Filters: Analysis and Design

Background

A filter is a circuit whose gain is a function of frequency. Essentially,

- Any circuit with inductors and/or capacitors is a filter.
- Any circuit which satisfies a differential equation is a filter.
- Any circuit where the input and output relationship is described by a transfer function is a filter.

Analysis

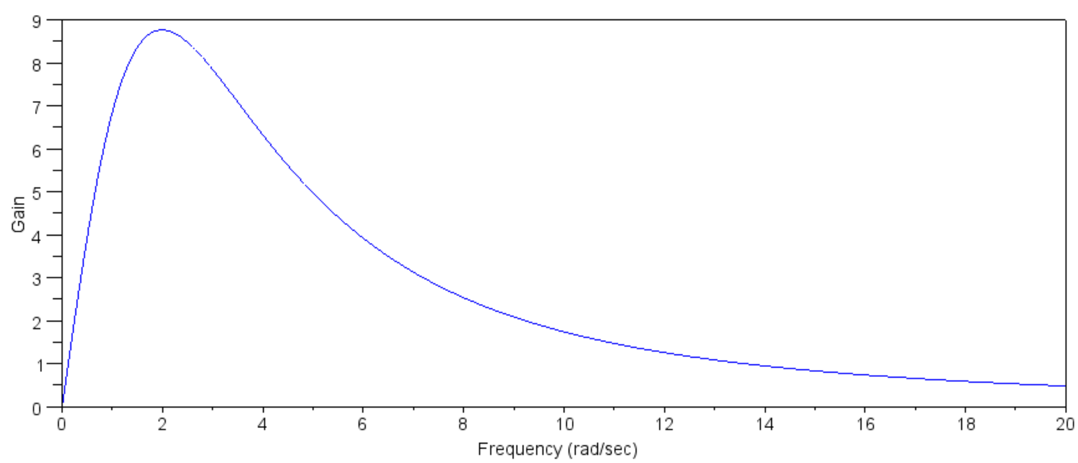
Filter analysis is easy: just evaluate G(jw) to find the gain vs. frequency.

Example 1: Plot the gain vs. frequency for the following filter:

$$Y = \left(\frac{200s}{(s+2)(s+3)(s+4)}\right)X$$

Solution: Using Matlab

```
w = [0:0.01:20]';
s = j*w;
G = 200*s ./ ( (s+2) .* (s+3) .* (s+4) );

plot(w,abs(G))
xlabel('Frequency (rad/sec)');
ylabel('Gain');
```



Gain of G(jw) from 0 to 20 rad/sec

What this graph tells you is:

- The gain is zero at DC (s = 0)
- The gain is a maximum at 2 rad/sec
- The gain goes back to zero as the frequency goes to infinity

Design

Design is a bit more tricky. Note that the transfer function in general has zeros and poles:

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

Graphically, the gain can be interpreted as

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

This leads to one design technique:

- Place zeros close to the frequencies you want to block (multiply by a small number)
- Place poles close to the frequencies you want to pass (divide by a small number)

The closer you place the pole or zero to the jw axis, the more selective the filter is.

Using Matlab, you can use the function *fminsearch()* to design a filter. *fminsearch()* finds the minimum of a function. For example, suppose you want to find the square root of two:

$$x = \sqrt{2}$$

To do that, set up a function where

- You pass your guess at x,
- Compute the error

$$e = x^2 - 2$$

- And return a cost function, J(x), which is the error squared

$$J = e^2$$

fminsearch() will then guess x over and over until it finds the value that minimizes J. In Matlab:

```
function [ J ] = cost( z )

   x = z;
   e = x*x - 2;
   J = e^2;

   end
```

From Matlab, you can guess the value of x over and over again. The goal is to find the value that returns zero (the squared error is zero, meaning the error is zero)

```
cost(5)
   529

cost(4)
   196

cost(3)
    49
```

or you can let Matlab do the guessing for you:

```
[z,e] = fminsearch('cost',5)

z = 1.4142

e = 6.7242e-009
```

