Filters in the s-Plane

Transfer Functions and LaPlace Impedances

In essence, any circuit with an inductor or a capacitor is a filter. Each of these requires a differential equation to describe the circuit due to the VI relationships being

$$V = L\frac{dI}{dt}$$

for an inductor, or

$$I = C\frac{dV}{dt}$$

for a capacitor. If you assume all functions are in the form of

$$y = e^{st}$$

(which is the basic assumption behind LaPlace transforms), then differentiation becomes multiplication by 's'

$$\frac{dy}{dt} = e \cdot e^{st} = sy$$

The transfer function of a circuit is short-hand notation for the differential equation relating the input and output. For example, if a circuit is described by the following differential equation

$$\frac{d^3y}{dt^3} + 7\frac{d^2y}{dt^2} + 9\frac{dy}{dt} + 15y = 10\frac{dx}{dt} + 3x$$

in LaPlace notation, this becomes

$$s^3Y + 7s^2Y + 9sY + 15Y = 10sX + 3X$$

or

$$Y = \left(\frac{10s+3}{s^3+7s^2+9s+15}\right)X$$

The gain from X to Y is called the *transfer function*

$$G(s) = \left(\frac{10s+3}{s^3+7s^2+9s+15}\right)$$

Note that this goes either way:

- Given a differential equation, you can find the transfer function by replacing each derivative with 's'
- Given the transfer function, you can find the differential equation by cross-multiplying and replacing each 's' with $\frac{d}{dt}$

Example: Find the differential equation relating X and Y

$$Y = \left(\frac{10s+3}{s^3+7s^2+9s+15}\right)X$$

Solution: Cross multiply

$$(s^3+7s^2+9s+15)Y=(10s+3)X$$

Replace each 's' with $\frac{d}{dt}$

$$\frac{d^3y}{dt^3}+7\frac{d^2y}{dt^2}+9\frac{dy}{dt}+15y=10\frac{dx}{dt}+3x$$

Analyzing Filtes for Sinusoidal Inputs

The transfer function defines the relationship between X and Y for all 's'. If x(t) is a sinusoid, such as

$$x(t)=a\cdot\cos(\omega t)$$

then all you care about is the gain at one particular 's'. From Euler's identity

$$\cos(\omega t)=\tfrac{1}{2}(e^{j\omega t}+e^{-j\omega t})$$

Since LaPlace transforms assume that all functions are in the form of

$$y(t)=e^{st}$$

this means that x(t) only exists at s = $j\omega$ and $s=-j\omega$. Likewise, you only care about the gain at these two values of 's'. Actually, you only need to analyze one of these: the other will just be the complex conjugate

Example: Find y(t) assuming

$$Y=\left(\frac{10s+3}{s^3+7s^2+9s+15}\right)X$$

and

$$x(t)=3\cos(4t)$$

Solution: x(t) is zero everywhere except for when $s=\pm j4$. So, analyze G(s) at s = j4

$$\left(\frac{10s+3}{s^3+7s^2+9s+15}\right)_{s=j4}=0.3973\angle-110^0$$

From output is gain times input:

$$y(t)=(0.3973\angle-110^0)\cdot 3\cos(4t)$$

$$y(t)=1.1919\cos(4t-110^0)$$

When you analyze G(s) at $s=j\omega$, the answer will be a complex number:

- The amplitude is the gain: how much larger the output is than the input
- The angle is the phase shift: how much the output is shifted in time

If you have multiple inputs, you can use superposition to analyze each frequency separately.

Example: Determine y(t) assuming

$$Y = \left(\frac{10s+3}{s^3+7s^2+9s+15}\right) X$$

$$x(t) = 3\cos(4t) + 5\cos(60t)$$

Solution: Treat this as two separate problems:

x(t) = 3 cos(4t)

s = j4

$$\left(\frac{10s+3}{s^3+7s^2+9s+15}\right)_{s=j4} = (0.3973\angle -110^0)$$

$$y_1(t) = (0.3973\angle -110^0)\cdot 3\cos(4t)$$

$$y_1(t) = 1.1919\cos(4t - 110^0)$$

x(t) = 5 cos(60t)

s = j60

$$\left(\frac{10s+3}{s^3+7s^2+9s+15}\right)_{s=j60} = 0.0028\angle -173^0$$

$$y_2(t) = (0.0028\angle -173^0)\cdot 5\cos(60t)$$

$$y_2(t) = 0.0139\cos(60t - 173^0)$$

The total answer is then

$$y(t) = y_1 + y_2$$

$$y(t) = 1.1919\cos(4t - 110^0) + 0.0139\cos(60t - 173^0)$$

Analysis of Filters: Bode Plots

Analyzing filters for sinusoidal inputs is fairly easy: just plug $s = j\omega$. If you want to look at how a filter behaves over a range of frequencies, you can do this in MATLAB as follows

