Digital Signal Processing

Custom FIR and Optimal Filters

Topics

- Introduction to Custom FIR Filters
- Transformation from Frequency Domain to Impulse Response
- In Class Problem
- Application – Deconvolution
- Example
- Application – Optimal Filtering

Filter Classification

FILTER IMPLEMENTED BY:

FILTER USED FOR:	Convolution *Finite Impulse Response (FIR)*	Recursion *Infinite Impulse Response (IIR)*
Time Domain *(smoothing, DC removal)*	Moving average (Ch. 15)	Single pole (Ch. 19)
Frequency Domain *(separating frequencies)*	Windowed-sinc (Ch. 16)	Chebyshev (Ch. 20)
Custom *(Deconvolution)*	FIR custom (Ch. 17)	Iterative design (Ch. 26)

Custom FIR Filters

- Most filters have low-pass, high-pass, band-pass, or band-reject characteristics.
- With FIR filters, it is possible to create an arbitrarily complex frequency response.
- The ability to tailor the frequency response enables two useful techniques
  - De-convolution and optimal filtering.

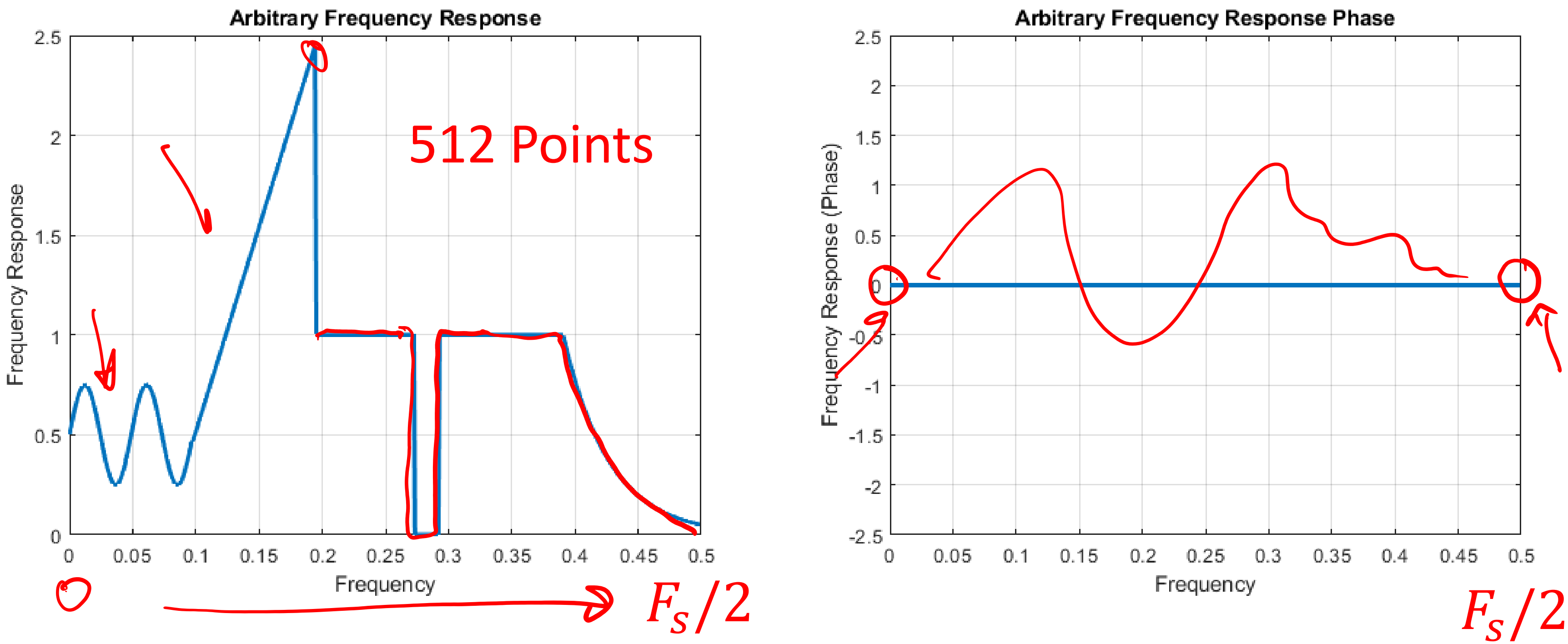
Steps to Create A Custom FIR Filter

- Identify the frequency response that is desired.

- Take the inverse FFT of the frequency domain response to get the impulse response of the FIR filter that implements this frequency response.

Custom FIR Example

- Start with an arbitrary frequency response



- Both magnitude and phase can be arbitrary
  - Phase must start and end at 0 (or 2π)

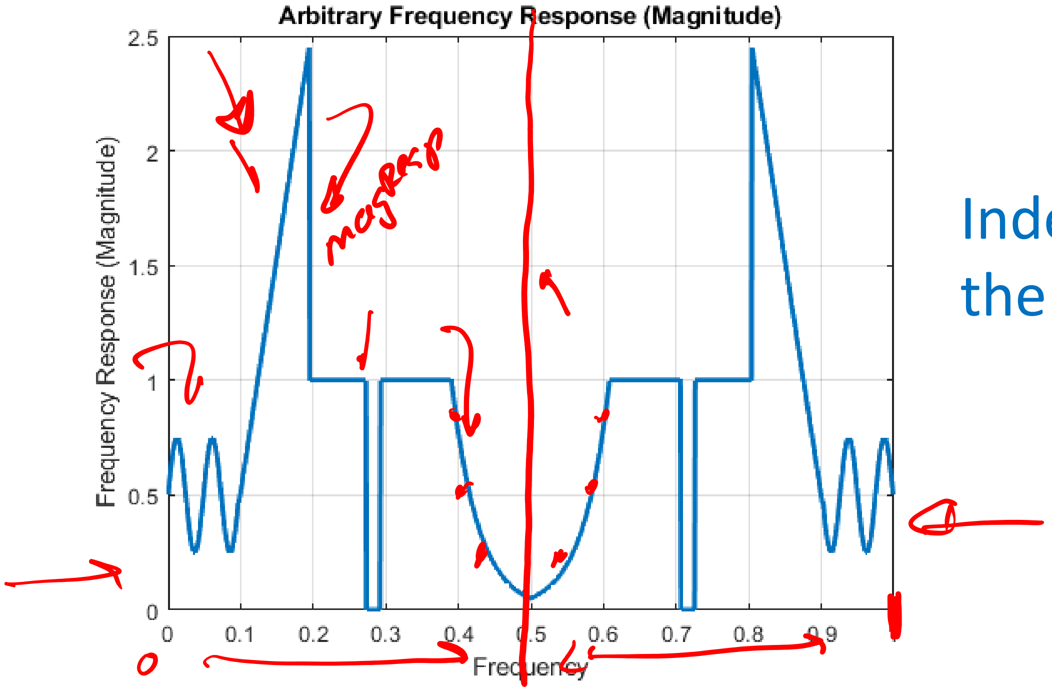
Custom FIR Example

- The impulse response will be found by taking the inverse FFT then shifting and windowing
- The FFT has samples from 0 to F_s the sampling frequency
- The frequency response is mirrored around the Nyquist frequency or $F_s/2$.
- Must add points to make it correctly mirrored

Mirror the Spectrum to Create the FFT

- Can concatenate row vectors easily in MATLAB

```
%  Mirror the frequency response about Nyquist

fullMagResp = [magResp, magResp(end:-1:1)];
fullPhaseResp = [phaseResp, phaseResp(end:-1:1) ];
```



