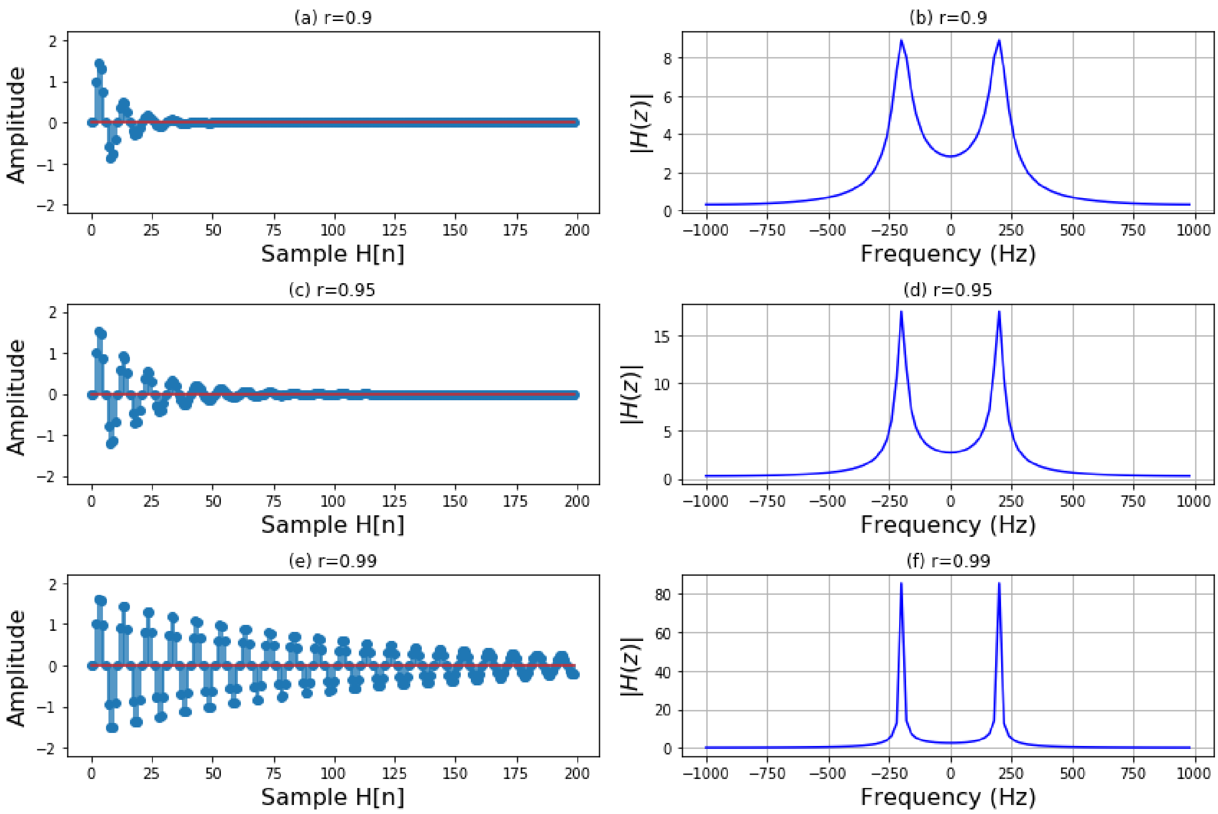


Figure 9. Illustration of the effect of changing r. (a) $r = 0.9$ and (b) associated Fourier magnitude spectrum. (c) $r = 0.95$ and (d) associated Fourier magnitude spectrum. (e) $r = 0.99$ and (f) associated Fourier magnitude spectrum.

We see that moving $r \to 1$, results in a more spatially localized frequency-domain filter (right-hand side); however, the trade off is that the time-series decays much more slowly (left-hand side), meaning that you need a much longer filter in order to realize the **convolutional** filtering operation.

Summary Comments

1. Locating a pole close to the unit circle (e.g., $r = 0.99$) will result in an oscillatory filter that can boost a signal at the chosen boost frequency f_b.
2. As pole gets closer to origin, the decay term is increased without any change in oscillation frequency, which is controlled by the f_b coefficient.

Low-pass Filtering with FIR operators

Recall that the magnitude of the frequency response of $H(z)$ determines the filter type (i.e., low-pass, high-pass, etc). For a FIR filter we may only use **zeros** to adjust the magnitude of the filter. In addition, as the order of the filter increases, additional **poles at z=0** are required due to causality arguments.

Here are a number of questions related to low-pass filter (LPF) design:

1. What is the Fourier transform of a pulse?
2. What is the ideal frequency form of a LPF?
3. What is the impulse response of an ideal LPF?
4. What is the method to design the most ideal LPF?

Using three coefficients (in the numerator), we want to design a FIR low-pass filter with cut-off frequency $f_c = 500$ Hz where our sampling rate is $f_s = 2000$ Hz. We know that we are looking at $\phi = 2\pi f_c/f_s = 0.5\pi$ However, we do not yet know how to find the coefficients $[b_0, b_1, b_2]$ of the following transfer function:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{z^{-1}}. \tag{20}$$

This filter has two-zeros and a two-fold pole at $z = 0$.

Obtaining Filter Coefficients

To obtain the required FIR filter coefficients we can use the built-in

scipy.signal.firwin(numtaps,cutoff)

command, which generates a set of optimized coefficients given the number of coefficients (numtaps=3) and the cut-off frequency $f_c/f_s = 0.5$ (i.e., with respect to the sampling rate f_s). Let's see what happens!

```
Lowpass numerator coefficients:  [ 0.0462215  0.907557   0.0462215]
```

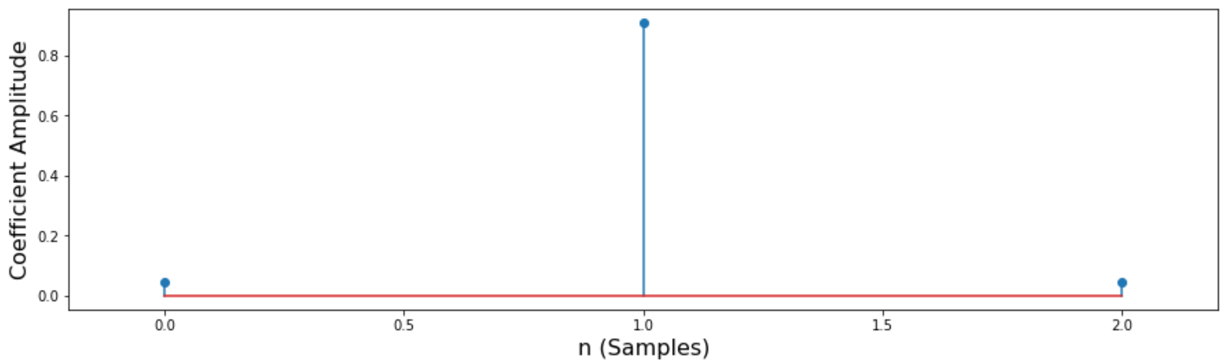


Figure 10. Illustration of the optimal filter coefficients when $Ncoeffs = 3$ is chosen for the low-pass filtering operation.

I've written out the coefficients so that you can see them ... note that they are symmetric. Let's now check out the effect of increasing the order of the approximation by increasing the number of coefficients (numtaps).

