Poles, Zeros, and Frequency Response

With the previous circuits, you can build filters with

- Real poles
- Complex Poles, and
- Zeros at s = 0

Filter design uses this to places poles and zeros to give a desired frequency response. In this lecture we look at how the poles and zeros affect the gain vs. frequency for a filter.

Analysis: Given a filter, find the gain vs. frequency.

This is actually really easy: just plug it into Matlab. For example, plot the gain vs. frequency for

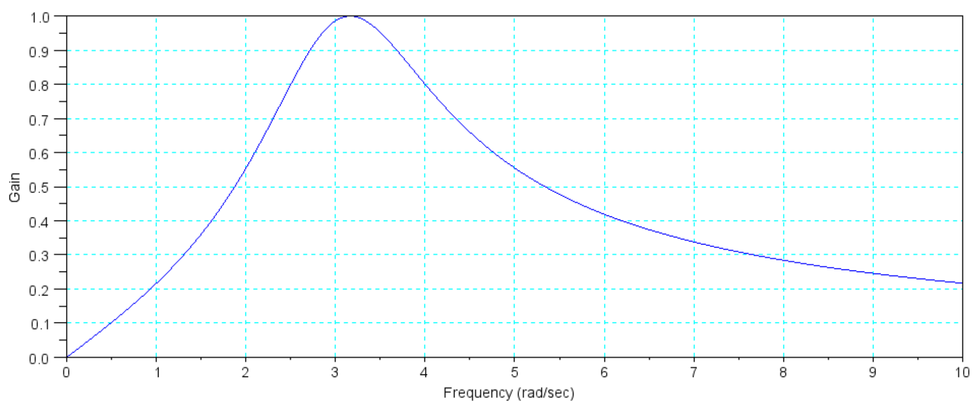
$$Y = \left(\frac{2s}{s^2+2s+10}\right)X$$

for

$$0 < \omega < 10 \text{ rad/sec}$$

In Matlab:

```
-->w = [0:0.01:10]';
-->s = j*w;
-->G = 2*s ./ (s.^2 + 2*s + 10);
-->plot(w,abs(G));
-->xlabel('Frequency (rad/sec)');
-->ylabel('Gain');
```



Design: Given a desired frequency response, design a filter to match (or come close) to your design.

This gets a lot trickier. To do this, let's first look at how poles and zeros affect the gain of a filter.

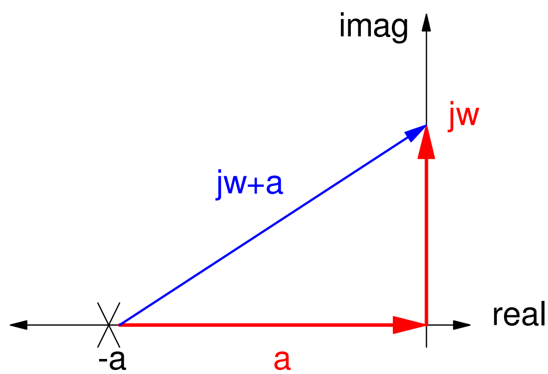
Let's start with about the simplest filter you can make:

$$Y = \left(\frac{1}{s+a}\right)X$$

The frequency response is obtained by letting $s \rightarrow j\omega$:

$$Y = \left(\frac{1}{j\omega+a}\right)X$$

Graphically, the gain is equal to the vector '1' divided by the vector '$j\omega + a$'. The latter term is equal to the vector from the pole at -a to the origin (a) plus the vector $j\omega$.



