FIR Filters

NDSU ECE 376

Lecture #29

Inst: Jake Glower

Please visit Bison Academy for corresponding lecture notes, homework sets, and solutions

Finite Impulse Response (FIR) Filter Design:

Time and frequency are related:

$$f(t) = \frac{1}{j2\pi}\int_{-j\infty}^{+j\infty} F(s)e^{st} \cdot ds$$

$$F(s) = \int_{-\infty}^{+\infty} f(t)e^{st}dt$$

If two linear filters have identical impulse responses, the two filters will have the same frequency response.

It really doesn't matter how you generate the impulse response of a filter. All that matters is the result.

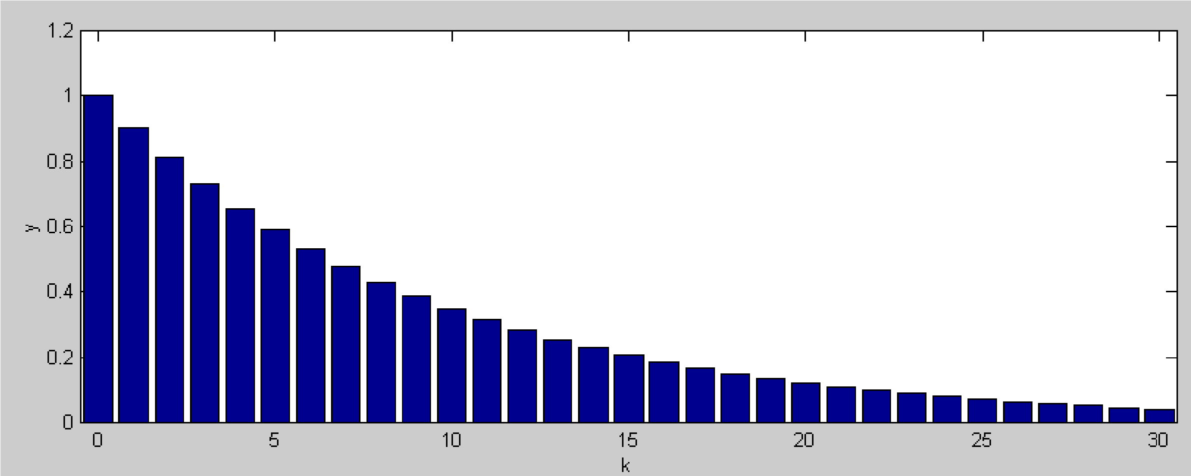
Example:

$$Y = \left(\frac{1}{z-0.9}\right)X$$

$$Y = \left(1 + \frac{0.9}{z} + \frac{0.9^2}{z^2} + \frac{0.9^3}{z^3} + \ldots\right)X$$

or

$$y(k) = (0.9)^k u(k)$$



Implementing G(z)

IIR Filter (recursive filter)

```
zY = 0.9*Y + X
 Y = zY
```

FIR Filter

```
for (i=100; i>0; i--) X[i] = X[i-1];
X[0] = a2d_read(0);

Y = 0;
for (i=0; i<=100; i++)
   Y += W[i] * X[i];


// W[i] = 0.9^i
```

Result

A Finite Impulse Response Filter

- Remembers the previous N values of the input (X),
- Combines these previous N inputs with weightings corresponding to the impulse response of the filter you want to implement,
- Thus generating your desired filter.

The neat thing about FIR filters is

- If you know the impulse response of the filter you want,
- You know how to implement this filter.