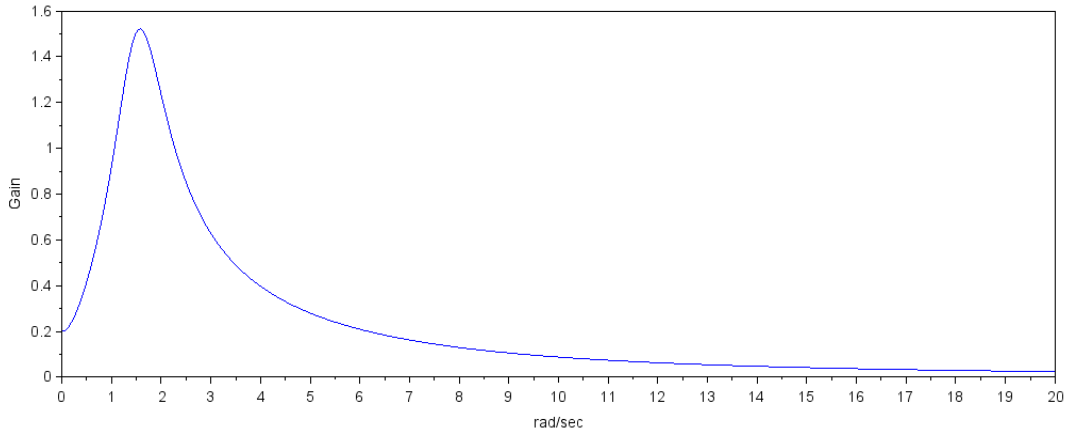
Example: Plot the gain vs. frequency for the following filter from 0 to 20 rad/sec

$$G(s) = \left(\frac{10s+3}{s^3+7s^2+9s+15}\right)$$

Matlab Code:

```
-->w = [0:0.01:20]';
-->s = j*w;
-->G = (10*s + 3) ./ (s.^3 + 7*(s.^2) + 9*s + 15);
-->plot(w,abs(G));
```

```
-->xlabel('rad/sec');
-->ylabel('Gain');
```



Gain vs. Frequency for G(s)

Note that the plot (termed a Bode plot) is a good way to describe how this filter behaves.

- Frequencies near 2 rad/sec are passed with a gain as much as 1.5
- Frequencies above 10 rad/sec are rejected, with a gain less than 0.2

Filter Design: Poles and Zeros

While analyzing a given filter is easy and straight-forward, designing a filter is a little more tricky. One way to help see how the transfer function relates to the gain of a filter is to look at the filter's poles and zeros.

In general, G(s) will have a numerator and a denominator polynomial

$$G(s) = k\left(\frac{z(s)}{p(s)}\right)$$

- The zeros are the roots of the numerator polynomial
- The poles are the roots of the denominator polynomial.

Graphically, the vector $(s+5)$ is equal to the vector from -5 to the point s. This means another way to interpret the gain of a filter is

$$G(s) = k \cdot \frac{\Pi(\text{distance from the zeros to j}\omega)}{\Pi(\text{distance from the poles to j}\omega)}$$

or, in other words

- **Place zeros near frequencies where you want the gain to be small (multiply by a small number)**
- **Place poles near frequencies where you want the gain to be large (divide by a small number)**

Types of Filters

Filters are categorized into different types:

Filter Type	Characteristic	Example
Low-Pass	Low-frequency gain is large (pass) High-frequency gain is small (reject)	$\left(\frac{10}{s+10}\right)$
High-Pass	High-frequency gain is large (pass) Low-frequency gain is smalle (reject)	$\left(\frac{10s}{s+10}\right)$
Band-Pass	High-frequency gain is small Low frequency gain is small Mid-range frequency is large	$\left(\frac{2s}{(s+1+j50)(s+1-j50)}\right)$

A filter's order is the number of poles the filter has. In general, the more poles a filter has, the better the filter.

Certain pole and zero locations also have special names as well. These have certain characteristics:

RC Filter:

An n-pole RC filter has all n-poles on the real axis.