Since you're dividing by $j\omega + a$, the gain is

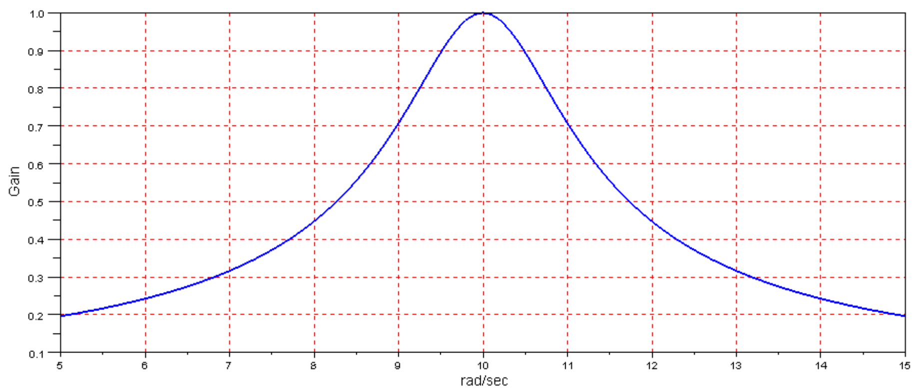
- A maximum when you're closest to the pole (i.e. at w = 0).
- Zero when you're far away from the pole (at infinity), and
- Down by $\sqrt{2}$ when the frequency is ja

Note that this also works for comples polex. If your filter has a pole at s = -1 + j10:

$$Y = \left(\frac{1}{s+1-j10}\right)X$$

then

- The gain will be a maximum at s = j10 (the closest point on the jw axis to the pole)
- The gain drops by $\sqrt{2}$ when at s = j9 and s = j11 (when you are 1 rad/sec away from the max gain point. 1 is the real part of the pole)



A generalized filter will look something like

$$Y = k\left(\frac{(s+z_1)(s+z_2)}{(s+p_1)(s+p_2)(s+p_3)}\right)X$$

where

- zi are the zeros,
- pi are the poles, and
- k is a constant gain

The graphical interpriation for this filter is

$$\boxed{gain = k \cdot \frac{\Pi(\text{distance from jw to the zeros})}{\Pi(\text{distance from jw to the poles})}}$$

Note that

- If you're close to a zero, the gain is small (multiply by a small number)
- If you're close to a pole, the gain is large (divide by a small number)

So, a design strategy could be

- **Place zeros near frequencies you want to reject**
- **Place poles near frequencies you want to pass.**

To do this, the Matlab *fminsearch()* can be useful.

Problem: Design a filter to approximate an ideal low-pass filter with a gain of

$$G_{ideal}(s) \approx \begin{cases} 1 & \omega < 4 \\ 0 & otherwise \end{cases}$$

With *fminsearch()*, you first assume the form of the filter. Let's start by assuming a 4th-order filter with real poles:

$$G(s) = \left(\frac{a}{(s+b)(s+c)(s+d)(s+e)}\right)$$

To determine how good this filter, define a cost function to be the sum-squared difference in the two filters

$$E(j\omega) = |G_{ideal}(j\omega)| - |G(j\omega)|$$

$$J = \int_0^{10} E^2(j\omega) \cdot d\omega$$

where the integration range is somewhat arbitrary (it covers the pass band and some of the reject region)

A Matlab m-file to do this is

```
function [ J ] = costf( z )
 a = z(1);
 b = z(2);
 c = z(3);
 d = z(4);
 e = z(5);

 w = [0:0.01:10]';
 s = j*w;
 Gideal = 1 .* (w < 4);

 G = a ./ ( (s+b) .* (s+c) .* (s+d) .* (s+e) );

 E = abs(Gideal) - abs(G);

 J = sum(E .^ 2);

 plot(w,abs(Gideal),w,abs(G));
 pause(0.01);

end
```