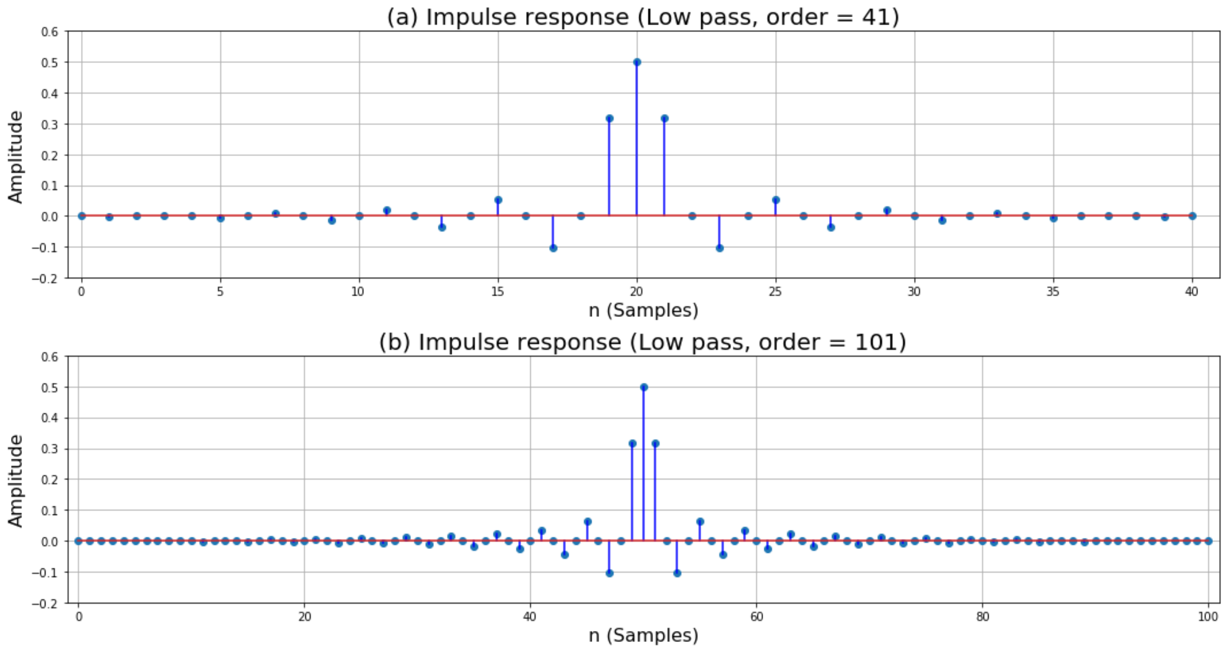


Figure 11. Illustrating the impulse response of different orders of low-pass filter. (a) 41 coefficients. (b) 101 coeffiients.

In the above examples, note the following:

1. Each impulse response has $numtaps + 1$ impulses
2. The filter is delayed by $numtaps/2$ for causal realization

How does this affect the frequency responses of the operators? To examine this we can use the python commands **freqz** which will give you the frequency response of the filter.

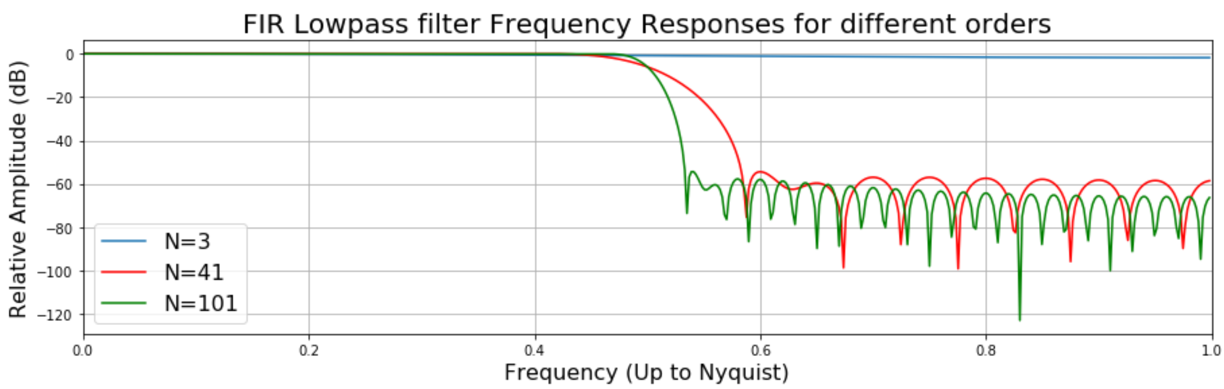


Figure 12. Resulting quality of the low-pass filtering operation for different orders. Vertical scale is in dB down.

Note that we are plotting the y-axis in terms of dB, which is a logarithmic scale! As the order of the filter increases, the filter's transition band becomes narrower and frequency response becomes sharper through cut-off frequency (which is $0.5 f_c/f_s$ for this case). Also, the filter is delayed by $numtaps/2$ for causality reasons.

High-pass filtering with FIR filters

Let's now look at high-pass filtering use the similar approach as before. We can actually use the same procedure as before, but pass an additional argument to *firwin*: *pass_zero=False*. Essentially, this says "take -1 of what we did previously!

Here's the examples for the 12- and 40-point high-pass filter responses:

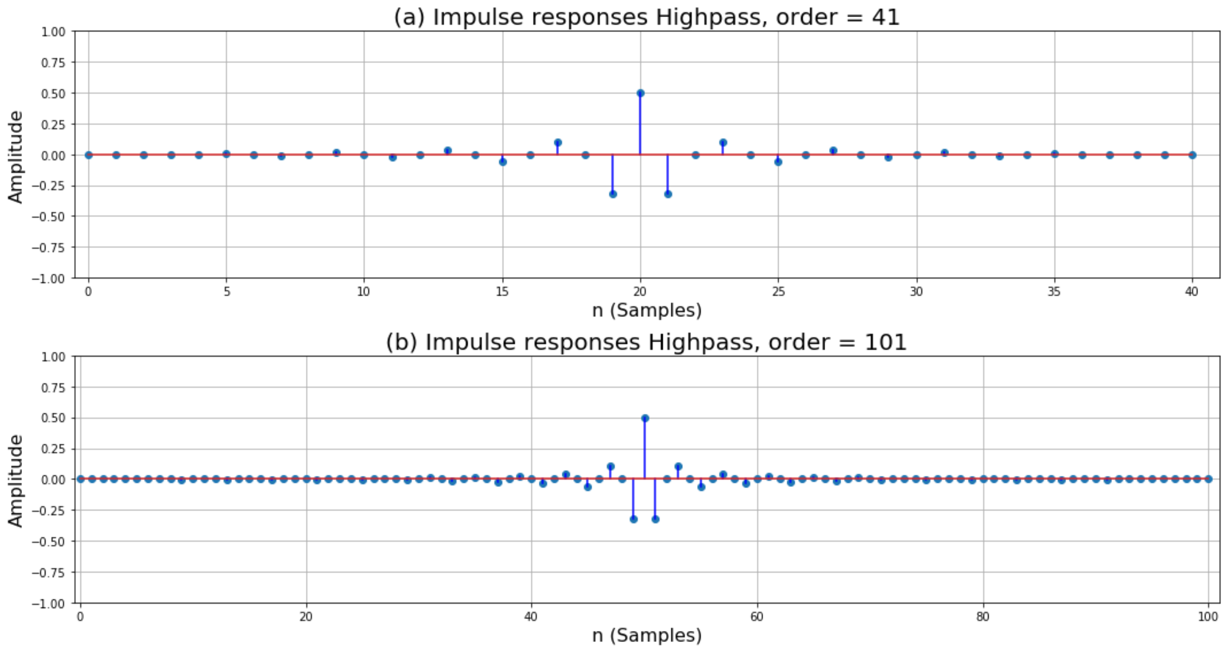


Figure 13. Illustrating the impulse response of different orders of high-pass filter. (a) 41 coefficients. (b) 101 coeffiients.