- It's advantage is you can build it with a passive RC filter (good)
- It's problem is it's a pretty poor filter.

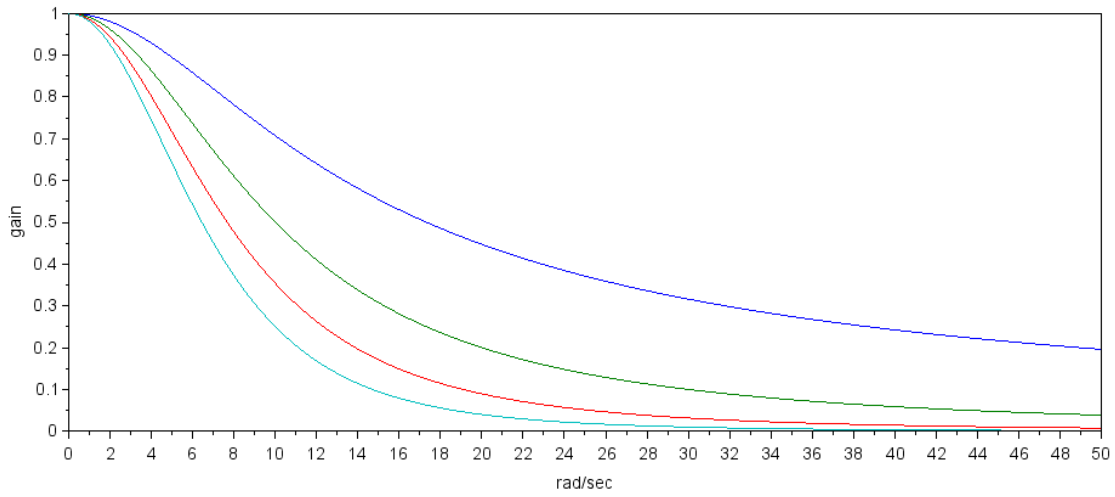
For example, the gain of

$$G(s) = \left(\frac{10}{s+10}\right)^n$$

for n=1, 2, 3, and 4 is shown below. Note that

- As n increases, the high-frequency gain gets smaller and smaller (good)
- However, the gain below 5 rad/sec starts to droop more an more (bad)

```
-->w = [0:0.1:50]';
-->s = j*w;
-->G = 10 ./ (s+10);
-->plot(w,abs([G,G.^2,G.^3,G.^4]));
-->xlabel('rad/sec');
-->ylabel('gain');
```



Gain of and RC filter, $\left(\frac{10}{s+10}\right)^n$, for n=1 (blue) to n=4 (cyan)

Butterworth Filter

If you use complex poles, you can do better. For example, take the case of n=5.

A 3rd-order RC filter with a corner at 10 rad/sec is

$$G = \left(\frac{10}{s+10}\right)^5$$

One of these poles has to be real. The other four, however, could be moved along the circle centered at the origin with a radius of 10. As you increase the angle of these poles, the gain at j10 increases. If you go too far, the gain at j10 starts to go above one.

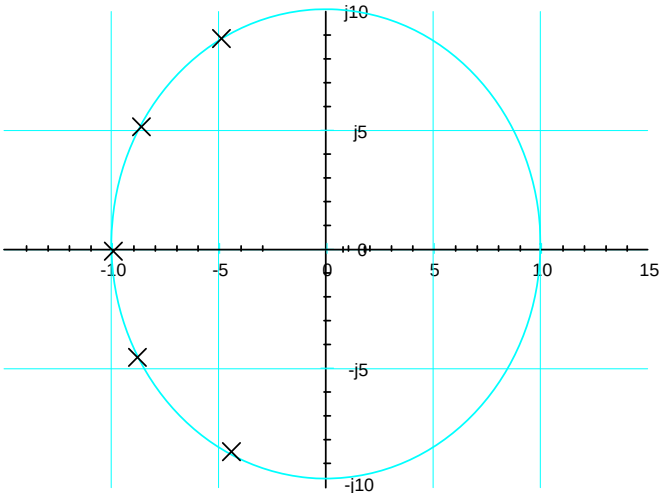
A Butterworth filter is the farthest you can slide the poles while keeping the maximum gain less than one

It turns out, the formula for an n-th order Butterworth filter is fairly easy:

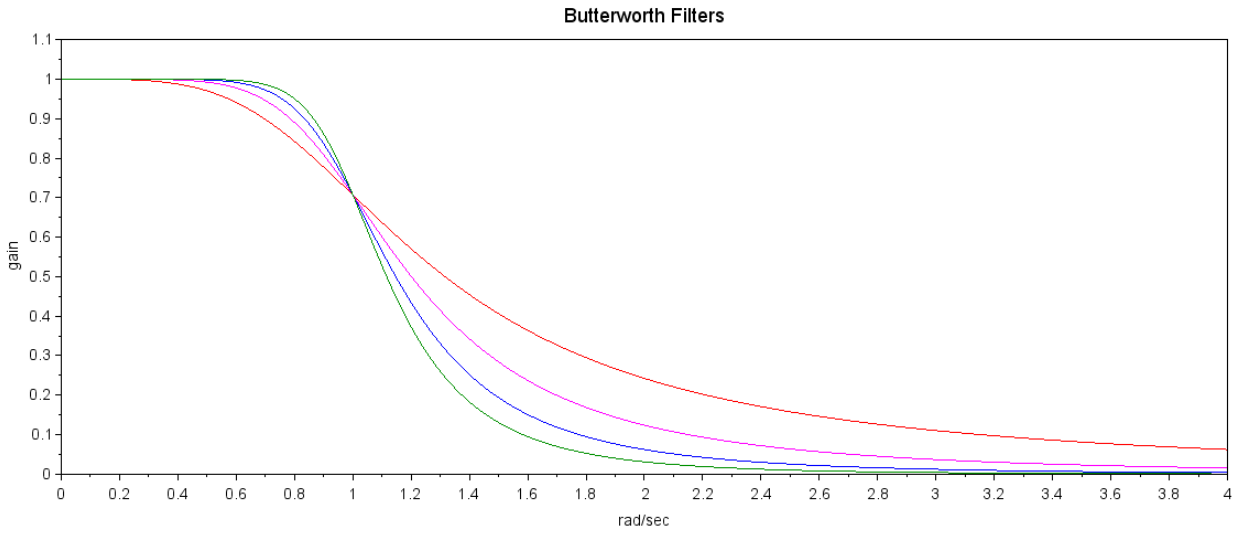
- The magnitude of the poles is equal to the corner frequency
- The angle between poles is $\left(\frac{180^0}{n}\right)$

For example, the location of a Butterworth filter with a corner at 1 rad/sec is given below:

	N=2	N=3	N=4	N=5	N=6
zeros	none	none	none	none	none
poles	$-1\angle\pm 45^0$	-1 $-1\angle\pm 60^0$	$-1\angle\pm 22.5^0$ $-1\angle\pm 67.5^0$	-1 $-1\angle\pm 36^0$ $-1\angle\pm 72^0$	$-1\angle\pm 15^0$ $-1\angle\pm 45^0$ $-1\angle\pm 75^0$



Pole Location for a 5th-Order Butteworth Filter with a Corner at 10 rad/sec



Gain of a Butterworth Filter for n=2 (red), 3 (magenta), 4 (blue), 5 (green)

Note that with a Butteworth filter,

- The more poles you have the closer it gets to an ideal low-pass filter.
- All poles have the same amplitude
- All poles have an equal spacing between them

Chebychev Filter

If you can tolerate a gain larger than 1.00, you can do even better. These filters are called *Chebychev Filters*.

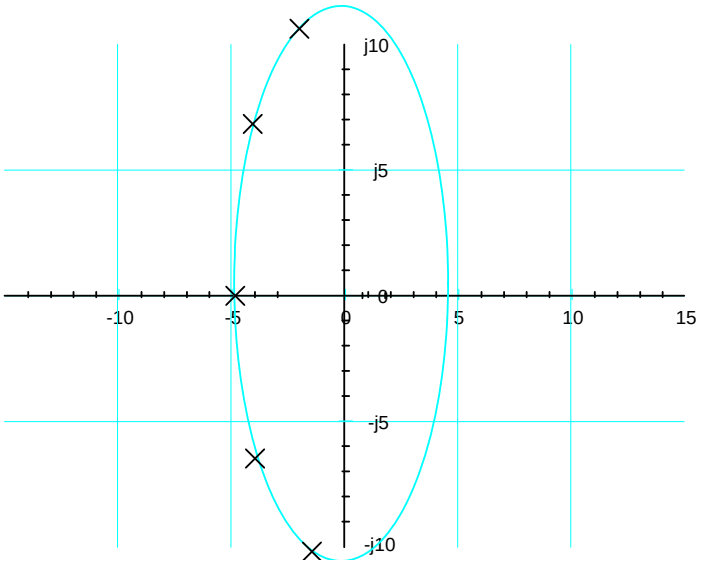
Unlike Butteworth filters, there are an infinite number of Chebychev filters - each depending upon how much above 1.000 you allow the gain to reach.