If you make your initial guess

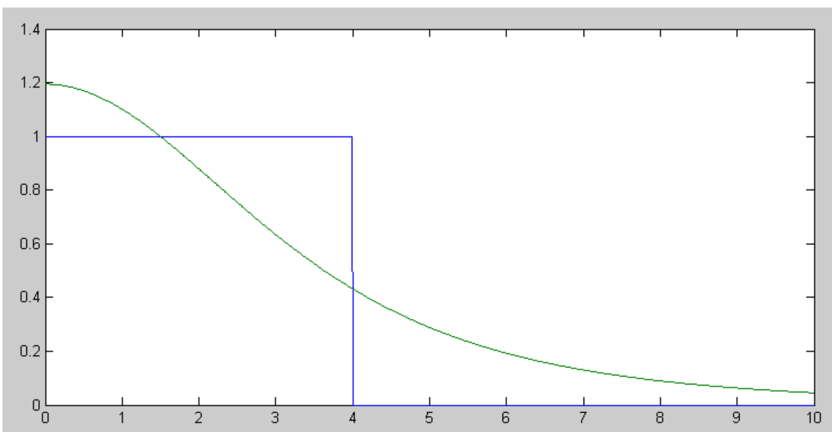
$$G(s) = \left(\frac{100}{(s+2)(s+3)(s+4)(s+5)}\right)$$

then

```
>> J = costf([100,2,3,4,5])

J =   145.8354
```

If you let *fminsearch()* guess and guess to minimize J you get

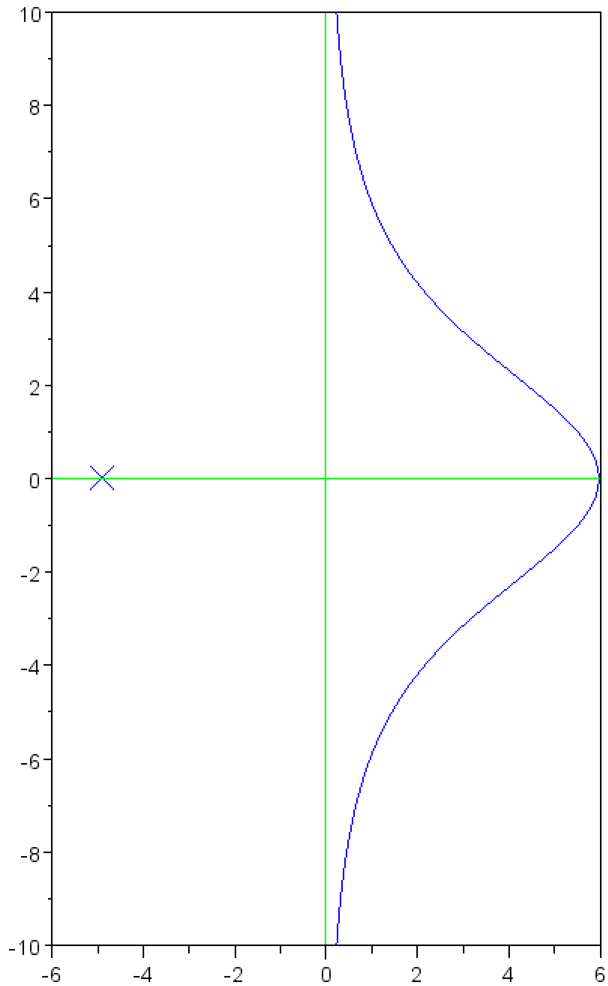
```
>> [a,b] = fminsearch('costf',[100,2,3,4,5])

a =

  697.8575    4.9165    4.9165    4.9165    4.9165


b =

   55.3564
```



The best Matlab could do if you constrain it to have real poles is to place the four poles at

$$G(s) = \left(\frac{697}{(s+4.916)^4}\right)$$

If you plot the pole location on the s-plane along with the gain (drawn sideways so that the y-axis is jw or frequency), it looks like this:



Pole Location in the s-plane along with the gain vs. frequency drawn sideways

Note that

- There are four poles at s = -4.91.
- The gain is large when you're close to the pole
- The grain drops as you move away from the pole (drops as 1 / distance ^{4})

It's also not a very good filter: you can't do much with just real poles.