Index the vector from the end to the first sample

Convert the Magnitude and Phase to Real and Imaginary

- From the magnitude and phase samples

$$M_k = Magnitude\ Samples$$

$$\theta_k = Phase\ Samples$$

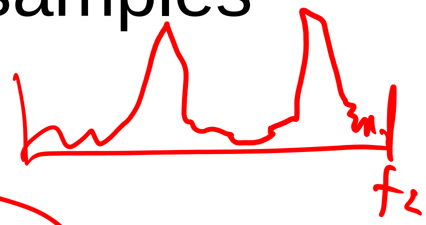
$$X(k) = M_k e^{j\theta_k} = \mathrm{M_k}(\cos(\theta_k) + j\sin(\theta_k))$$

$$Re(X_k) = \boxed{\mathrm{M_k}\cos(\theta_k)}$$

$$Im(X_k) = \boxed{\mathrm{M_k}\sin(\theta_k)}$$

Convert the Magnitude and Phase to Real and Imaginary

- From the magnitude and phase samples



$$Re(X_k) = \mathrm{M_k}\cos(\theta_k)$$

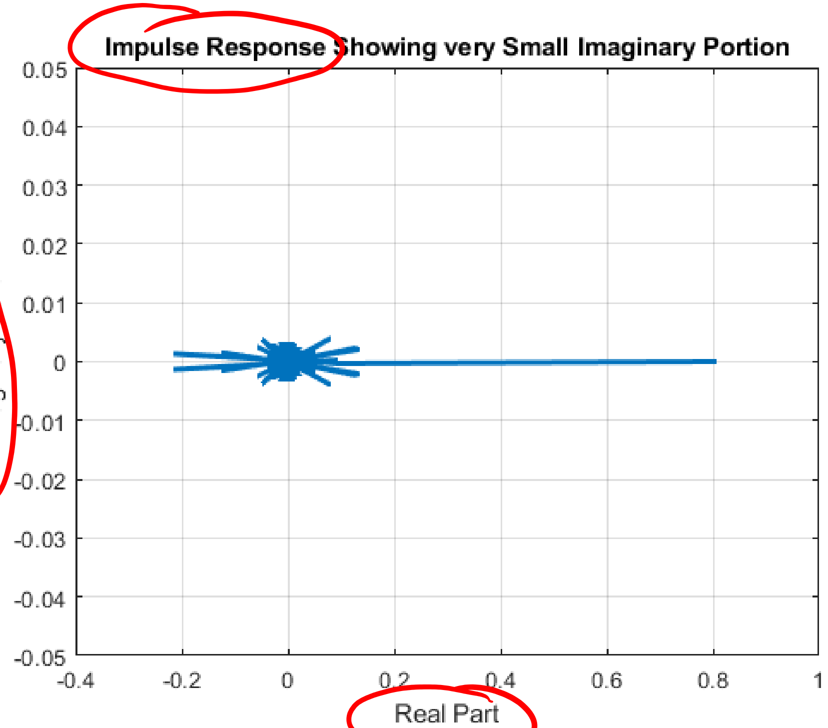
$$Im(X_k) = \mathrm{M_k}\sin(\theta_k)$$

```
realFreqResp = fullMagResp .* cos( fullPhaseResp );
imagFreqResp = fullMagResp .* sin( fullPhaseResp );

cplxFreqResp = realFreqResp + 1j*imagFreqResp;
```

$X(l) = Re(l) + j\,Im(l)$

Take the IFFT of the Frequency Response

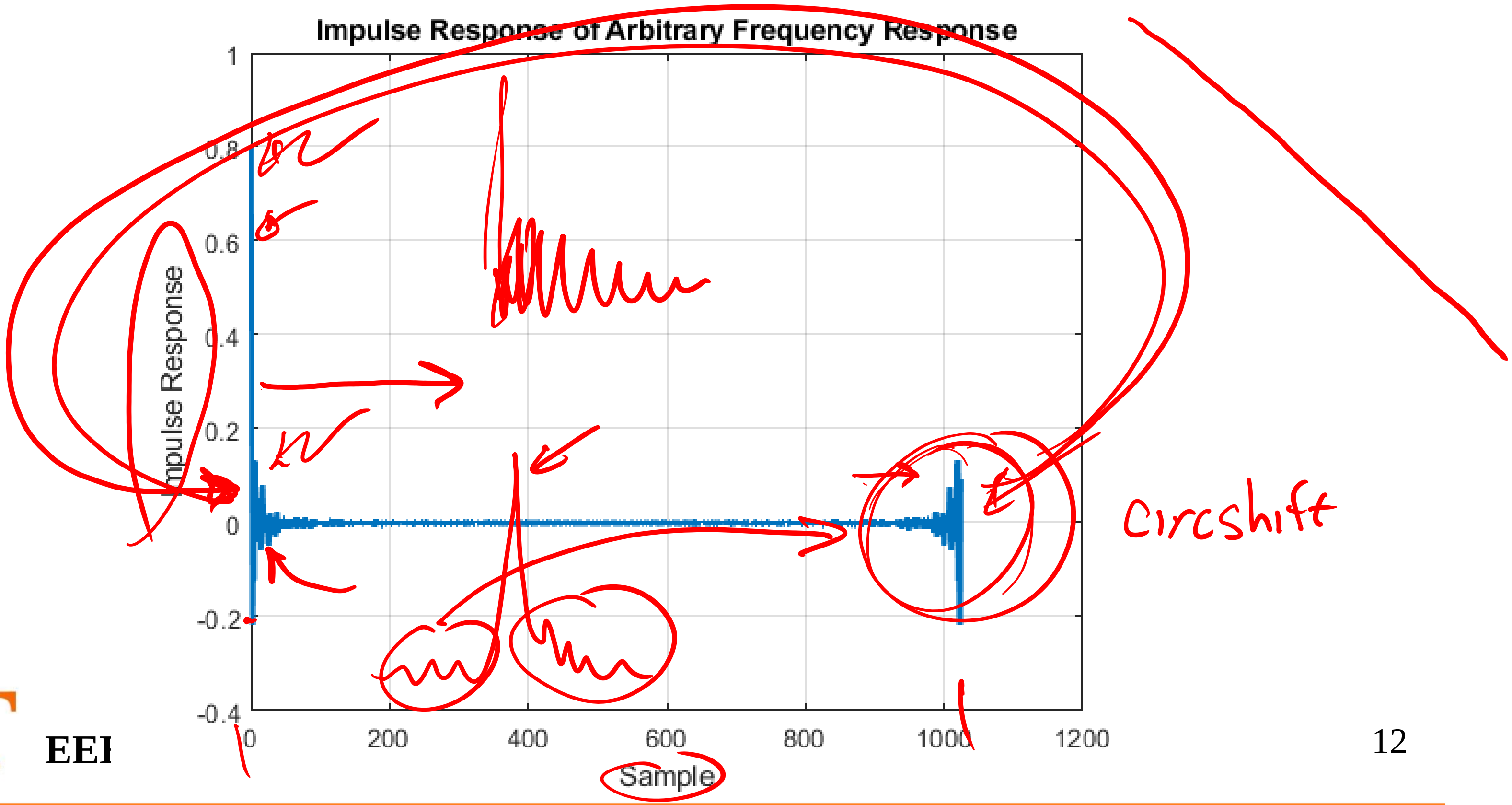
- The IFFT consists of 1024 samples
- It should be all real but due to numerical precision some imaginary components may exist
- Keep just the real part



Real(x[n])

Take the IFFT of the Frequency Response

- After taking the real part the impulse response is shown
- It is centered at sample 0 and wraps at the end points



Shift, Truncate and Window the Impulse Response

- To use the impulse response we have to circularly shift the response

- Its full length will be the same as the FFT length and we may wish to truncate it to limit its length

- After truncation, window the impulse response to remove the abrupt transition at the ends

Shift, Truncate and Window the Impulse Response

Use the circshift function in MATLAB

500

```
%  Shift the impulse response by 256 samples and truncate

filterLength = 500;
filterShift = 250;
testImpulse = circshift( arbImpulse, filterShift );
testImpulse = testImpulse(1:filterLength);
```

Truncate the impulse to the desired filter length

Shift, Truncate and Window the Impulse Response

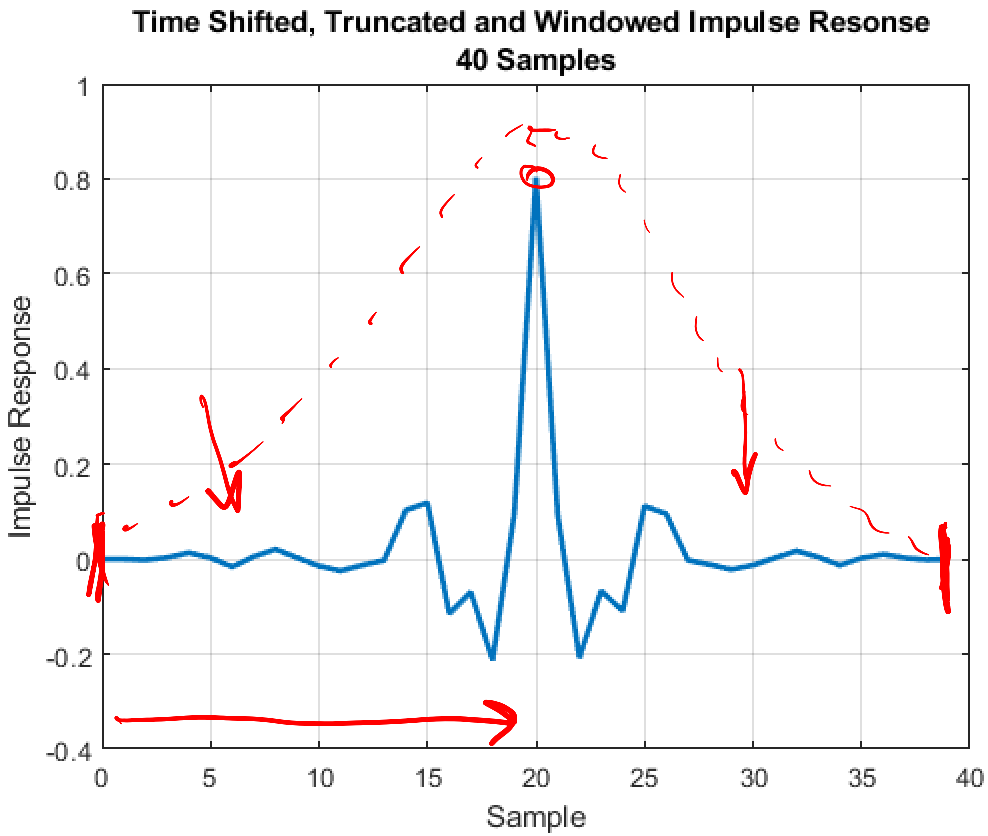
Apply a Hamming Window using the "hamming" function in MATLAB

```
%  Apply a Hamming Window

testImpulse = testImpulse .* hamming(filterLength)';
```