The net result is a Chebychev filter is like a Butterworth filter, only

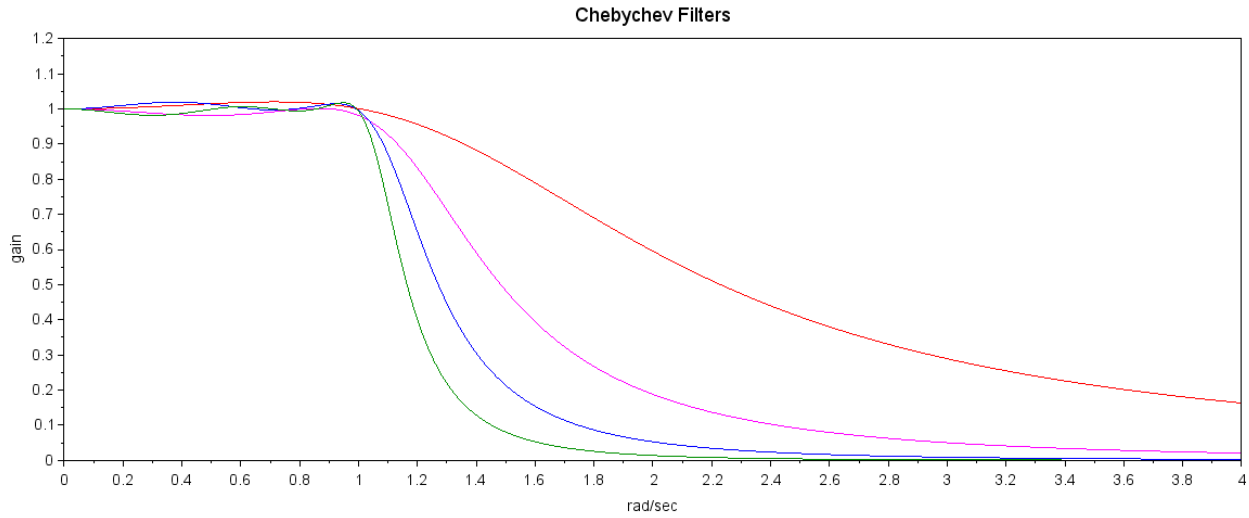
- The poles are located on an squashed circle which is stretched out past the bandwidth, and
- The poles are stretched further away.

From SciLab, the location of the poles for a Chebychev filter with a 0.2 ripple are given below.

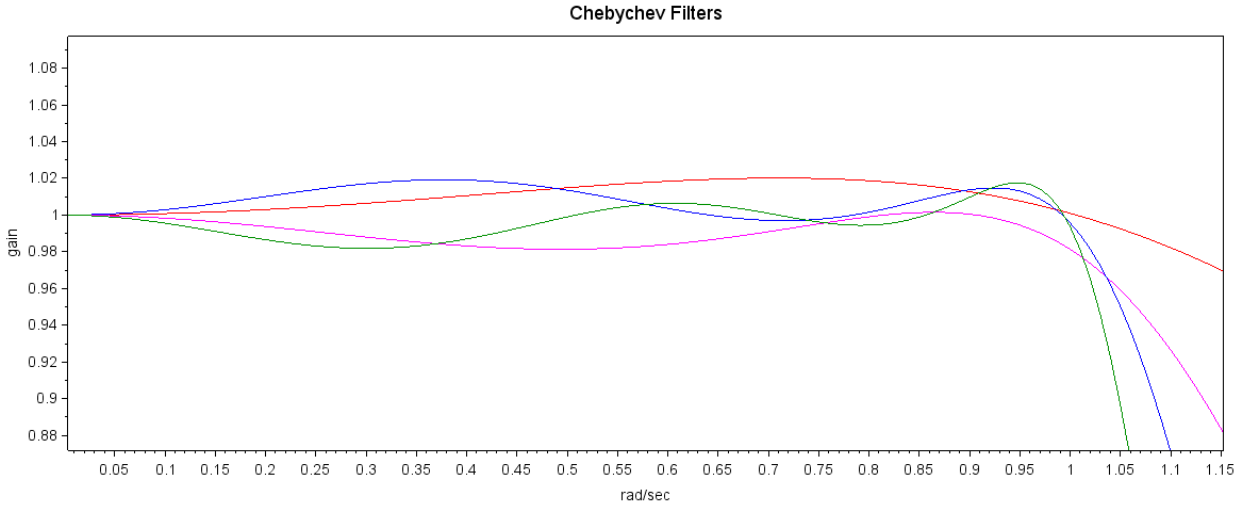
	N=2	N=3	N=4	N=5	N=6
zeros	none	none	none	none	none
poles	$-1.60\angle\pm 50.7^0$	-0.85 $-1.21\angle\pm 69.5^0$	$-0.72\angle\pm 38.5^0$ $-1.11\angle\pm 77.8^0$	-0.48 $-0.76\angle\pm 59.3^0$ $-1.06\angle\pm 82.0^0$	$-0.47\angle\pm 36.1^0$ $-0.81\angle\pm 69.8^0$ $-1.04\angle\pm 84.4^0$