This tells you that

- The error is almost zero (*fminsearch()* was able to find the solution), and
- That solution was 1.4142

Example 2: Real Poles

Design a filter of the form

$$G(s) = \left(\frac{4abcd}{(s+a)(s+b)(s+c)(s+d)} \right)$$

so that the gain of the filter is as close as possible to an ideal low-pass filter:

$$|G_d(j\omega)| = \begin{cases} 4 & 0 < \omega < 3 \\ 0 & \omega > 3 \end{cases}$$

Solution: Set up a function in Matlab where

- You guess (a, b, c, d),
- If computes G(jw) for these values of (a, b, c, d),
- It computes the difference (error) in the gain between G(s) and Gd(s), and
- It returns the sum squared error

Matlab Code:

```
function [ J ] = cost2( z )

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

   w = [0:0.01:10]';
   s = j*w;

   Gd = 4 * (w < 3);

   Gs = 4*a*b*c*d ./ ( (s+a) .* (s+b) .* (s+c) .* (s+d) );

   e = abs(Gd) - abs(Gs);

   J = sum(e.^2);

   plot(w,abs(Gd),w,abs(Gs));
   pause(0.01);

   end
```

Calling this in Matlab:

```
>> [z,e] = fminsearch('cost2',[1,2,3,4])

z =     4.0828     4.0828     4.0827     4.0828

e =     747.6329
```

This tells you that

- It wasn't able to exactly match the desired response. The best it could do had a sum-squared error of 747.36
- The best filter Matlab could come up with for this cost function was

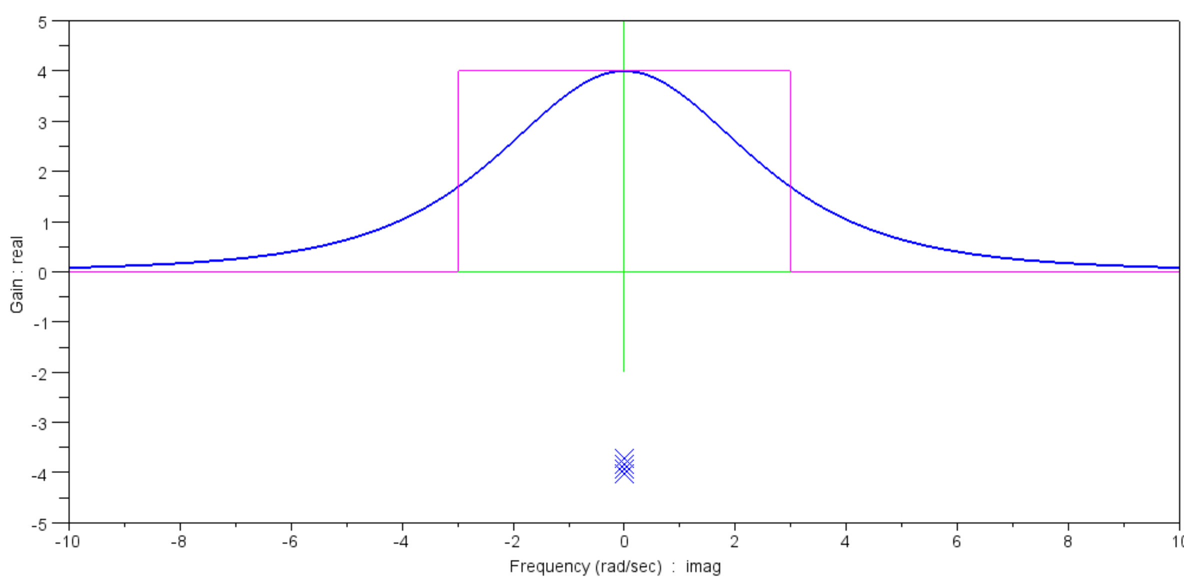
$$G(s) = 4 \cdot \left(\frac{4.0828^4}{(s+4.0828)^4} \right)$$

Pictorially, the graph below shows

- The location of the four poles on the complex plane (marked by x)
- The gain times 4 vs. frequency, rotated (shown in blue)

Note that

- When you are close to the poles, the gain is large (near w = 0)
- When you move away from the poles, the gain drops



Location of the poles (blue x) and G(jw) vs. frequency (rotated - shown in dark blue)

This isn't a very good filter. If you constrain yourself to using real poles, you can't do much.