Complex Poles: Instead, let G(s) be of the form

$$G(s) = \left(\frac{a}{s^4 + bs^3 + cs^2 + ds + e} \right)$$

An m-file for this filter is

```
function [ J ] = costf( z )
 a = z(1);
 b = z(2);
 c = z(3);
 d = z(4);
 e = z(5);

 w = [0:0.01:10]';
 s = j*w;
 Gideal = 1 .* (w < 4);

 G = a ./  (s.^4 + b*s.^3 + c*s.^2 + d*s + e );

 E = abs(Gideal) - abs(G);

 J = sum(E .^ 2);

 plot(w,abs(Gideal),w,abs(G));
 pause(0.01);

end
```

Minimizing the cost:

```
>> [a,b] = fminsearch('costf',10*rand(1,5))

a =

   36.6716   -1.3547   13.7226  -24.3743   39.3082


b =

   13.0720
```

meaning

$$G(s) = \left(\frac{36.67}{s^4 + 1.3547s^3 + 13.7226s^2 + 24.3743s + 39.3} \right)$$

The roots of the denominator are:

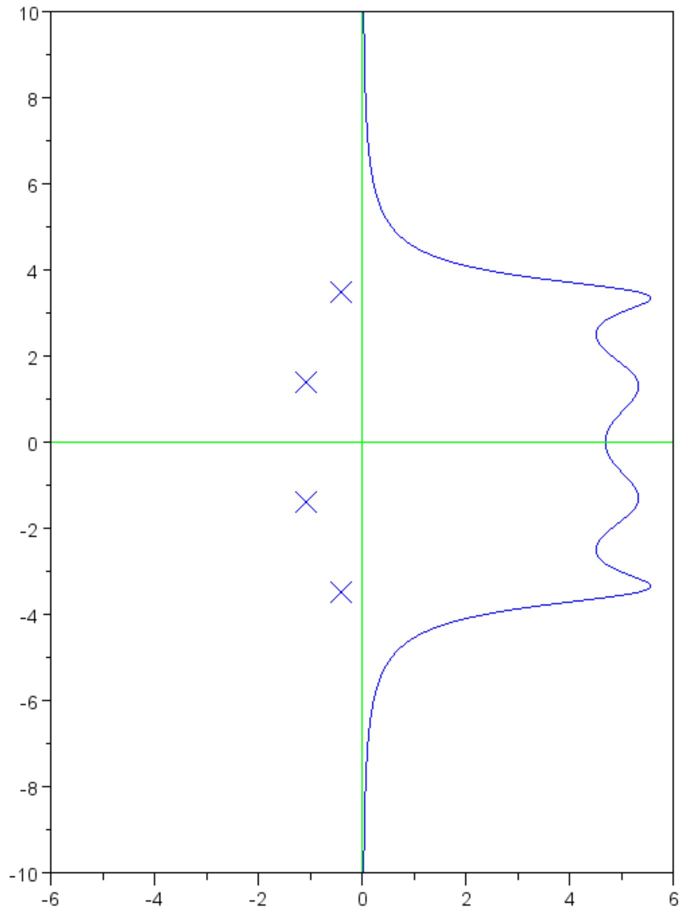
```
>> roots([1,a(2:5)])

ans =
```

```
  0.4157 + 3.4910i
  0.4157 - 3.4910i
 -1.0930 + 1.4091i
 -1.0930 - 1.4091i
```

which is unstable. Reflecting the unstable poles to the left-half plane gives

$$G(s) = \left(\frac{36.67}{(s+0.41 \pm j3.49)(s+1.09 \pm j1.41)} \right)$$

The gain vs. frequency and pole location looks like:



Note that to design an 'optimal' low-pass filter with 4 poles,

- You place the 4 poles in the pass-band region (-4 to +4 rad/sec)
- Close to the jw axis
- Spread out so that as you go from -j4 to +j4, you're always close to a pole (and the gain is large)
- As you move past 4 rad/sec, the distance to the poles increases, resulting in the gain dropping.

You can to a lot better if you're allowed to use complex poles.