Use point by point multiplication (.*) to window the impulse response

Shift, Truncate and Window the Impulse Response

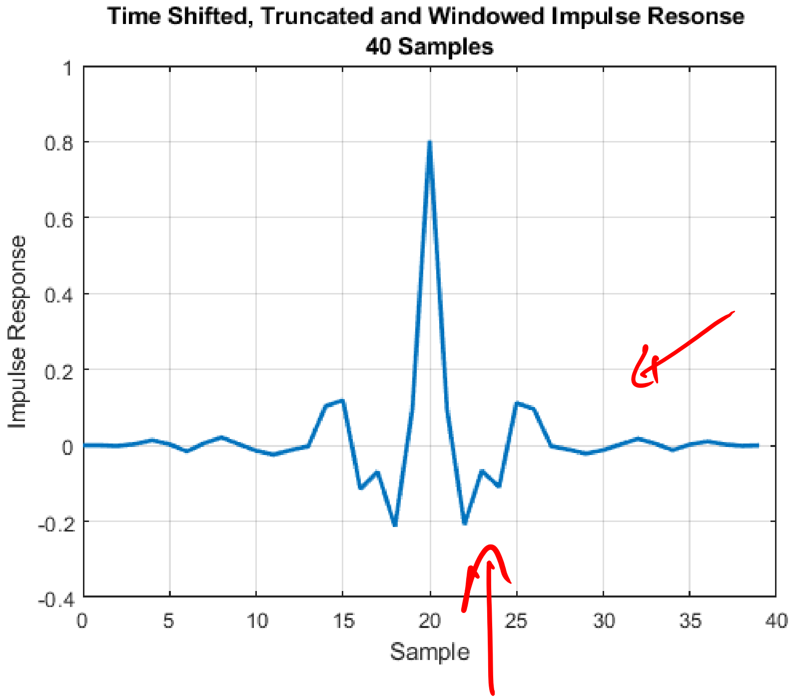
- The length of the response will determine how well the frequency response is reproduced

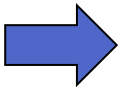


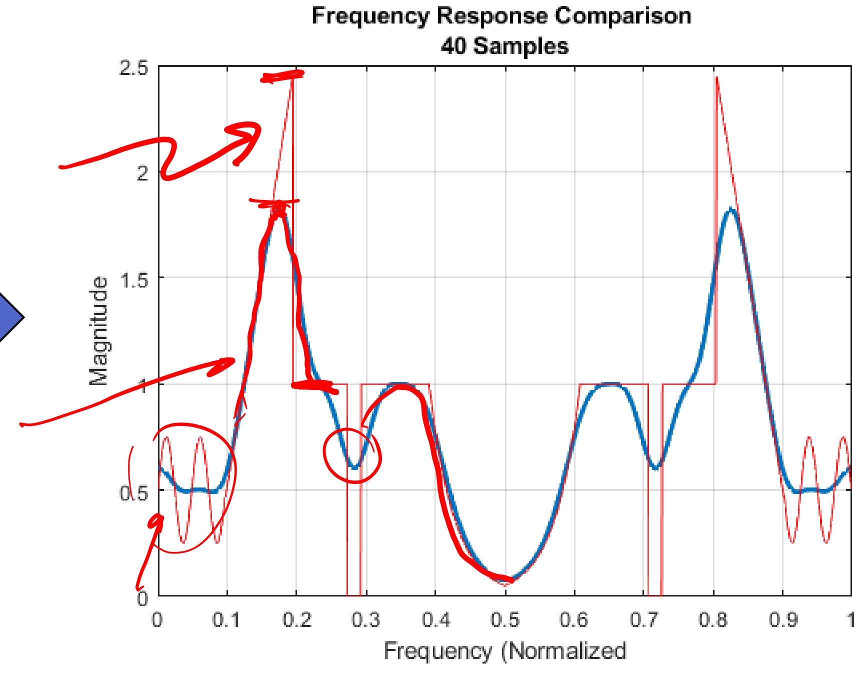
For a length M impulse response circularly shift the impulse response by M/2 samples

Test the Response (Length 40)

- Check the frequency response by taking the FFT
- Pad the sequence to improve frequency resolution

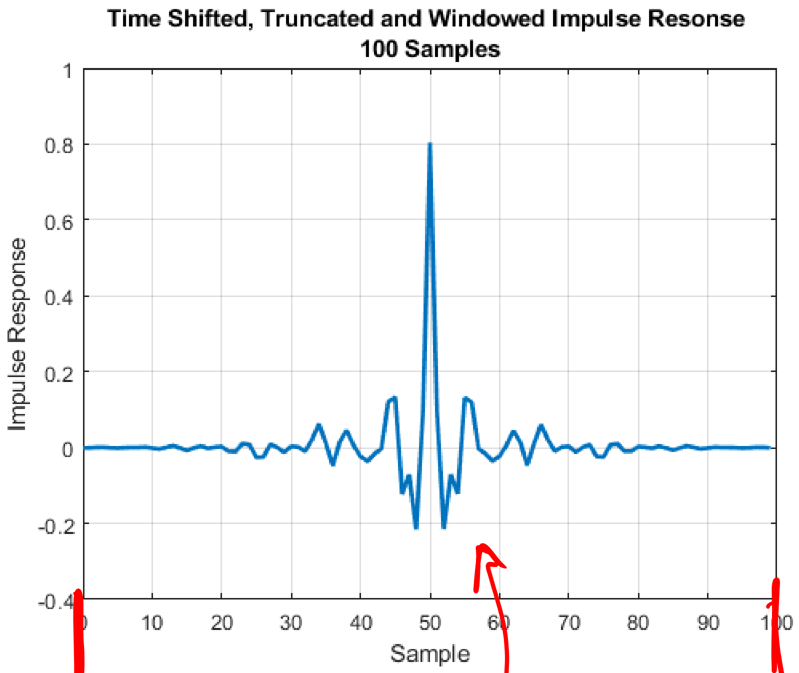


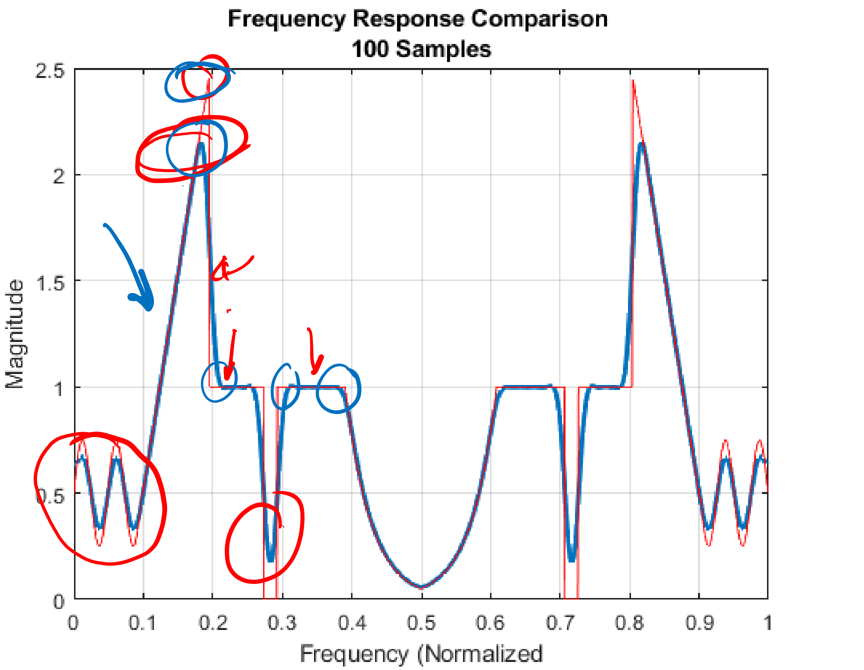




Test the Response (Length 100)