Let's now look at the frequency response for the different orders:

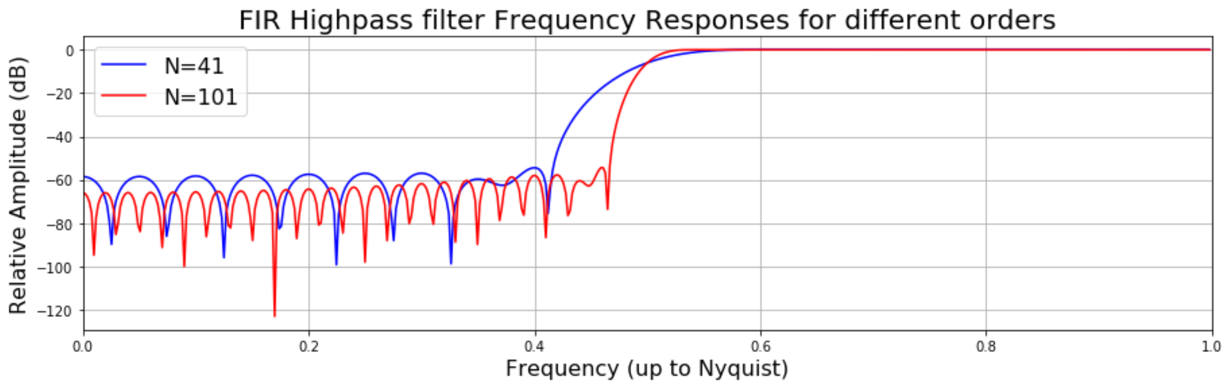


Figure 12. Resulting quality of the high-pass filtering operation for different orders. Vertical scale is in dB down.

Bandpass filtering with FIR filters

We also may want to apply a bandpass filter that is the combination of the low- and high-pass filtering results. To do this we can again call *firwin* in a modified way. Instead of giving a single f_c, we can give two values that are interpreted as the low- and high-cut frequencies, f_{lc} and f_{hc}, respectively. The example we are looking at uses $f_{lc} = 0.2$ and $f_{hc} = 0.4$.

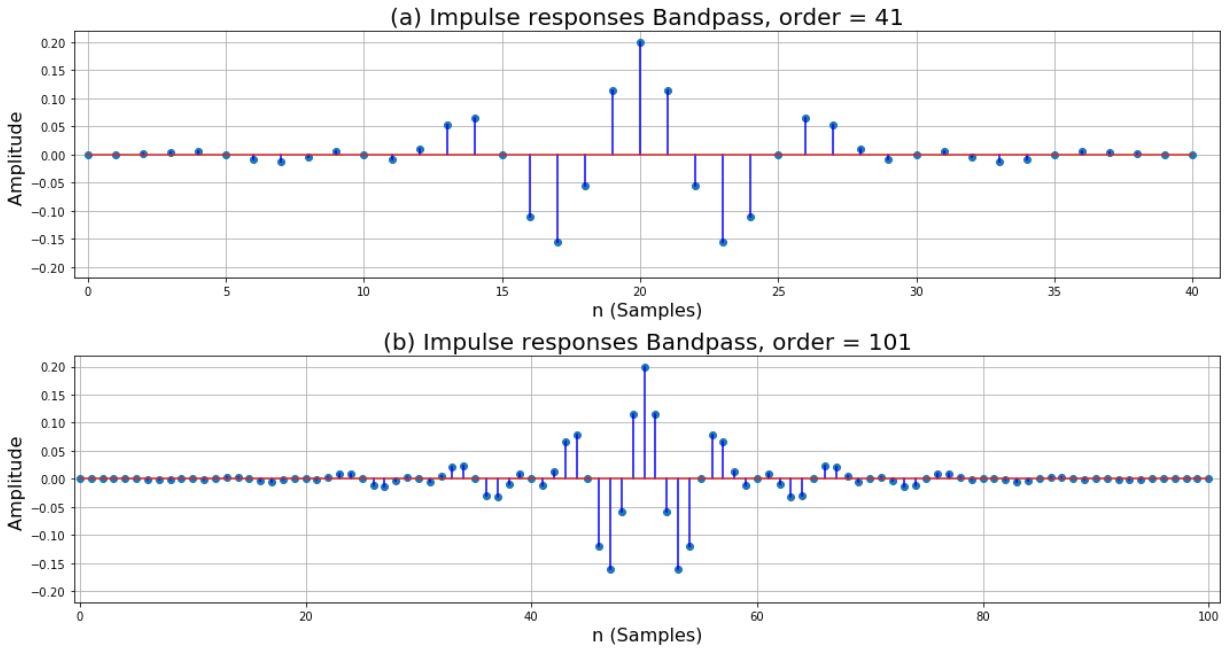


Figure 14. Illustrating the impulse response of different orders of band-pass filters. (a) 41 coefficients. (b) 101 coefficients.

Let's look at the frequency responses:

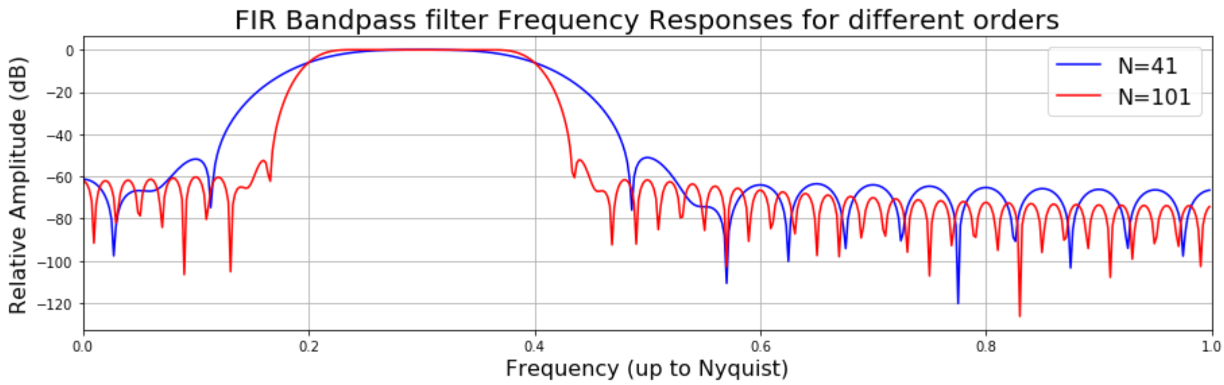


Figure 15. Illustrating the impulse response of different orders of high-pass filter. (a) 41 coefficients. (b) 101 coefficients.