Just for fun, try one more filter of the form

$$G(s) = \left(\frac{a \cdot c \cdot e}{(s+a)(s^2+bs+c)(s^2+ds+e)}\right)$$

This has five poles along with a DC gain of one:

```
function [ J ] = costf( z )
 a = z(1);
 b = z(2);
 c = z(3);
 d = z(4);
 e = z(5);

 w = [0:0.01:10]';
 s = j*w;
 Gideal = 1 .* (w < 4);

 G = a*c*e ./  ( (s+a) .* (s.^2 + b*s + c) .* (s.^2 + d*s + e ) );

 G = abs(G);
 G2 = max(0, G-1);

 E = abs(Gideal) - abs(G);

 J = sum(E .^ 2) + 10 * sum(G2.^2);

 plot(w,abs(Gideal),w,abs(G));
 pause(0.01);

end
```

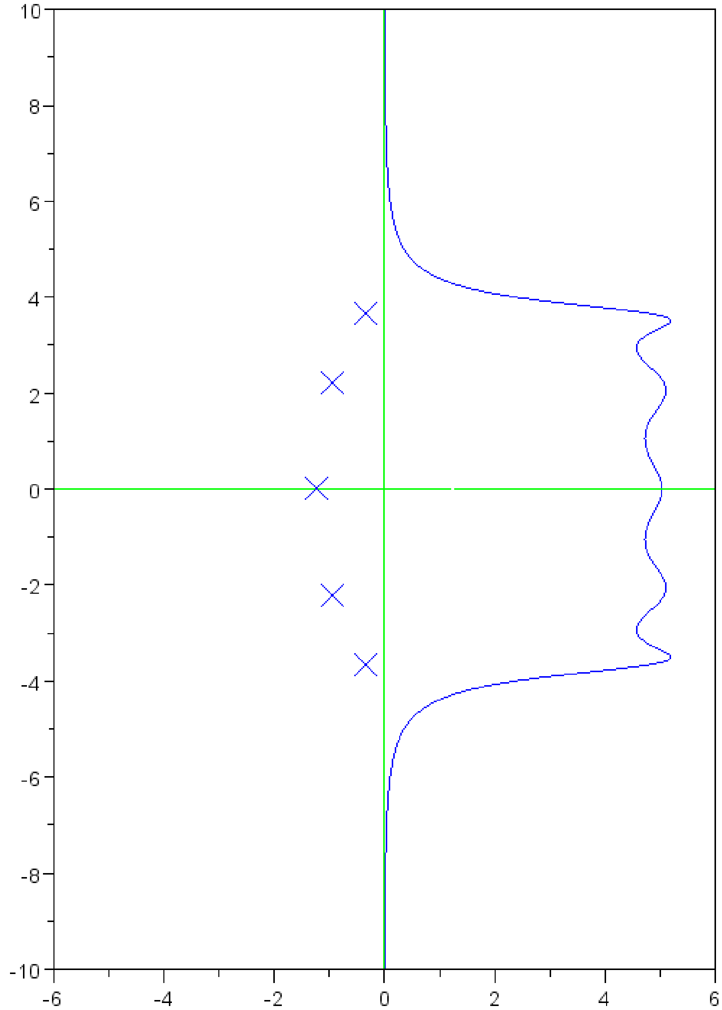
Running in Matlab:

```
>> [a,b] = fminsearch('costf',10*rand(1,5))

a =

    1.2226    0.6761   13.5006    1.8855    5.7318


b =

    9.6110
```

meaning

$$G(s) = \left(\frac{96.4}{(s+1.222)(s^2+0.6761s+13.5)(s^2+1.88s+5.73)}\right)$$

Plotting the pole location vs. gain like before:



Pole Location & Gain (drawn sidweways)

Note that there is definately a pattern here:

- You scatter N poles in the pass-band
- It appears the poles are placed on an ellipse - farthest away from the jw axis at w=0 and closer as you move away from w = 0