Example 3: Complex Poles

Design a filter of the form

$$G(s) = \left(\frac{4bd}{(s^2+as+b)(s^2+cs+d)} \right)$$

so that the gain of the filter is as close as possible to an ideal low-pass filter:

$$|G_d(j\omega)| = \begin{cases} 4 & 0 < \omega < 3 \\ 0 & \omega > 3 \end{cases}$$

Solution: Change the cost function:

```
function [ J ] = cost2( z )

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

   w = [0:0.01:10]';
   s = j*w;

   Gd = 4 * (w < 3);

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

   e = abs(Gd) - abs(Gs);

   J = sum(e.^2);


   plot(w,abs(Gd),w,abs(Gs));
   pause(0.01);

   end
```

Let Matlab iterate to find the best filter:

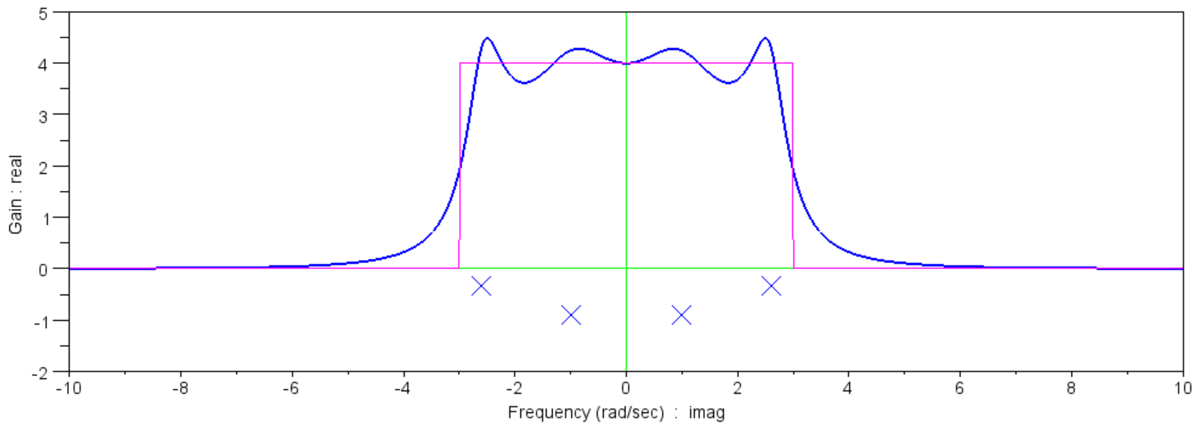
```
[z,e] = fminsearch('cost2',[1,2,3,4])

z =   1.7864    1.7789    0.6427    6.8938

e =   160.6407
```

This tells you

- The filter is better: the sum squared error is now 160 vs. 747
- The 'best' filter given this cost function is

$$G(s) = 4\left(\frac{1.7789 \cdot 6.8938}{(s^2+1.7864s+1.7789)(s^2+0.6427s+6.8938)} \right)$$

$$G(s) = 4\left(\frac{1.7789 \cdot 6.8938}{(s+0.8932 \pm j0.9905)(s+0.3213 \pm j2.6059)} \right)$$



Gain vs. Frequency with 4 complex poles (thick blue line) and pole locations (blue x's)

Note that

- This filter is much better: if you are allowed to use complex poles, you can approximate an ideal filter much better than with real poles.
- The poles are scattered across the passband: from -j3 to +j3
- When you are close to a pole, you get a resonance (the gain has a peak at that frequency)