Band-reject using FIR filters

Finally, the last permutation of the cases is to look at band reject. We can do the same as the bandpass filtering above, but we have to remove the *pass-zero* option (or set it to True).

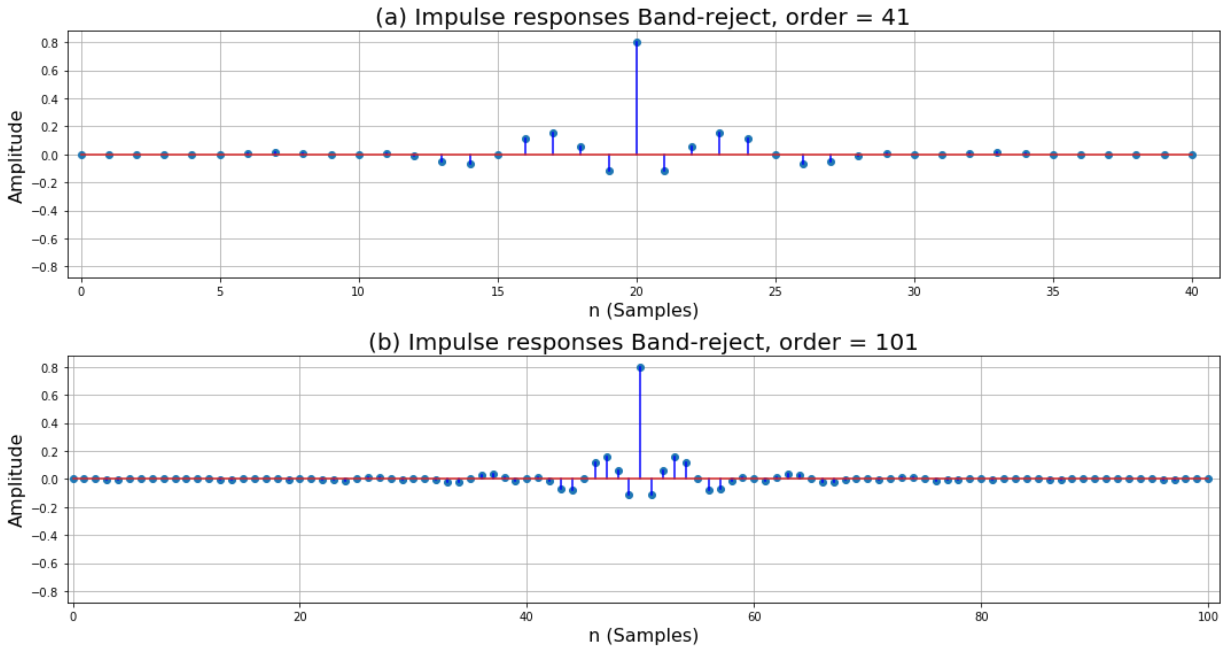


Figure 16. Illustrating the impulse response of different orders of band-reject filters. (a) 41 coefficients. (b) 101 coefficients.

And also the frequency response:

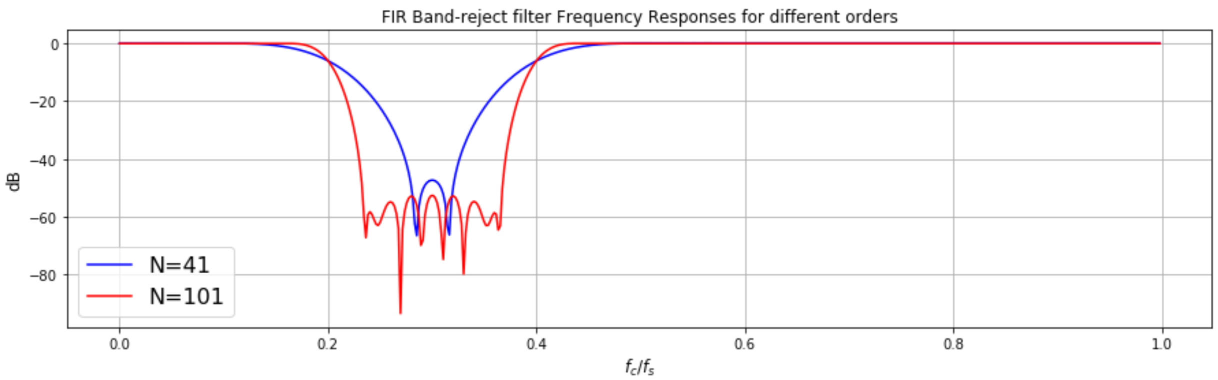


Figure 17. Illustrating the impulse response of different orders of band-reject filters. (a) 41 coefficients. (b) 101 coeffiients.

Example: Filtering a multi-tone signal

Now we will generate a multi-component sinusoidal signal with frequencies 440 Hz, 880 Hz and 1320 Hz that is sampled at f_s=44100 Hz. These frequencies are equivalent to the A note below the middle C note on a piano as well the two octaves above it. We choose an arbitrary duration of 3s.

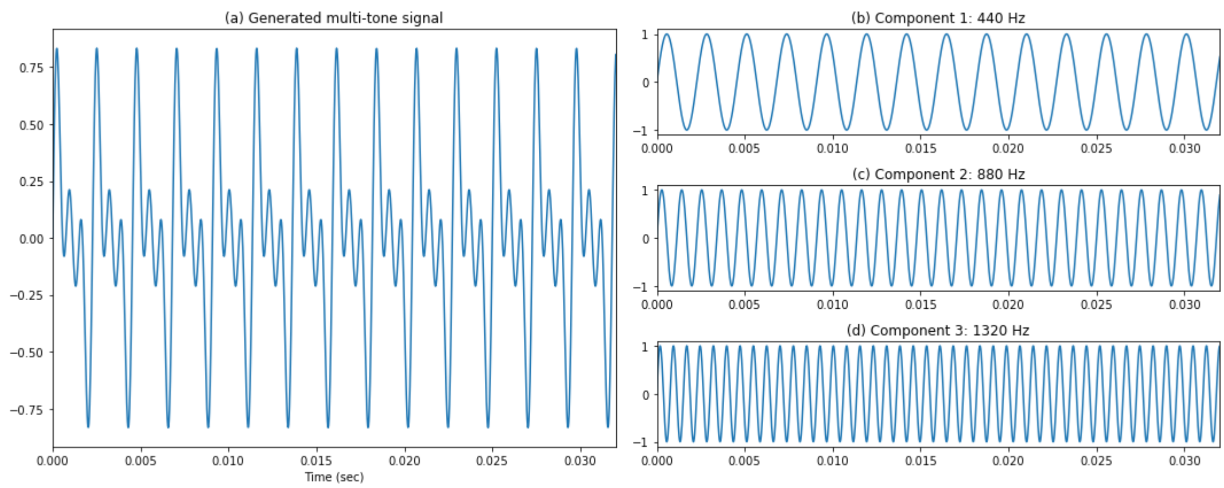


Figure 18. (a) Superposition of three notes comprised of (b) low (440 Hz), (c) medium (880 Hz) and (d) high (1320 Hz) frequencies.

