Simple Filter Design

Arnav Goyal - 251244778

March 25, 2024

Pole & Zero Placement

We are trying to design a filter to fit the following specifications:

- Gain of 0.9 - 1.1 within the 0 - 20 Hz band
- Gain of < 0.1 within the 180 - 200 Hz band

$$\omega_f = \frac{2\pi f}{F_s} \tag{1}$$

Given a sampling frequency of $F_s = 400$ Hz, we can find the equivalent frequencies of the ones listed above through the use of (1). This gives equivalent frequencies shown below. The format below is $f \to \omega$

- $0 \to 0$
- $20 \to \frac{\pi}{10}$
- $180 \to \frac{9\pi}{10}$
- $200 \to \pi$

To choose the conjugate pole locations, we can use the endpoints of the amplification band because we must put the poles near frequencies we want to *emphasize*, thus we need conjugate poles at $\pm\pi/10$. In other words ... we will need poles $p_{1,2}$ such that:

$$p_{1,2} = r \cdot e^{\pm j \frac{\pi}{10}}$$

This gives a degree of freedom with the *radial distance*, r. According to lecture slides, one possible pole/zero location for a low-pass filter is to have the two zeroes at the origin. This lets us use the numerator coefficient to control the DC-Gain (numerator coeff.) of the response, which is our second degree of freedom. Now to meet specifications I just experimented with different values of the radial distance and the DC-Gain until one set of values worked.

This set of values was $r = 0.65$ and numerator coeff = 0.195, and it produced the following magnitude response that meets specification. The pole-zero pattern, and magnitude-phase plots are also included here.