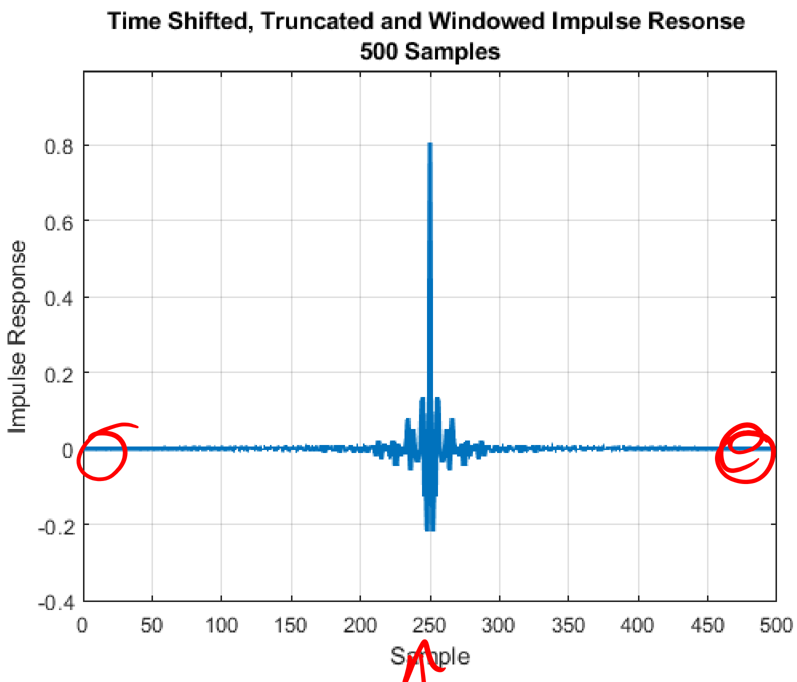
- Check the frequency response by taking the FFT
- Pad the sequence to improve frequency resolution

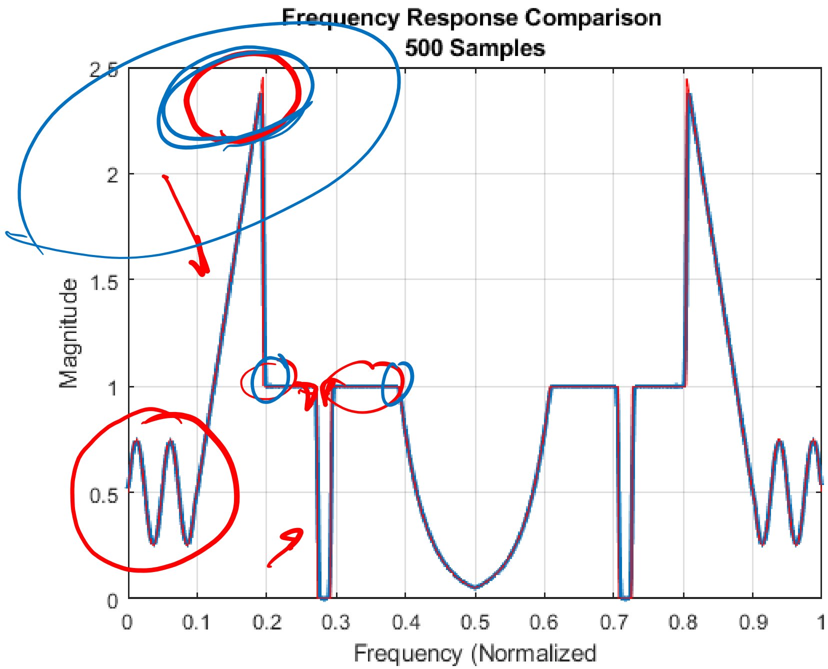


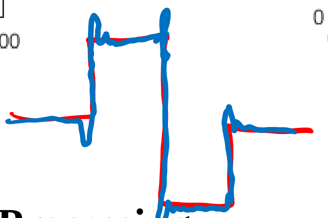


Test the Response (Length 500)

- Check the frequency response by taking the FFT
- Pad the sequence to improve frequency resolution



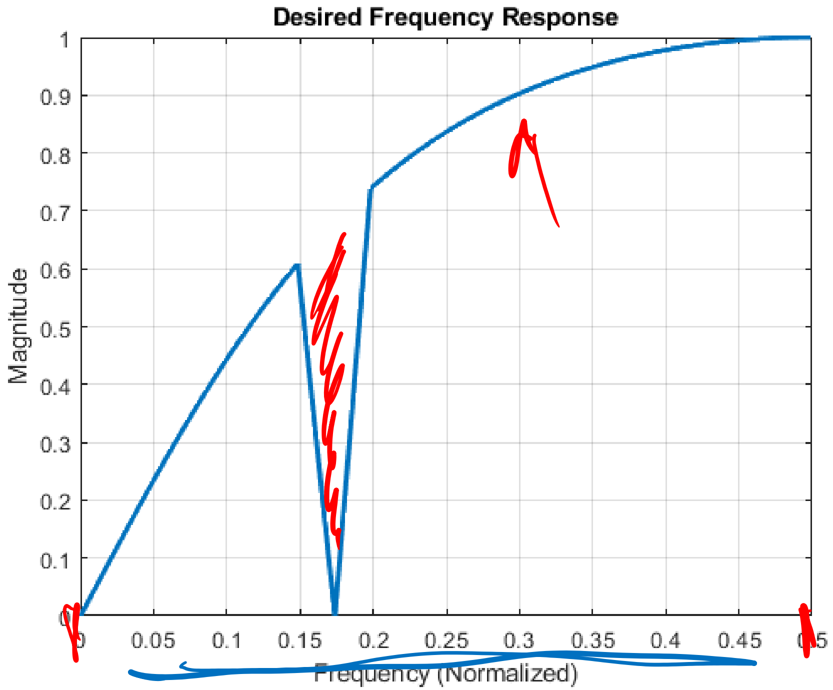




In Class Problem

- Create the impulse response for this custom filter
  - Length 100

- Combination of a high pass with a notch in the transition region



Download "Custom_Filter_ICP.mat" in myCourses

In Class Problem - Steps

- Mirror the frequency response making the frequency response twice as long

N [end:-1:1]

- Take the IFFT of the frequency response
- Circular shift the response by ½ the length of the filter

100

- Truncate the response to the length of the filter

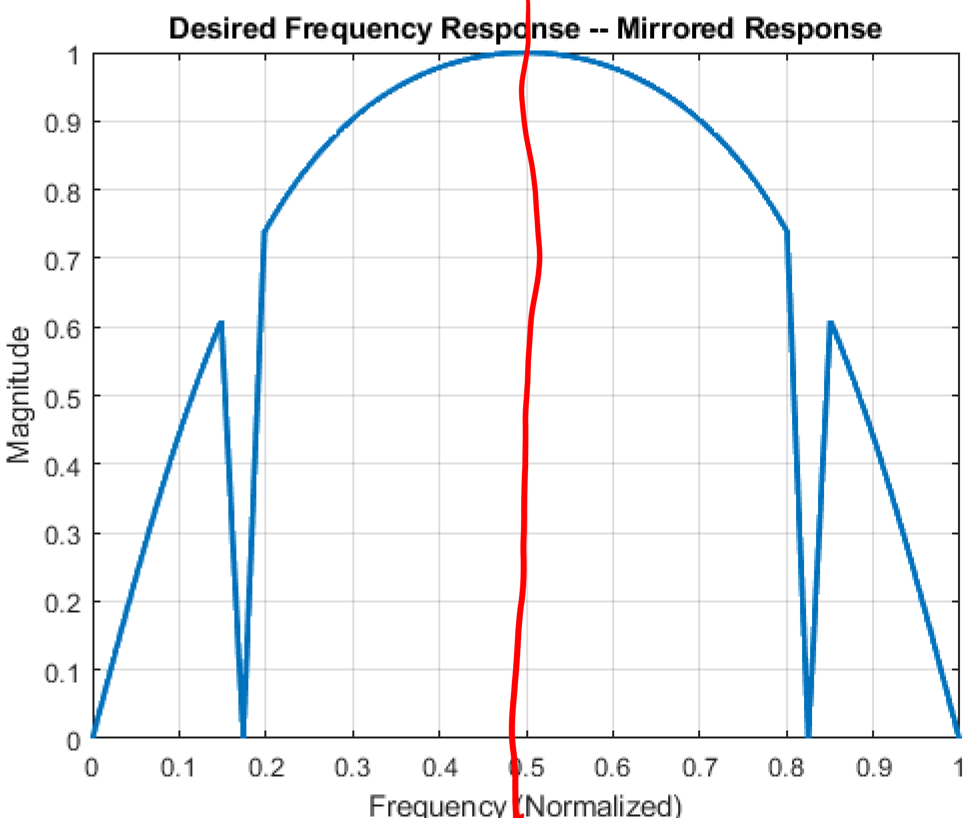
.*

- Window the response with a hamming filter
- Test the frequency response using the FFT to a desired length (your call)