Example: Ideal Low-Pass Filter:

Design a filter with the gain of

$$F(s) = \begin{cases} 1 & |\omega| < 1 \\ 0 & otherwise \end{cases}$$

Solution: Find the impulse response

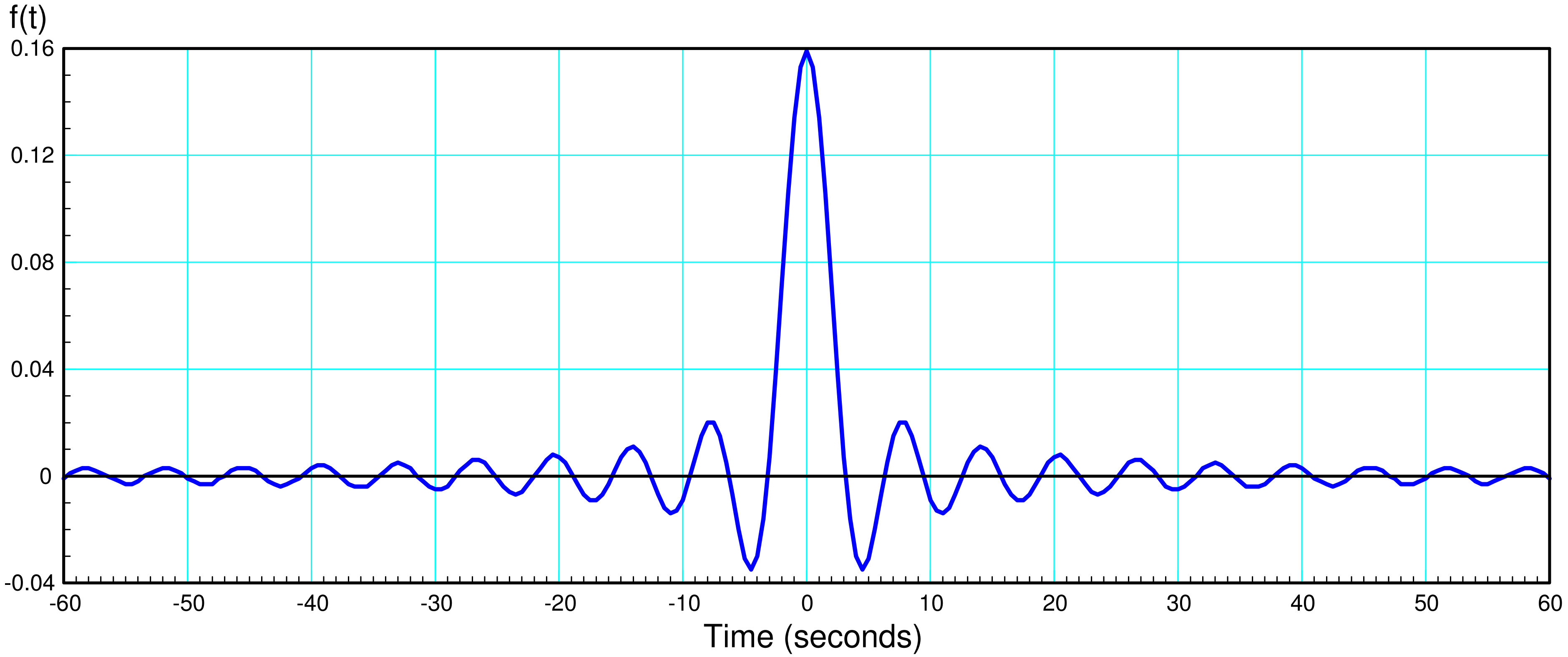
$$f(t) = \frac{1}{j2\pi} \int_{-j\infty}^{+j\infty} F(s) e^{st} \cdot ds$$

$$\vdots$$

$$f(t) = \left(\frac{1}{2\pi}\right)\left(\frac{\sin(t)}{t}\right)$$

Problems:

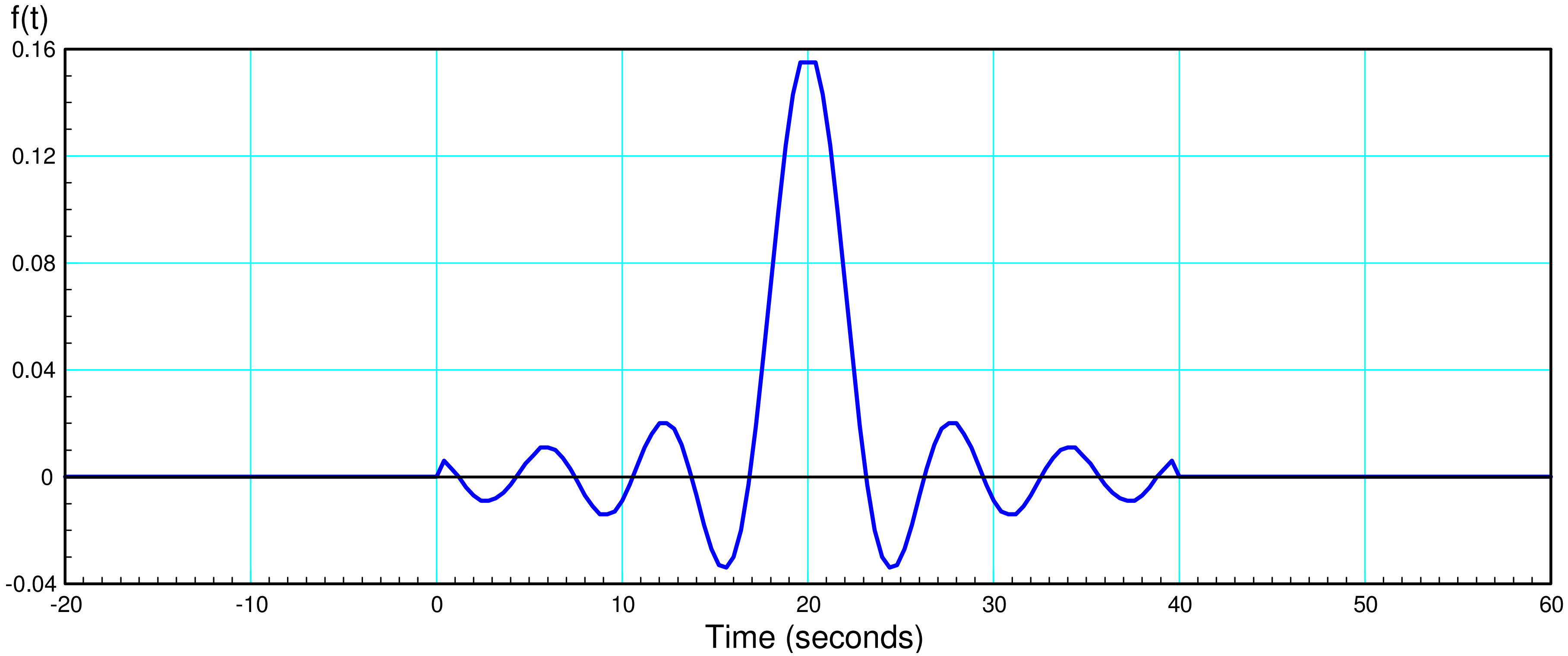
- Non-Causal
- Goes from $-\infty < t < \infty$.



Approximations:

- Truncate for -20 seconds < t < +20 seconds
- Delay 20 seconds

Results in a causal filter (with a 20 second delay)



Frequency Response:

The frequency response will be equal to

$$G(s) = \sum W(i) \cdot z^{-i}$$

where

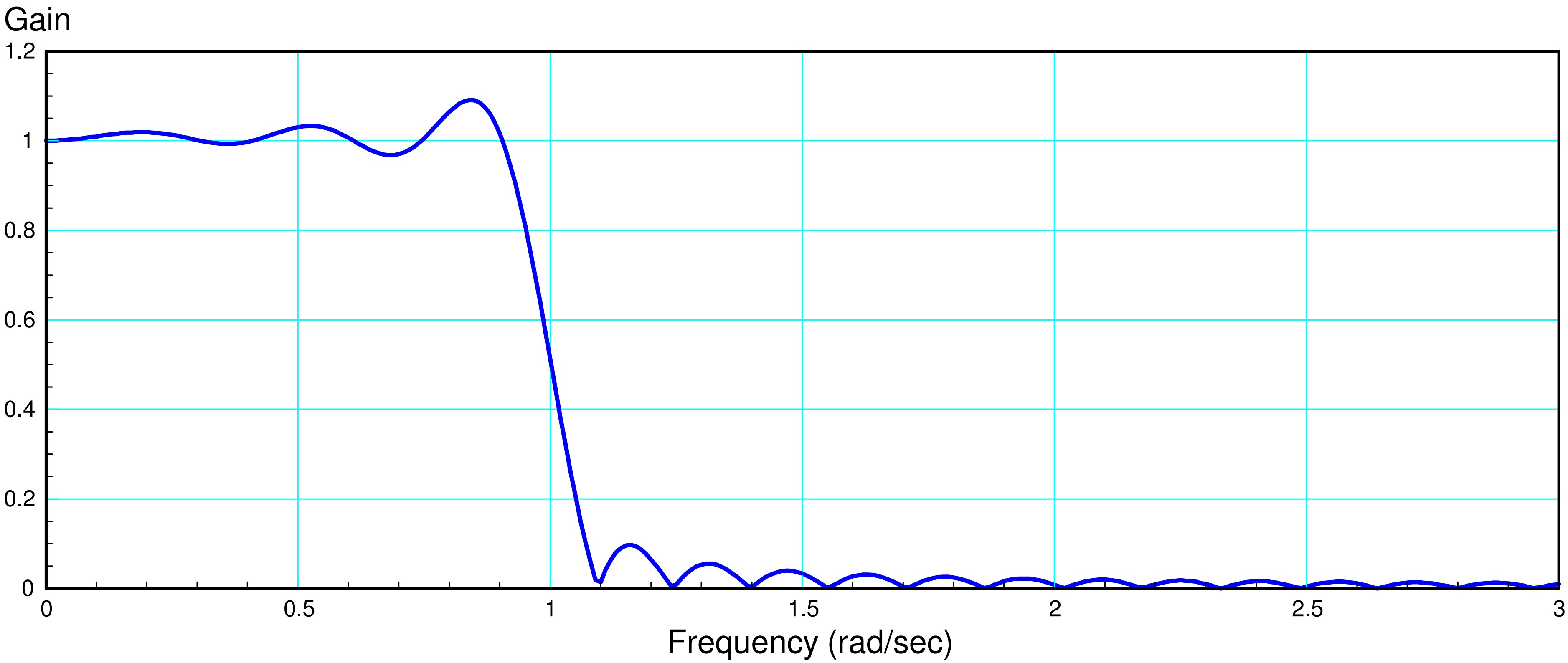
$$z = e^{j\omega T}$$

MATLAB Code:

```
t = [-20:T:20]'  +  1e-6;
f = 1 / (2*pi) * sin(t) ./ t ;

w = [0:0.01:3]';
s = j*w;
T = 0.4;
z = exp(s*T);
G = 0*w;
for i=1:length(f)
   G = G + f(i) * (z .^ (-i));
   end
plot(w,abs(G))
```

$$f(t) = \left(\frac{1}{2\pi}\right)\left(\frac{\sin(t)}{t}\right)$$



On the plus side:

- This is a very good low-pass filter, closely approximating an ideal low-pass filter.
- If you want a better filter, extend the tails of the impulse response
- This filter is very easy to implement: all you need is to know the impulse response of your filter

On the minus side:

- It involves remembering 400 previous inputs (requiring more RAM than is available of a PIC).
- It involves 400 floating point multiplies and 400 floating point additions
- It would take a PIC about 0.8 seconds to compute y(k) each sample and you only have 0.01 second to do so.

In short, a FIR filter is not a good option for a PIC. A DSP (digital signal processor) is designed specifically for this type of filter.