Let's listen to these three notes superimposed:

Out[21]:

0:00 / 0:03

Low-pass filtering

Let's look at low-pass filtering of the above that will pass only the 440Hz components. For this we have to design a lowpass filter with a cut-off frequency larger than 440 Hz. Let's pick cut-off as 660 Hz and order as 101 points. To apply the filter we calculated with *firwin* we use the *lfilter* routine:

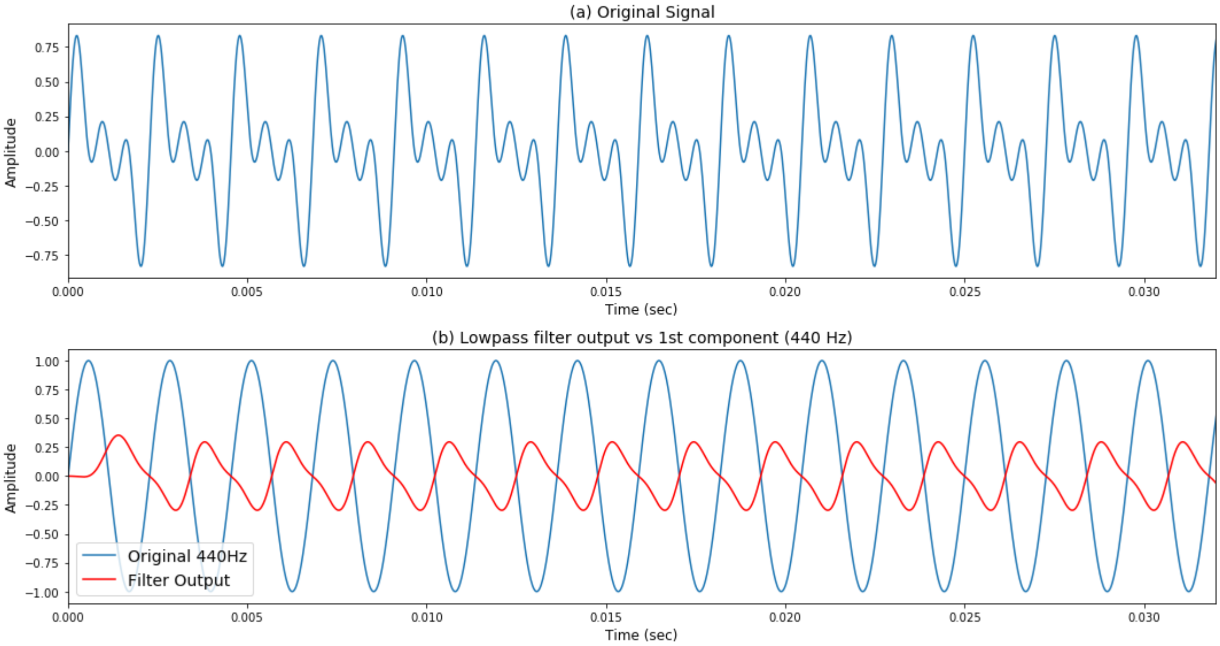


Figure 19. Low-pass filtered result showing (a) original signal and (b) signal after filtering out all but the lowest-frequency component.

Let's listen to the low-pass filtered signal as well as the original:

Out[23]:

0:00 / 0:03

Out[24]:

0:00 / 0:03

High-pass filtering

Let's now do some high-pass filtering to remove only frequencies 440 Hz and 880 Hz from the original signal. Thus, we need to design a highpass filter with a cut-off frequency that is smaller than 1320 Hz. Let's pick it as 1100 Hz and order 101.

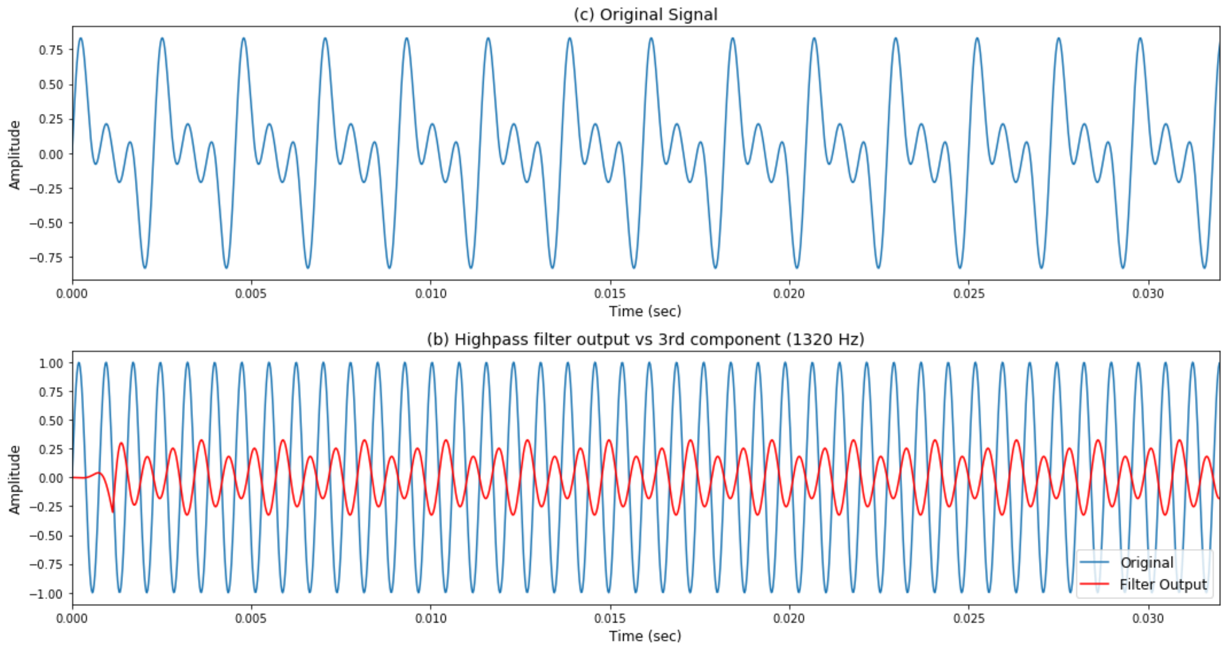


Figure 20. High-pass filtered result showing (a) original signal and (b) signal after filtering out all but the highest-frequency component.

Let's now listed to the high-passed and original signals:

Out[26]:

0:00 / 0:03

Out[27]:

0:00 / 0:03

Bandpass filtering

Let's now do some band-pass filtering to remove only frequencies 440 Hz and 1320 Hz from the original signal. Thus, we need to design a bandpass filter with a cut-off frequency that is greater than 440Hz but smaller than 1320 Hz. Let's pick it as 660 and 1100 Hz and order 101.