Custom Filter ICP

NO PRINT

- Mirror the frequency response around Nyquist



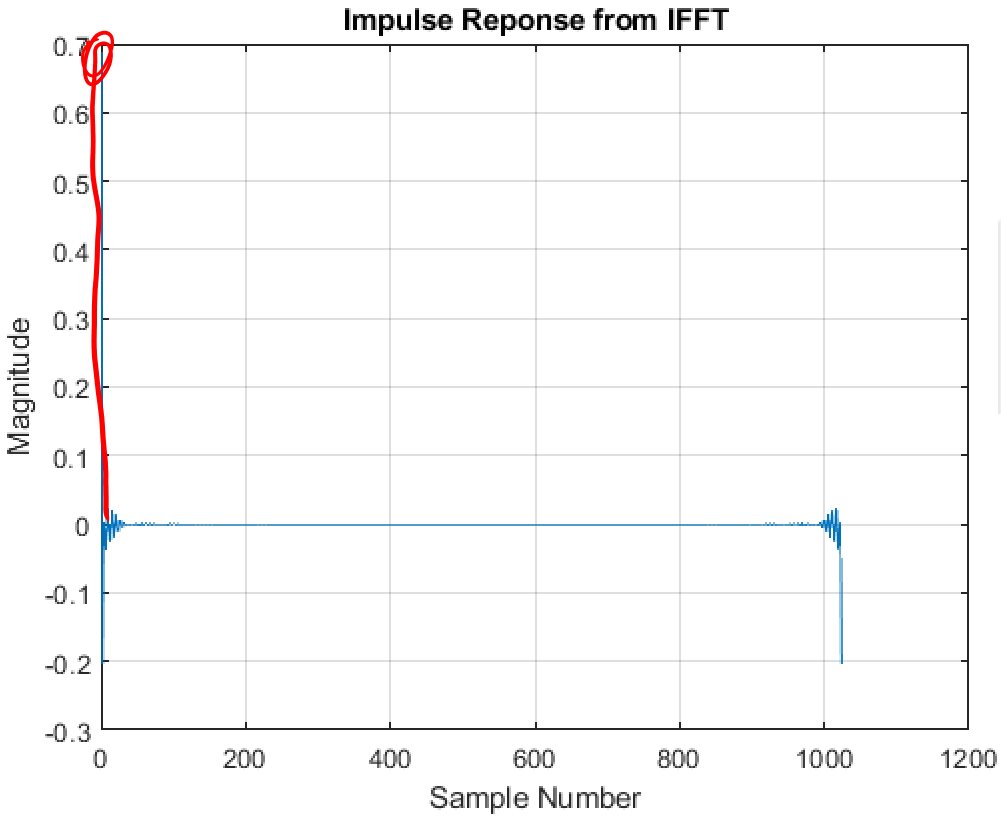

```
% Mirror the frequency response

hFull = [hHPF, hHPF(end:-1:1) ];
```

Concatenate the filter. Use "end" for the last element in the array

Custom Filter ICP

NO PRINT

- Take the IFFT of the frequency response
- Keep only the real part of the IFFT result

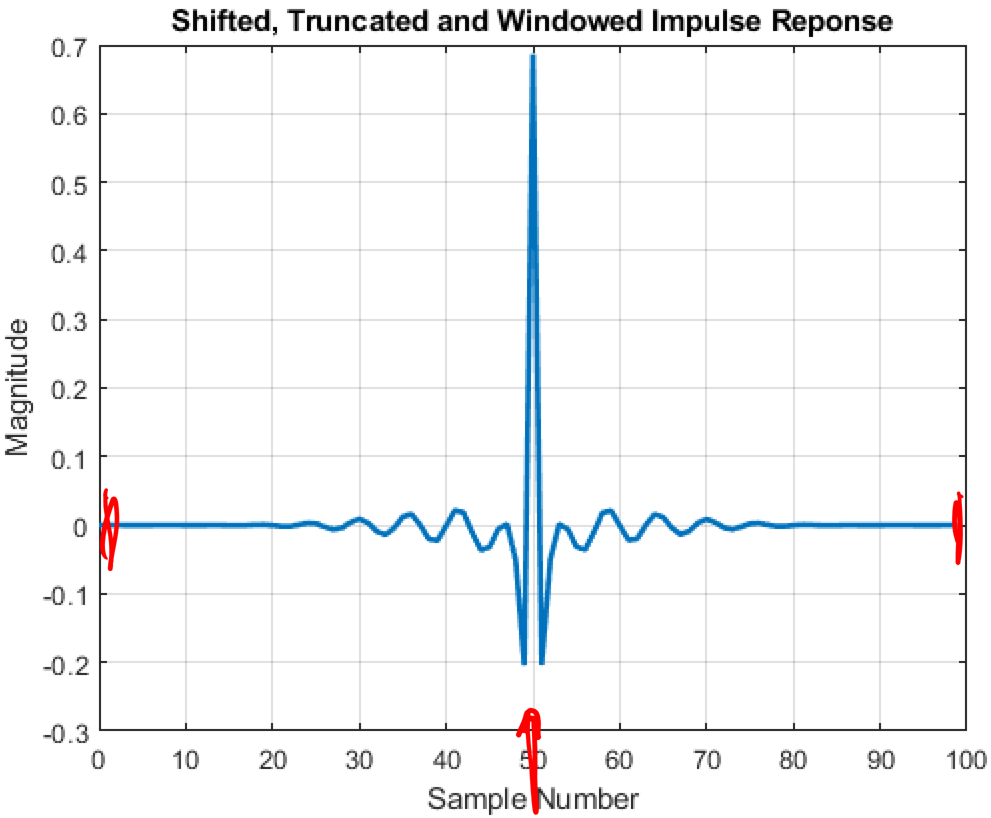



```
%  Take the inverse FFT

iResponse = real(ifft(hFull));
```

Custom Filter ICP

NO PRINT

- Circular shift, truncate and window the impulse response



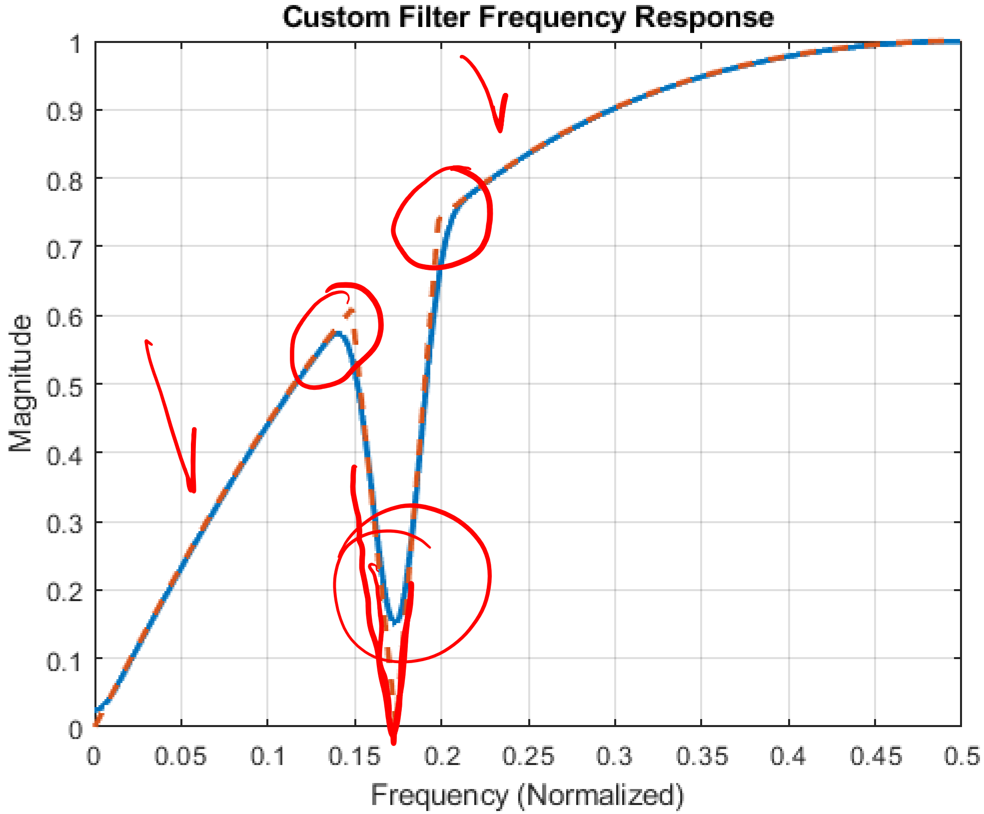

```
% Circular shift, truncate and window

iResponse = circshift( iResponse, filterLength/2 );
iResponse = iResponse(1:filterLength);
iResponse = iResponse .* hamming( filterLength )';
```