Example 4: Complex poles and zeros

Design a filter with 4 poles and two zeros:

$$G(s) = 4\left(\frac{es^2+fs+bd}{\left(s^2+as+b\right)\left(s^2+cs+d\right)}\right)$$

Solution: Modify the cost function

```
function [ J ] = cost2( z )

   a = z(1);
   b = z(2);
   c = z(3);
   d = z(4);
   e = z(5);
   f = z(6);

   w = [0:0.01:10]';
   s = j*w;

   Gd = 4 * (w < 3);

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

   e = abs(Gd) - abs(Gs);

   J = sum(e.^2);

   plot(w,abs(Gd),w,abs(Gs));
   pause(0.01);

   end
```

Call it using Matlab:

```
[z,e] = fminsearch('cost2',[1,2,3,4,5,6])

z =    0.4223    7.8893    2.2963    3.1890    1.9867   -0.0001

e =    92.5262
```

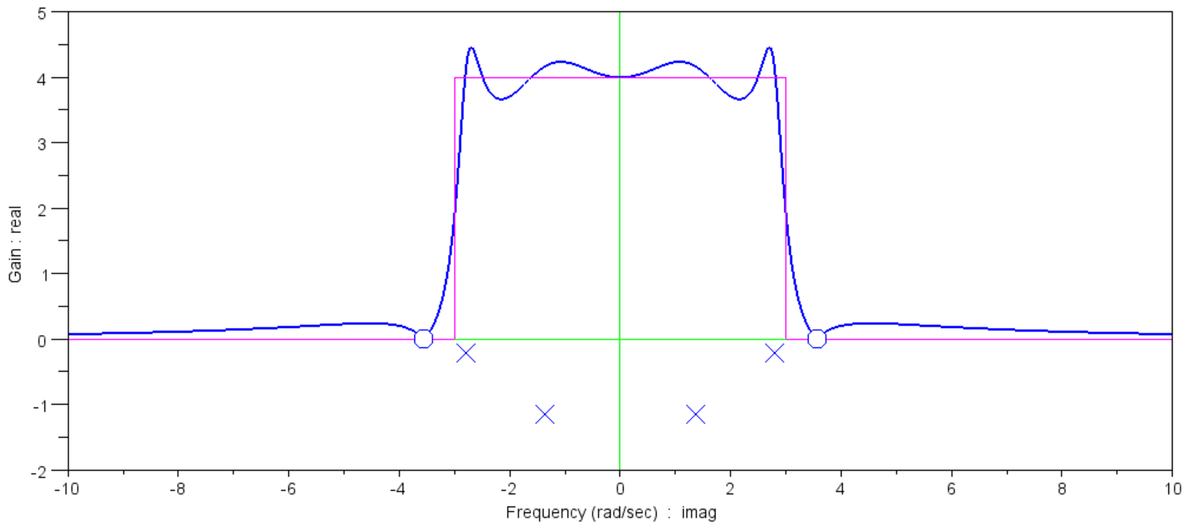
This tells you

- The filter is better: the sum-squared error is now 92 (vs 747 or 160)
- The best Matlab can do with this cost function is

$$G(s) = 4\cdot\left(\frac{1.9867s^2-0.0001s+7.8893\cdot 3.1890}{\left(s^2+0.4223s+7.8893\right)\left(s^2+2.2963s+3.1890\right)}\right)$$

or

$$G(s) = 4\cdot\left(\frac{1.9867(s\pm j3.5586)}{(s+0.2111\pm j2.8008)(s+1.1481\pm j1.3678)}\right)$$



Gain vs. Frequency (dark blue line) and pole / zero location

Note that

- By adding a zero at j3.55, the gain is zero at 3.55 rad/sec.
- The pole near the zero was pushed out and towards the real axis (pole at s -0.2111 + j2.8008)