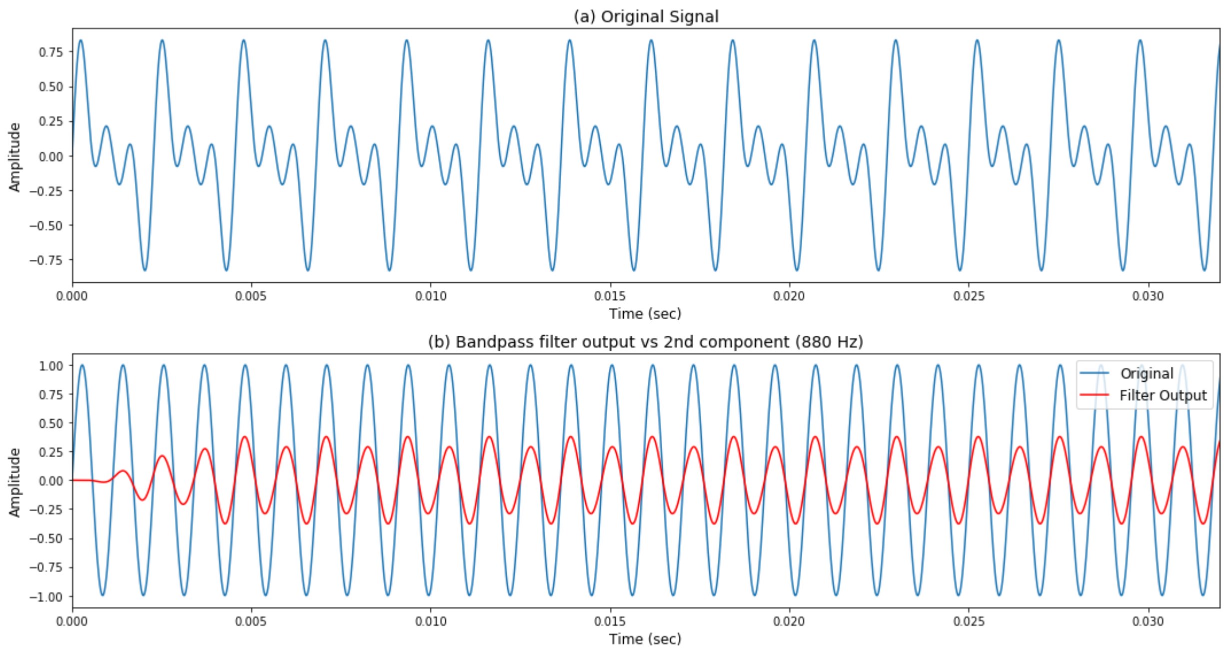


Figure 21. High-pass filtered result showing (a) original signal and (b) signal after filtering out all but the middle-frequency component.

Let's now listed to the bandpassed and original signals:

Out[29]:

0:00 / 0:03

Out[30]:

0:00 / 0:03

Band-reject filtering

Finally, let's now do some band-reject filtering to remove only 3kHz frequency while keeping that at 440 Hz and 1340 Hz. Thus, we need to design a band-reject filter. Let's do this in a similar way as above by pass an extra argument that states we're going to have zero in the pass band.

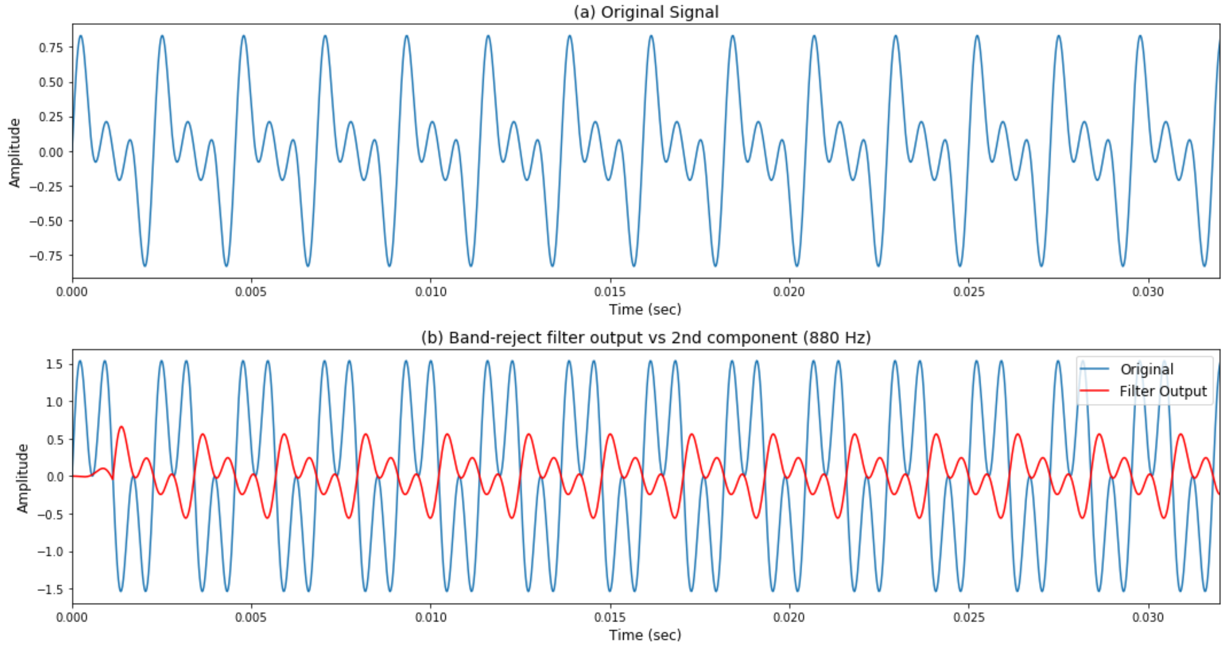


Figure 22. Band-reject filtered result showing (a) original signal and (b) signal after filtering out only the middle-frequency component.

Let's now listed to the band-rejected and original signals:

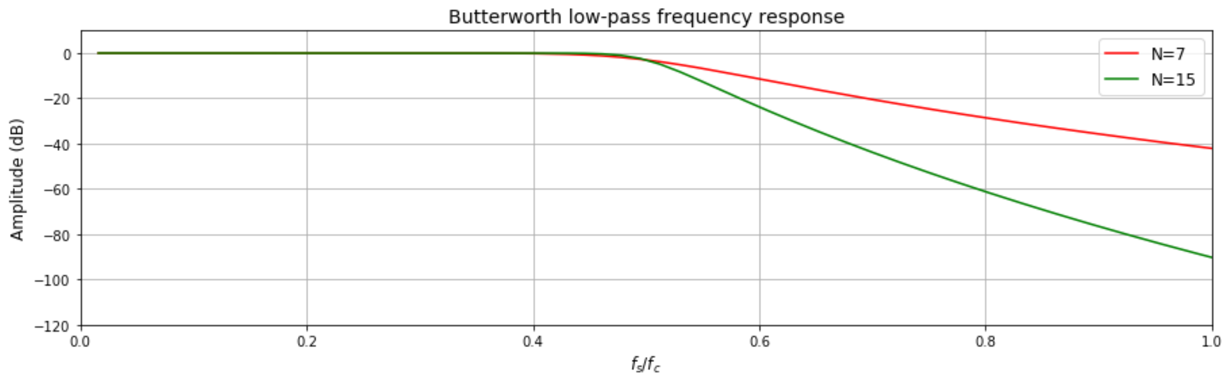
Out[32]:

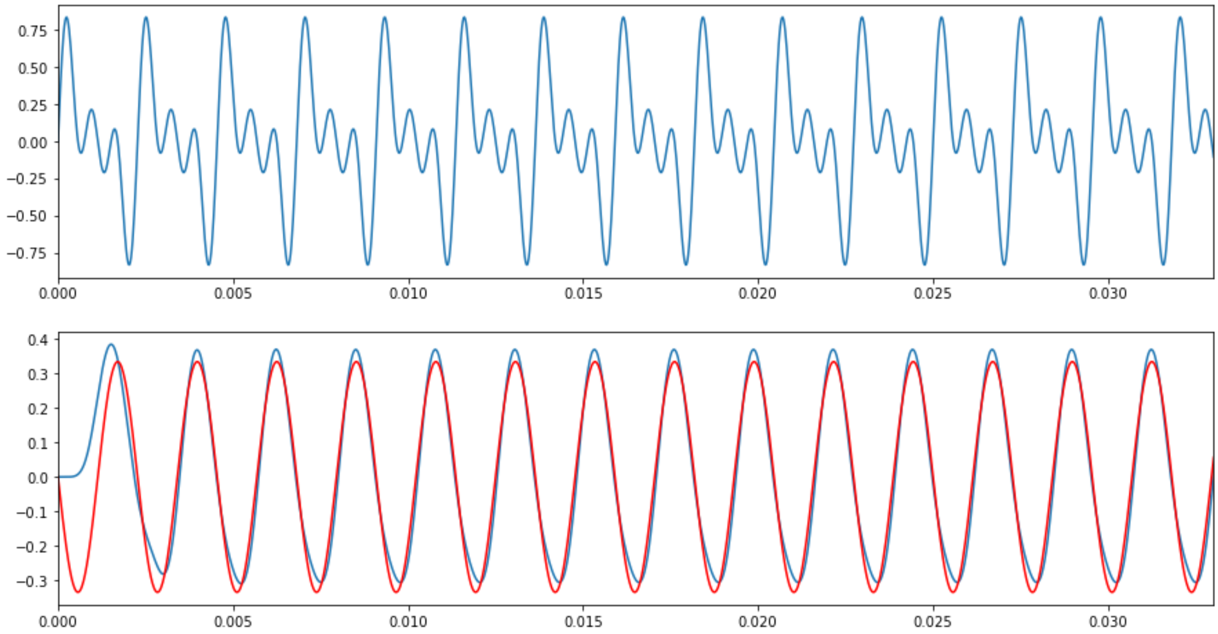
0:00 / 0:03

Out[33]:

0:00 / 0:03

Low-pass filtering with IIR operators





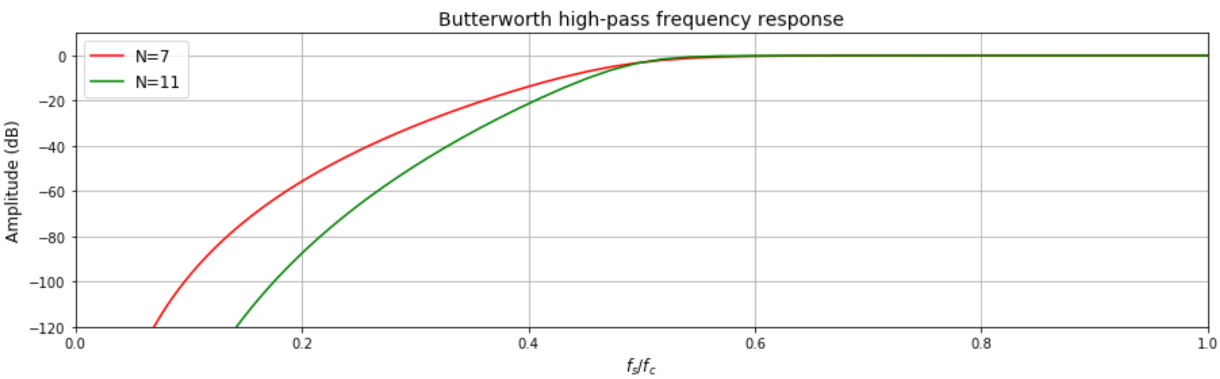
Out[331]:

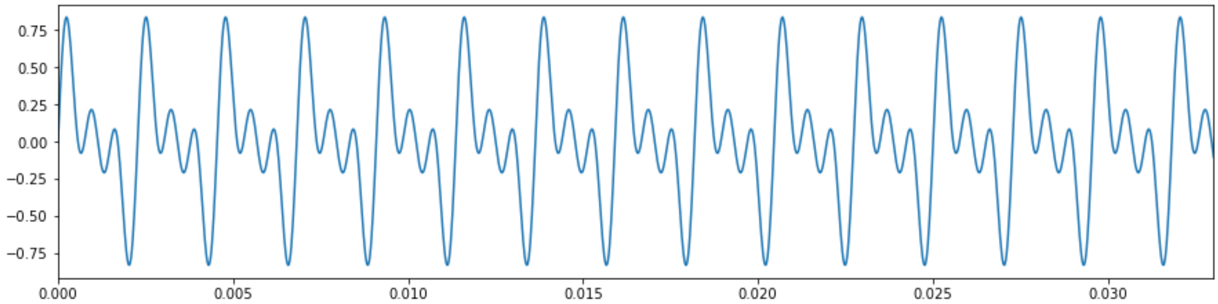
0:00 / 0:03

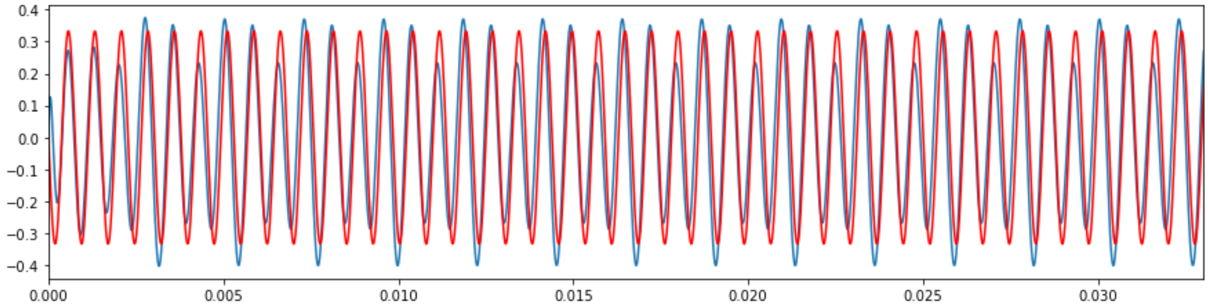
Out[332]:

0:00 / 0:03

High-pass filtering with IIR operator







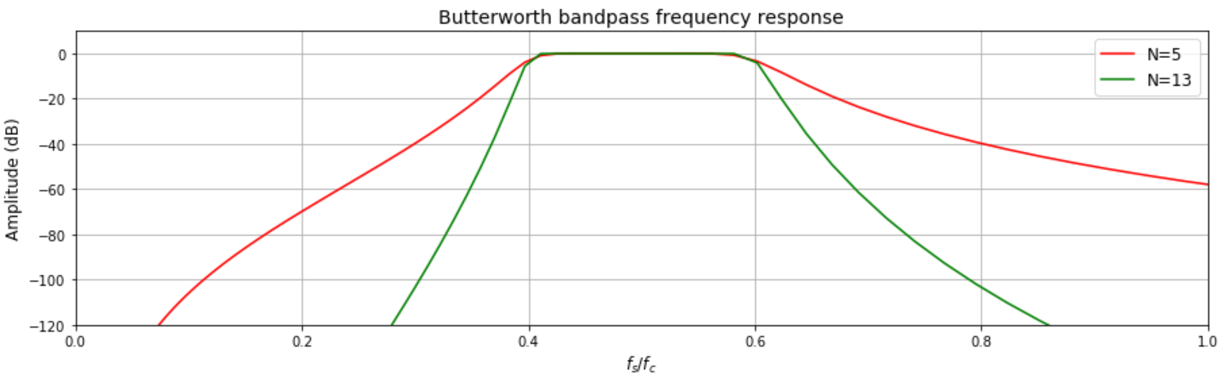
Out[333]:

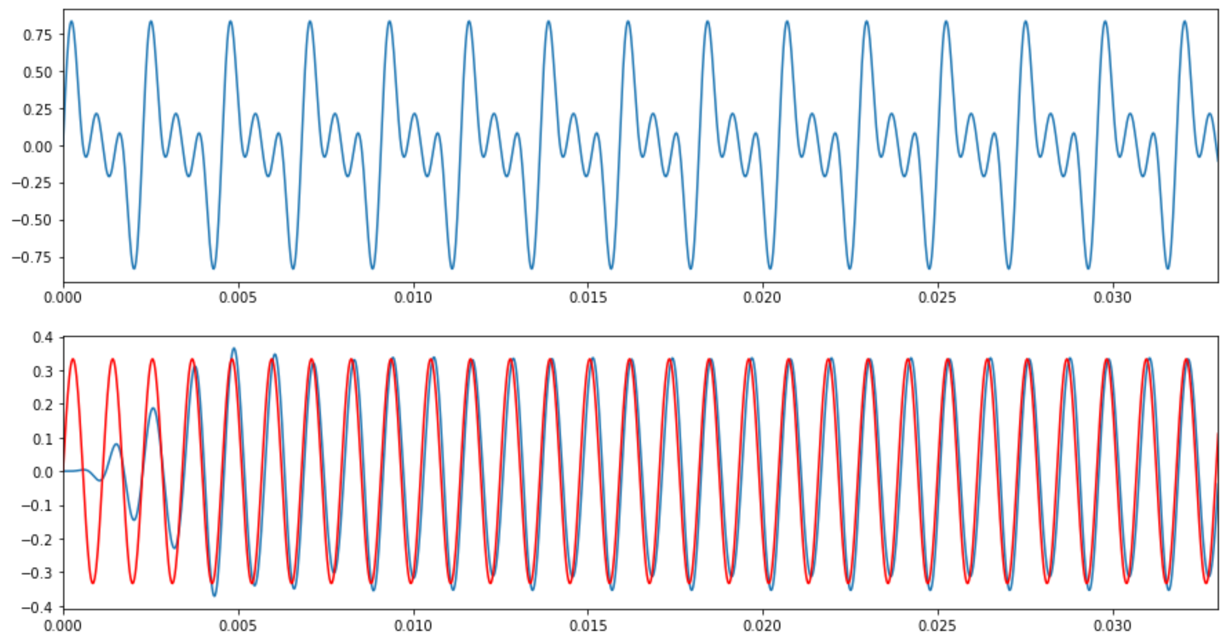
0:00 / 0:03

Out[334]:

0:00 / 0:03

Band-pass filtering with IIR operator





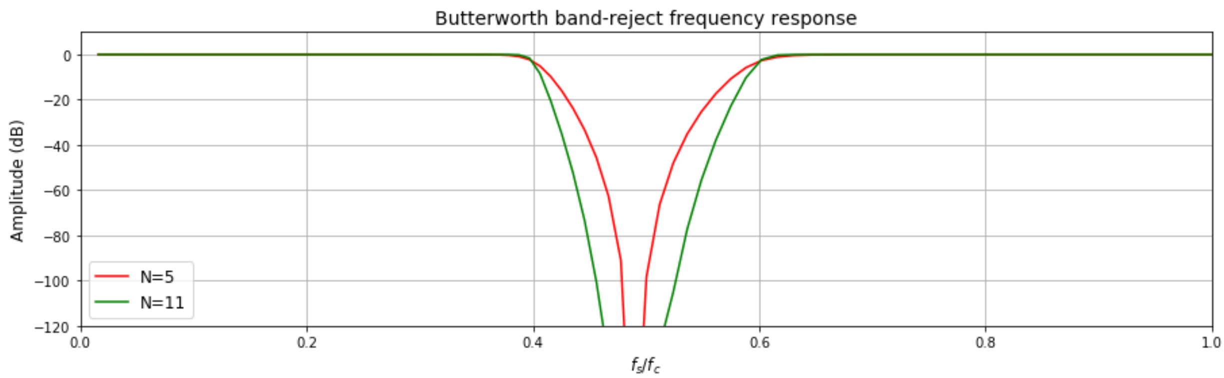
Out[335]:

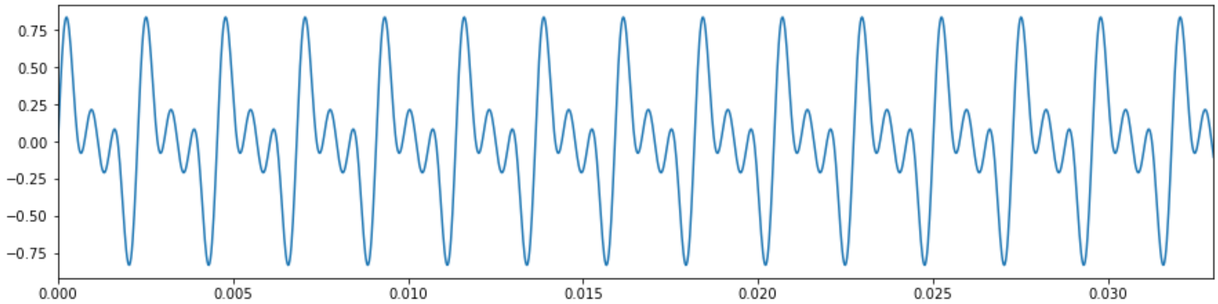
0:00 / 0:03

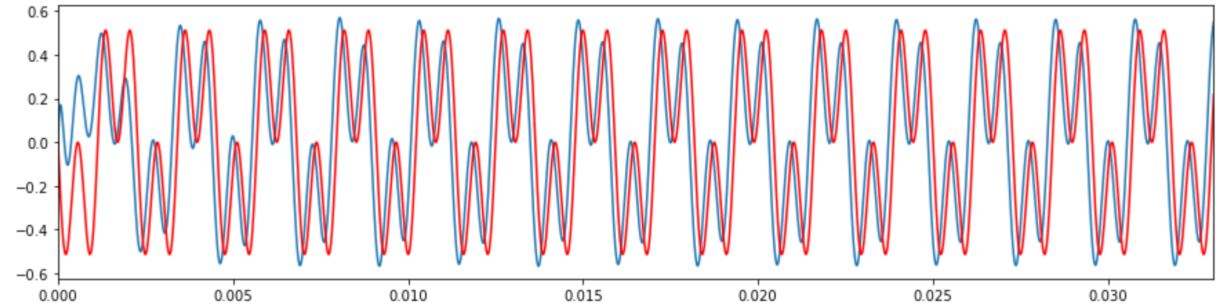
Out[336]:

0:00 / 0:03

Band-reject filtering with IIR operator







Out[338]:

0:00 / 0:03

Out[337]:

0:00 / 0:03