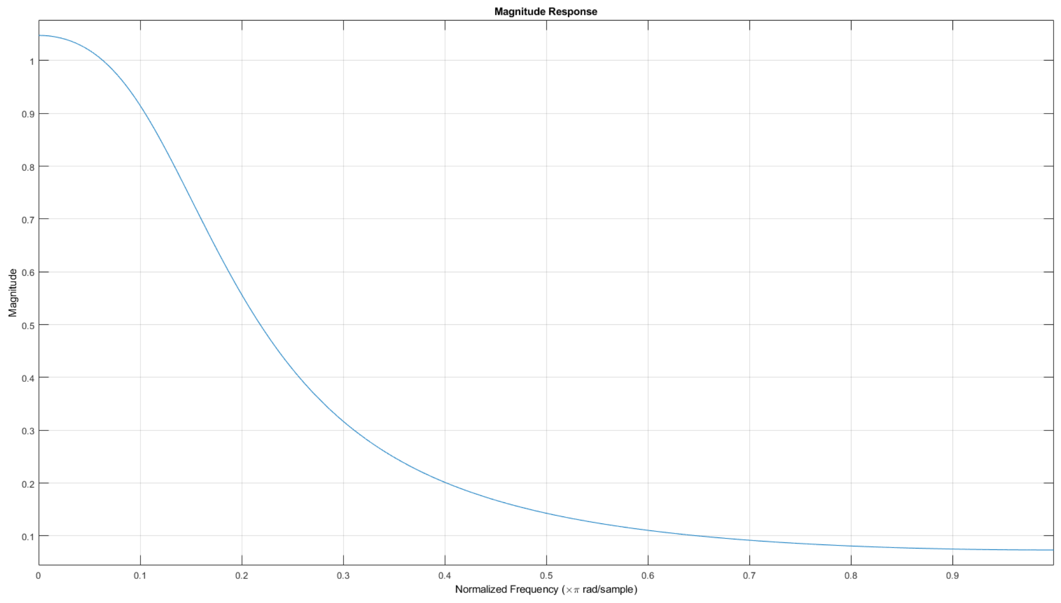

Figure 1: Plot of LPFs magnitude response

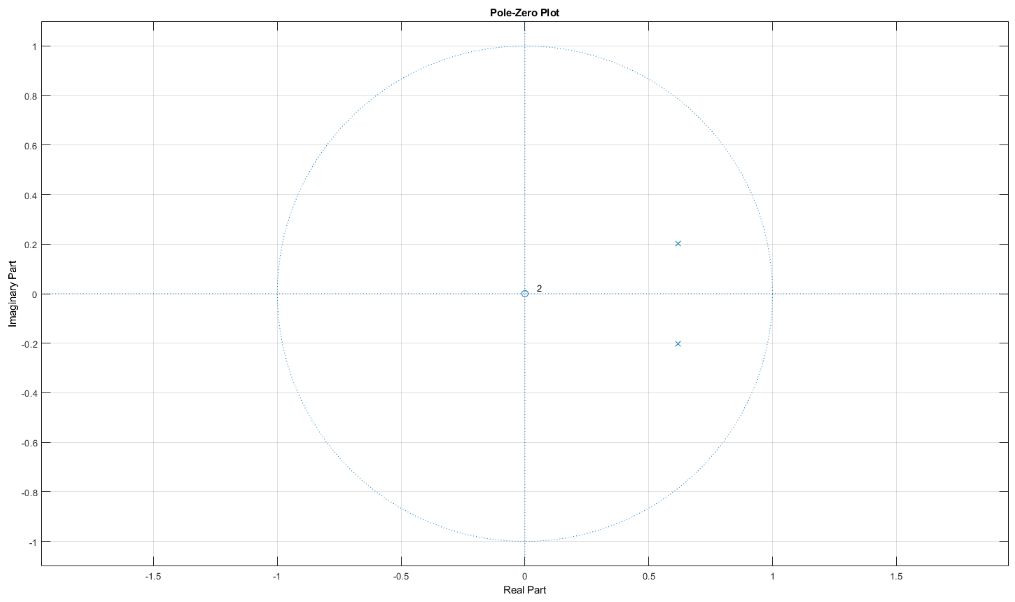

Figure 2: LPFs zero-pole plot

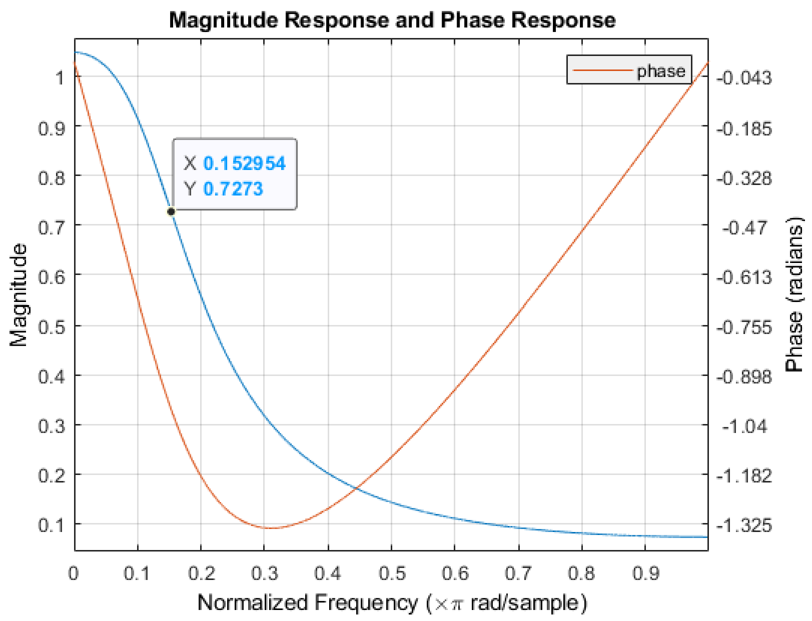

Figure 3: LPFs magnitude (blue) and phase (orange) plot

The script prints our the numerator and denominator coefficient lists, so it is quite easy to obtain the corresponding transfer function, and difference equation. The script provides the coefficient lists as:

- `num = [ 0.1950 ]`
- `den = [ 1 , -1.2364 , 0.4255 ]`

This means that we have the following transfer function:

$$H(z) = \frac{0.1950}{1 - 1.2364z^{-1} + 0.4255z^{-2}} = \frac{0.1950z^2}{z^2 - 1.2364z + 0.4255}$$

$$\frac{Y(z)}{X(z)} = \frac{0.1950}{1 - 1.2364z^{-1} + 0.4255z^{-2}}$$

And that gives the following difference equation:

$$Y - 1.2364Yz^{-1} + 0.4255Yz^{-2} = 0.1950X$$

$$y(n) - 1.2364y(n-1) + 0.4255y(n-2) = 0.1950x(n)$$

To prove it works I created a signal with two sinusoidal components, one at 20 Hz and one at 200 Hz, and sampled it at 400 Hz. The input and output are shown in Figure 4.

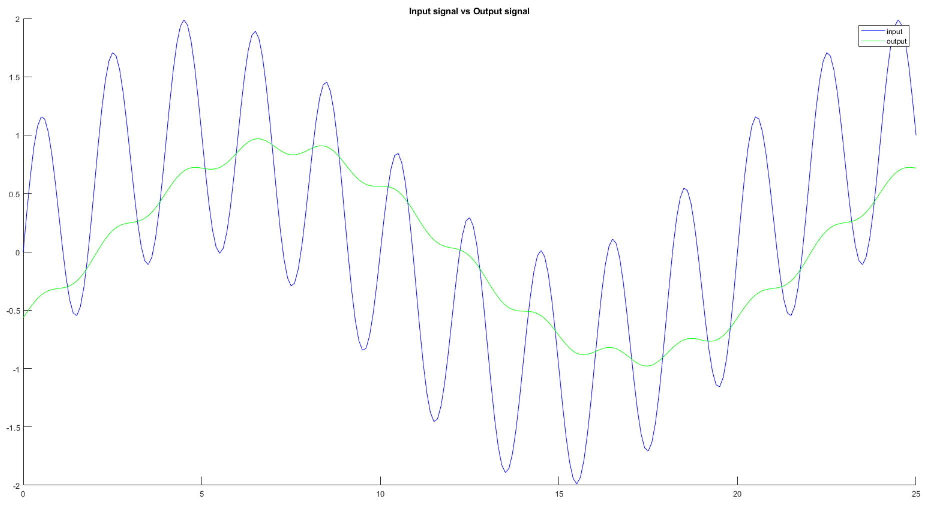

Figure 4: Filter Input vs Filter Output

MATLAB Code

Note: Please uncomment the fvtool line to see the fvtool output.

```
clear;
%% Obtain conj pole locations
r = 0.65; % radial distance
dc_gain = 0.195;

pang = pi/10; % pole angle
xp = r * cos(pang);
yp = r * sin(pang);


%% Low Pass Filter
num = dc_gain;
% expression to guarantee conj poles.
% derived very painfully :(
den = [1, -2*xp, xp^2 + yp^2];

disp(num);
disp(den);

% uncomment to see everything
% fvtool(num, den)

%% Implement the transfer function
figure;
hold on;
H = @(w) 0.1950 / (1 - 1.2364 * exp(-1i*w) + 0.4255 * exp(-2*1i*w));
% initial signal
t = 0 : 0.1 : 25;
x = sin(pi/10 * t) + sin(pi*t);
% filtered signal
y = abs(H(pi/10))*sin(pi/10 * t + angle(H(pi/10))) + abs(H(pi))*sin(pi*t + angle(H(pi)));
plot(t, x, 'blue');
plot(t, y, 'green');
legend('input', 'output');
title('Input␣signal␣vs␣Output␣signal');
```