Example 2:

Design an FIR low-pass filter with a corner at 2 rad/sec

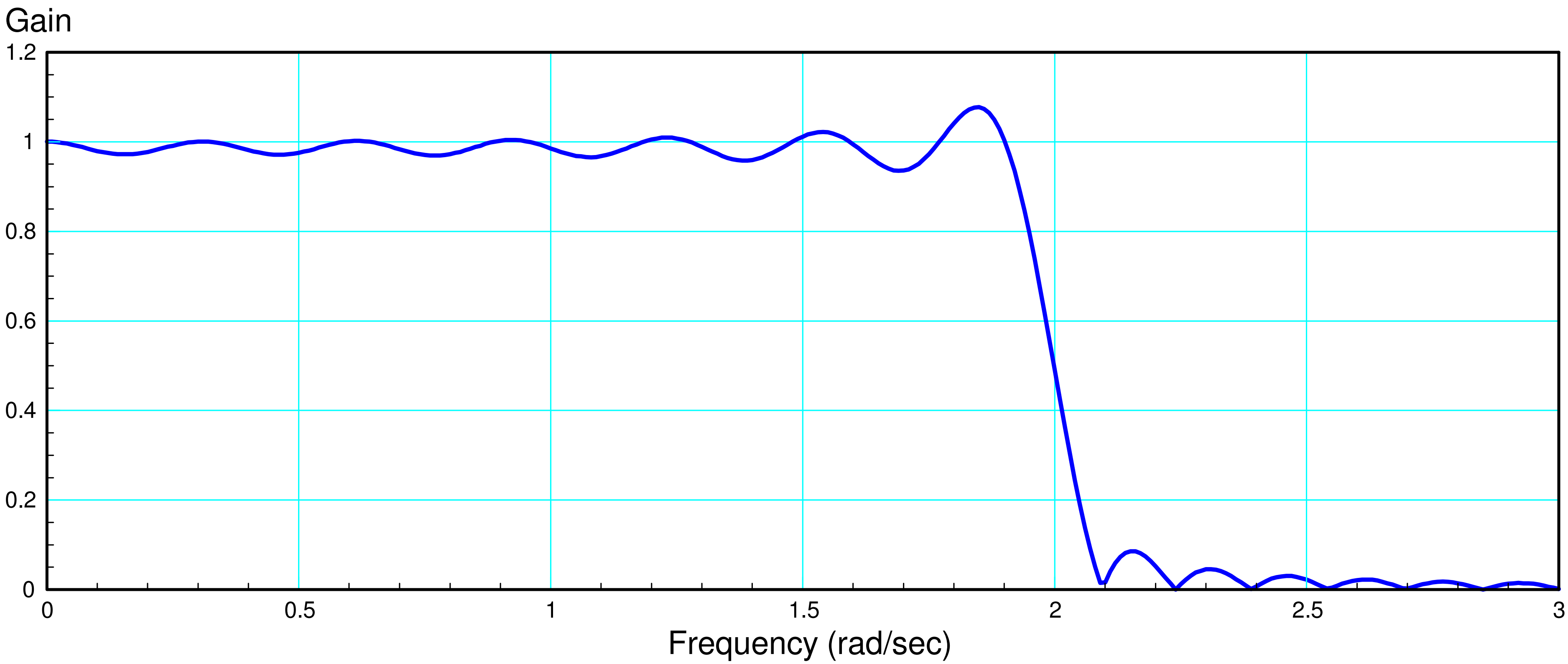
Solution:

- If you double the bandwidth, you double the speed of the filter
- Speed up time 2x

$$f(t) = \left(\frac{1}{2\pi}\right)\left(\frac{\sin(t)}{t}\right)$$ corner = 1 rad/sec

$$f(t) = 2\left(\frac{1}{2\pi}\right)\left(\frac{\sin(2t)}{2t}\right)$$ corner = 2 rad/sec

$$f(t) = \left(\frac{1}{2\pi}\right)\left(\frac{\sin(2t)}{t}\right)$$



Example 3:

Design a band-pass filter

$$F(s) = \begin{cases} 1 & 4 < \omega < 6 \\ 0 & otherwise \end{cases}$$

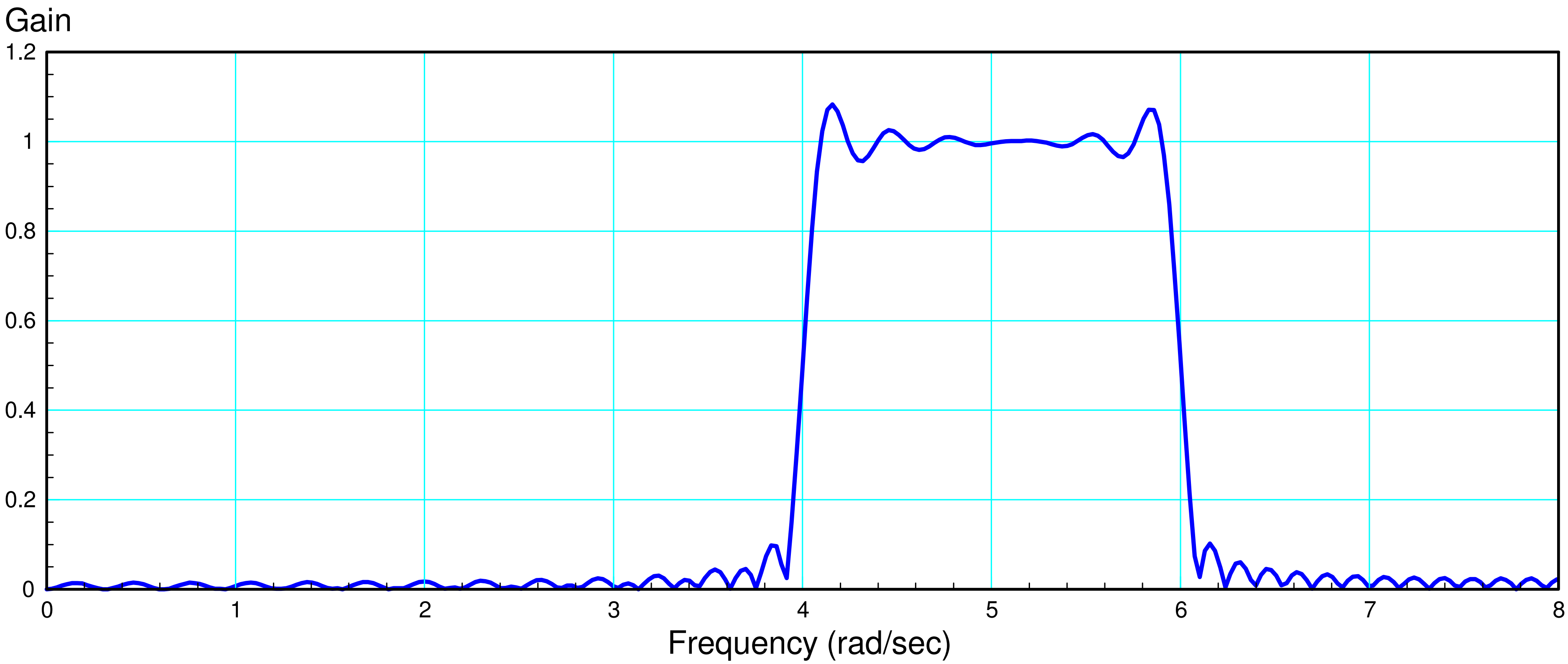
Solution: Subtract

f(t) = Low-Pass Filter with a corner at 6 rad/sec

- Low Pass Filter with a corner at 4 rad/sec

$$f(t) = \left(\frac{1}{2\pi}\right)\left(\frac{\sin(6t)}{t}\right) - \left(\frac{1}{2\pi}\right)\left(\frac{\sin(4t)}{t}\right)$$

$$f(t) = \left(\frac{1}{2\pi}\right)\left(\frac{\sin(6t)}{t}\right) - \left(\frac{1}{2\pi}\right)\left(\frac{\sin(4t)}{t}\right)$$



FIR Summary

- If you know the impulse response of your filter, you can implement it with an FIR filter
- FIR filters are easy to design
- FIR filters are easy to implement

But....

- The require a LOT of floating-point computations
- PIC processors are not designed for this
- DSP processors *are* designed for this (ECE 444: Digital Signal Processors)