Pole Location for a 5th-Order Chebychev Filter with a Corner at 10 rad/sec



Gain of a Type-1 Chebychev Filter with 0.2 Ripple for n=2 (red), 3 (magenta), 4 (blue), 5 (green)



Gain of a Type-1 Chebychev Filter with 0.2 Ripple for n=2 (red), 3 (magenta), 4 (blue), 5 (green)

fminsearch()

Another way to design filters is to use the function *fminsearch* in MATLAB. This routine finds the minimum of a function.

For example, suppose you want to find the square root of two. First, write an m-file in MATLAB

```
% function cost.m

function y=cost(z)

e = z*z - 2;
y = e*e;
end
```

The minimum of this function will be zero when z = 1.414. You can iterate in MATLAB to find the square root of two

```
>> cost(4)

   196

>> cost(3)

    49

>> cost(2)

     4

>> cost(1.414)

  3.6482e-007
```

You can also find this with *fminsearch*

```
>> [a,b] = fminsearch('cost',4)

a =

    1.4143


b =

  1.5665e-008
```

fminsearch iterates to find the minimum of the function called *cost*. The solution it found was 1.4143 which resulted in an error of 1.566e-8

Problem: Determine a filter of the form

$$G(s) = \left(\frac{as^2+bs+c}{s^3+ds^2+es+c} \right)$$

which has a gain vs frequency equal to

- G(s) = 1 for frequencies less than 2 rad/sec, and
- G(s) = 0 for frequencies above 2 rad/sec.

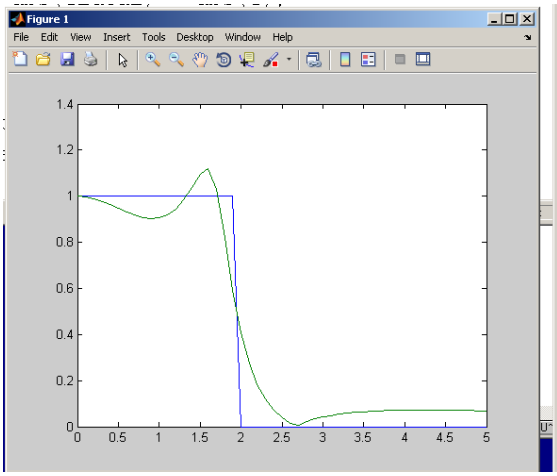
Step 1: Create an m-file where you pass it five numbers (a, b, c, d, e) and it returns how 'good' this filter is. Make the minimum when it is equal to the ideal filter.

To write this function,

- Compute G(s) for frequencies between 0 and 10 rad/sec.
- Calculate the difference between the gain of G(s) and the ideal gain
- Define the cost (goodness of the filter) to be the sum-squared error

```
function y=cost(z)

a = z(1);
b = z(2);
c = z(3);
d = z(4);
e = z(5);

w = [0:0.1:5]';
s = j*w;
Gideal = 1*(w<2);

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

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

y = sum(e .^ 2 );

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

The last two lines are just for fun: they plot the current filter vs. the ideal filter so you can see how things are progressing.




```
>> cost([0,0,1,3,3])

ans =

    7.5926

>> Z = fminsearch('cost',[0,0,1,3,3]);
>> Z

Z =

    0.4371   -0.0000    3.1036    1.5736    3.5070

>>
```

The 'best' filter is thus

$$G(s) = \left(\frac{0.4371s^2 + 3.1036}{s^3 + 1.5736s^2 + 3.5070s + 3.1036} \right)$$