Make sure the dimensions are the same when windowing

Custom Filter ICP

NO PRINT

- Test the filter by taking the FFT to any length
- Plot from 0 to 0.5




```
%  Test the filter
fftLen = 1024;
newFilter = fft(iResponse, fftLen );
```

Application - Deconvolution

- Convolution is a process by which a signal is transformed.
  - Example – Filtering a signal with a windowed SINC impulse response – Causes high frequency content to be removed

- Deconvolution reverses this process

- Think of it as an inverse filter.
  - Cancels the effect of the filter

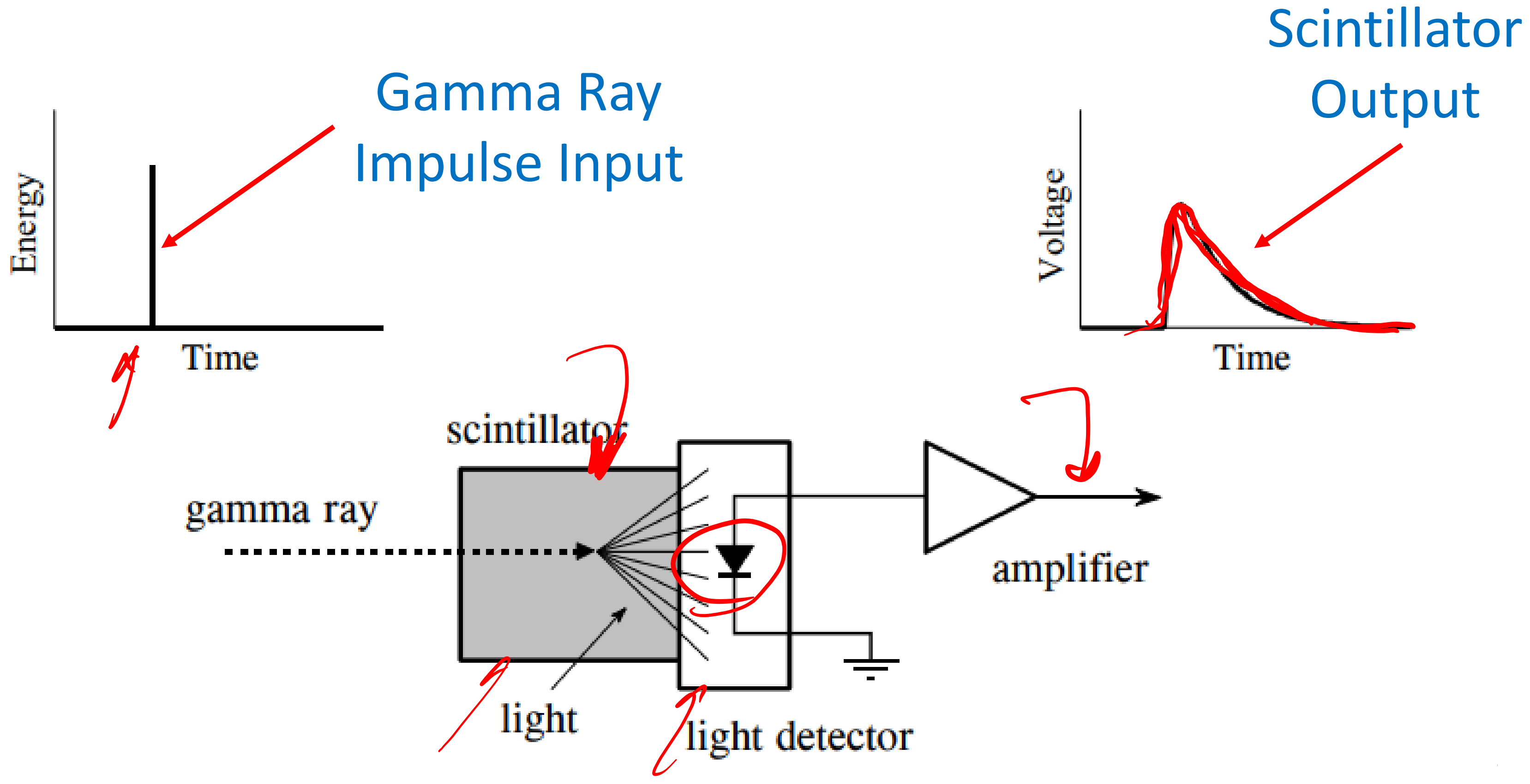
Application - Deconvolution

- In some cases signals are unintentionally filtered and transformed in some way
- It can be helpful to reverse the process to get the original signal back
- Example from the text is the Gamma Ray detector
  - Impulses are filtered/transformed by the detector

27

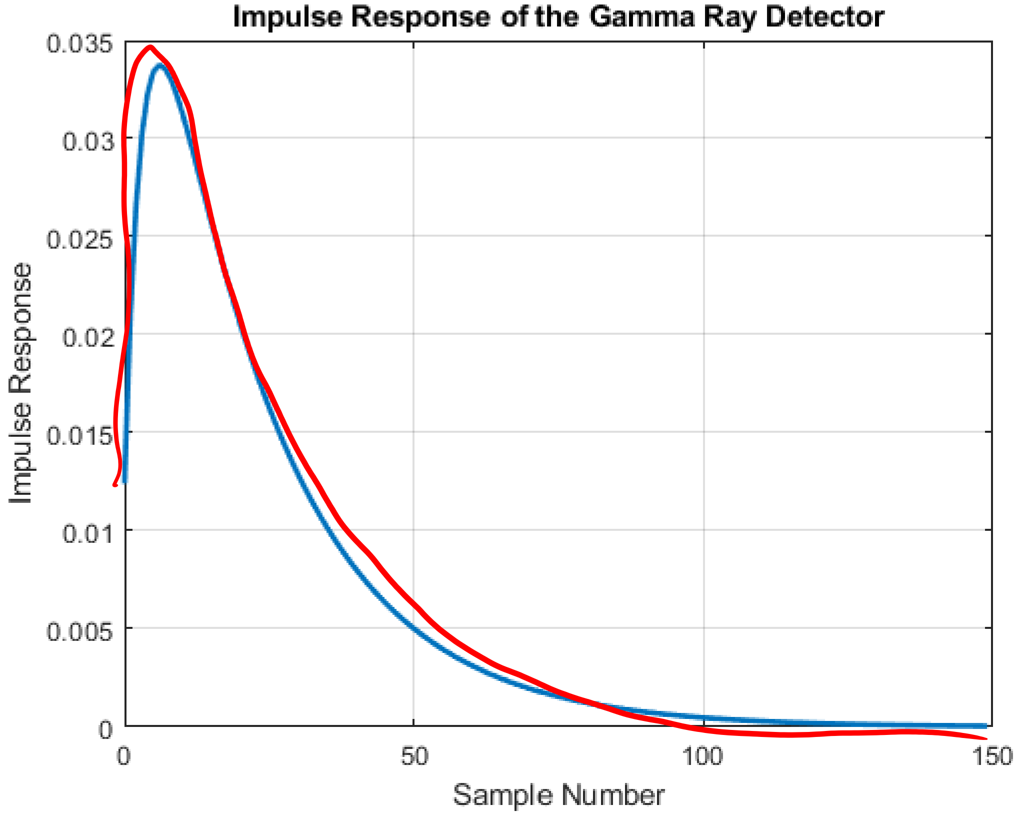
Gamma Ray Detector

- Gamma Rays are impulses
- The scintillator detects the impulses and puts out a signal
- The output of the detector spreads out the pulse in time



Impulse Response of the Gamma Ray Detector

- Single sided exponential with some rounded edges

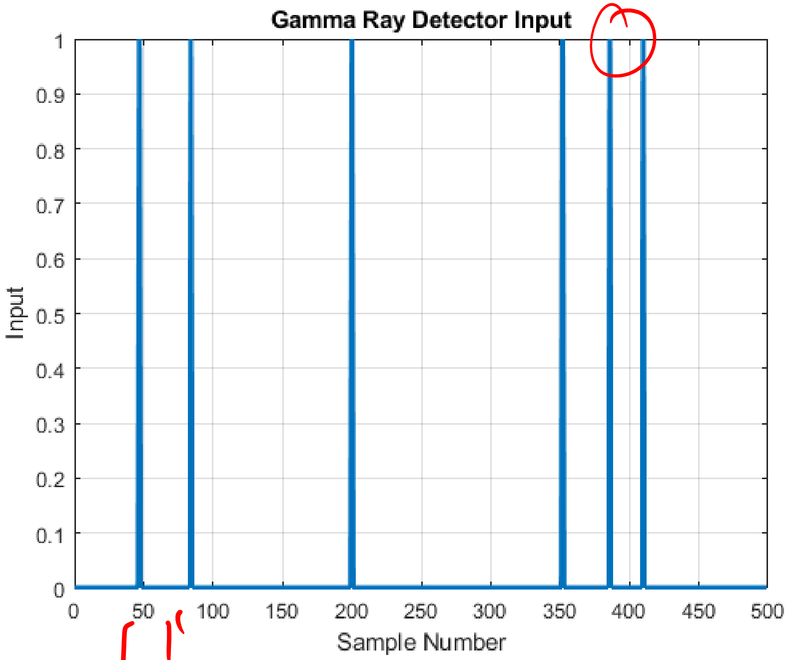


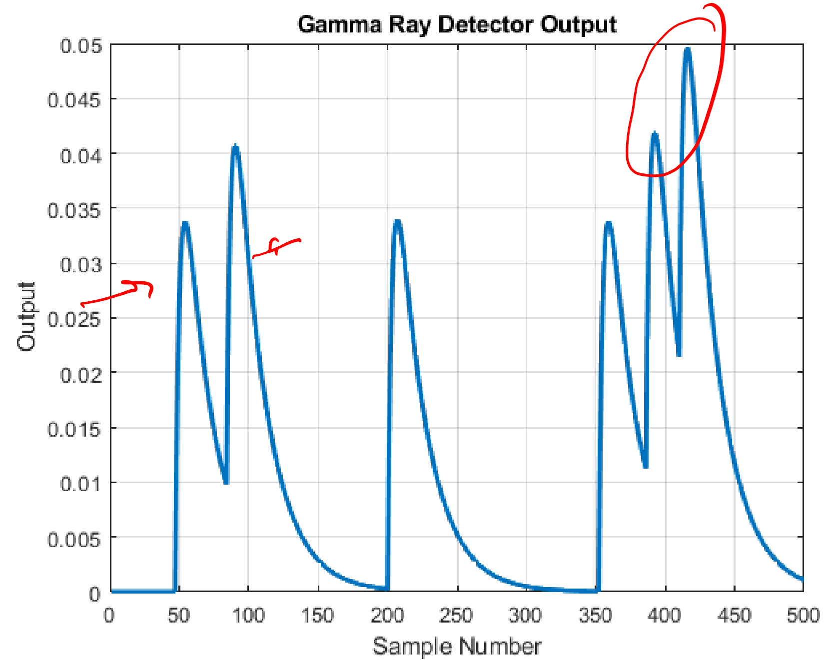
This was created using two single pole IIR filters in series

One with a very low frequency corner (.0075) and one with a higher corner frequency (.05)

Response to Multiple Impulses

- The outputs of the detector overlap one another





Fixing the Response

- Can we create an inverse filter to remove the impact of the detector?
- Since we know the impulse response of the filter this is possible!
  - Deconvolution
- If we don't know the impulse response the problem becomes much harder
  - Blind deconvolution

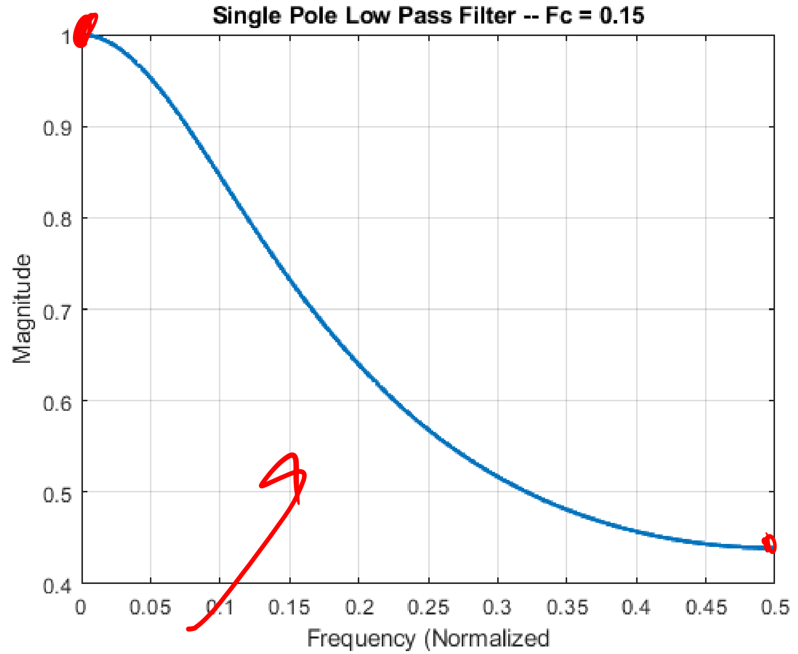
Deconvolution

- Deconvolution is easier to understand in the frequency domain
- Let's say I have a filter frequency response that I want to find the inverse of
- The inverse frequency response would have a magnitude for each sample that is the reciprocal of the original frequency response
- The phase of the inverse frequency response would have the opposite sign of the original response

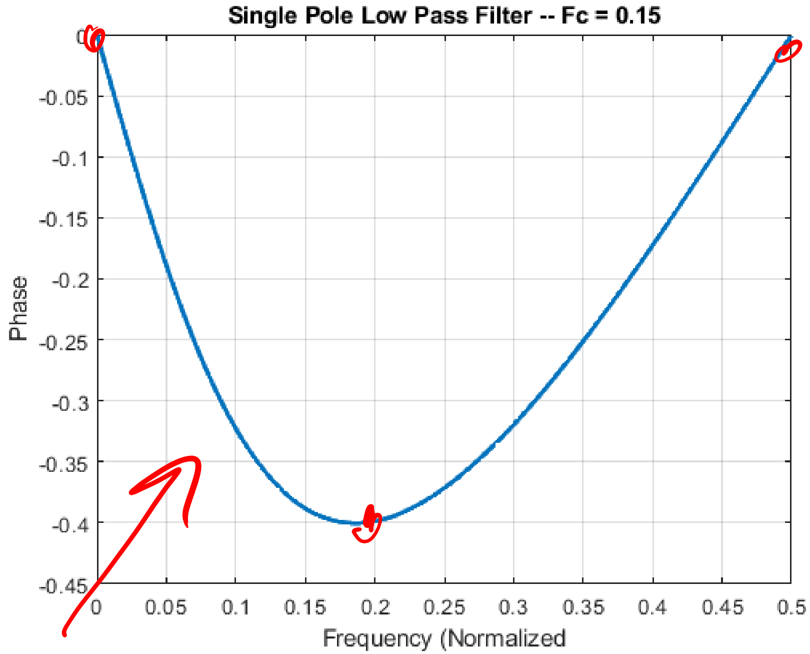
Example – Single Pole LPF

- Find the inverse filter for a single pole LPF

Magnitude

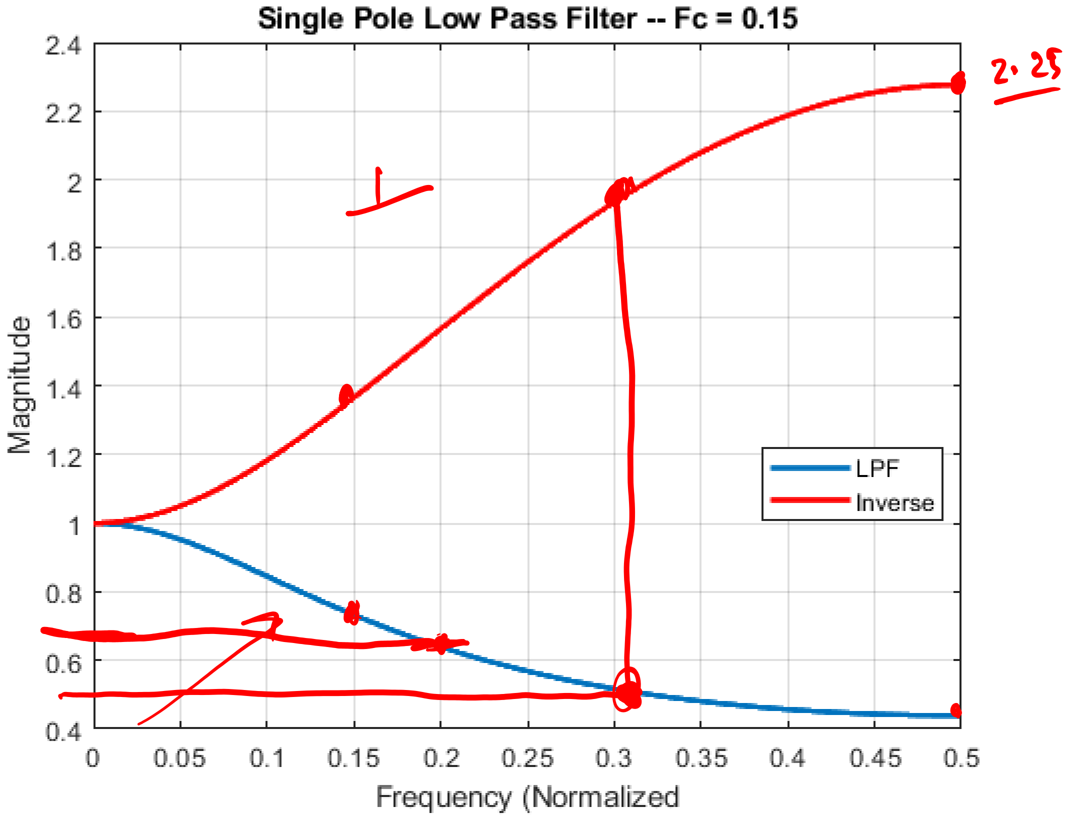


Phase



Example – Single Pole LPF

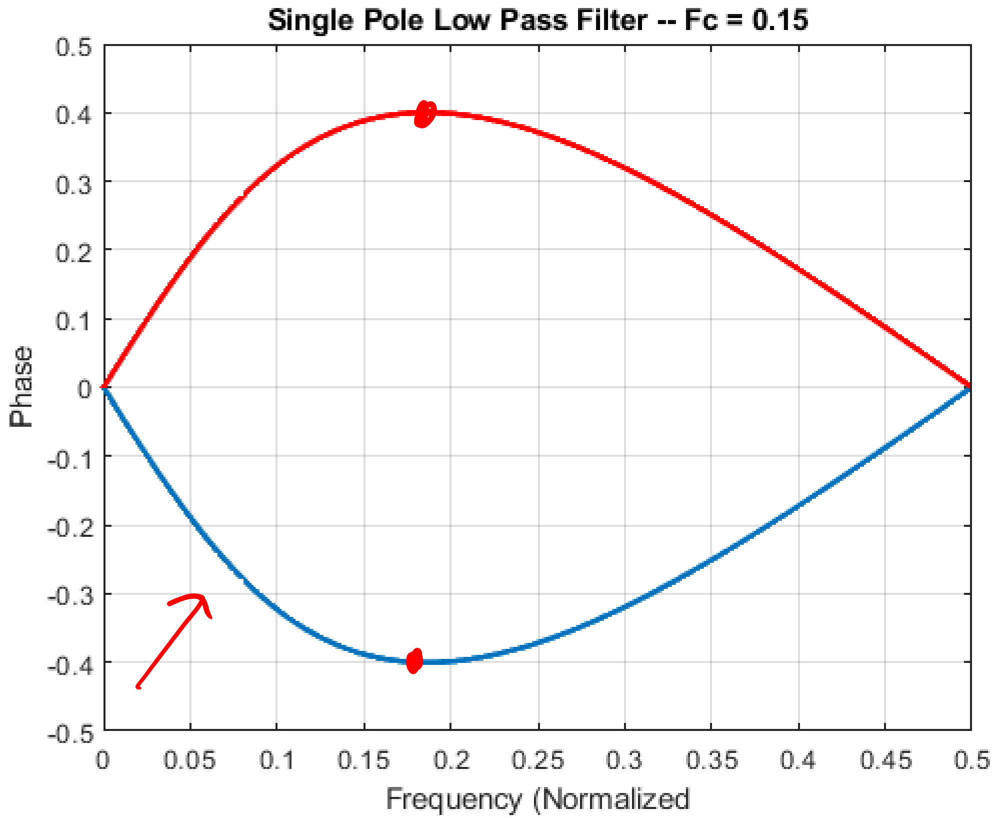
- The magnitude of the inverse filter is the reciprocal of the magnitude of the LPF



Magnitude

Example – Single Pole LPF

- The phase of the inverse filter is negative of the phase of the LPF

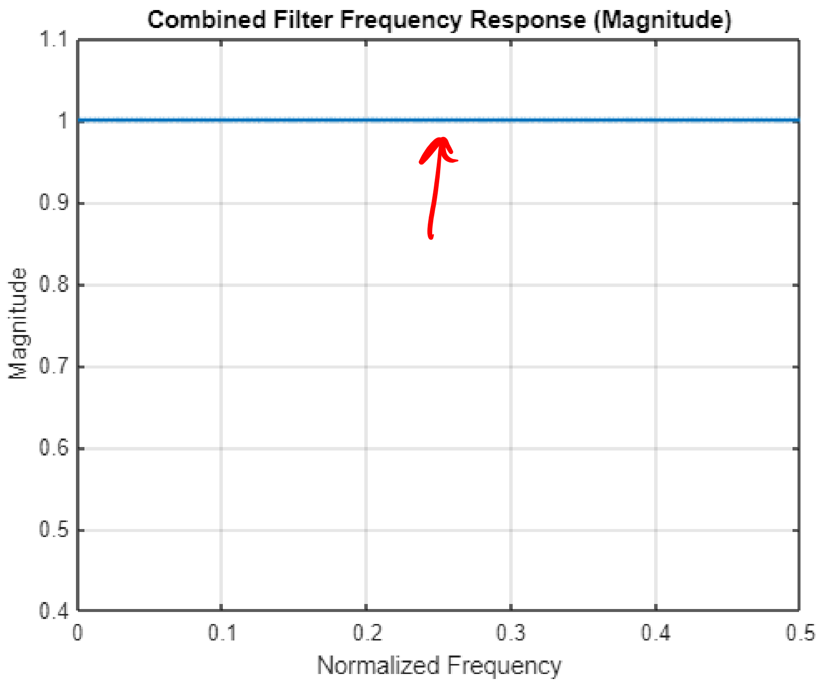


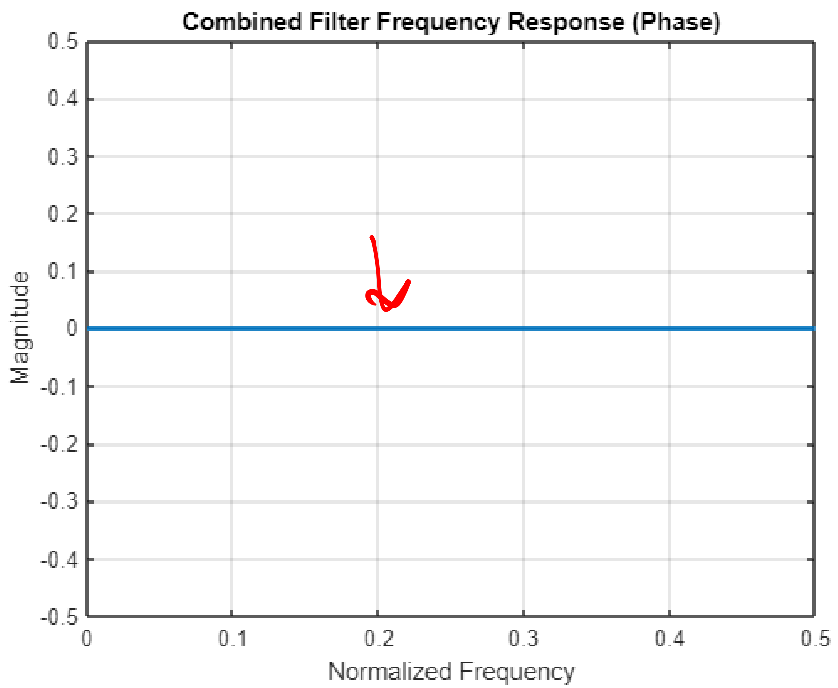
Phase

RIT

Example – Single Pole LPF Combined Filters

- Multiplying the two frequency responses together results in a flat frequency response
- Can you think of an example in lab where we used this concept (but in the time domain)?





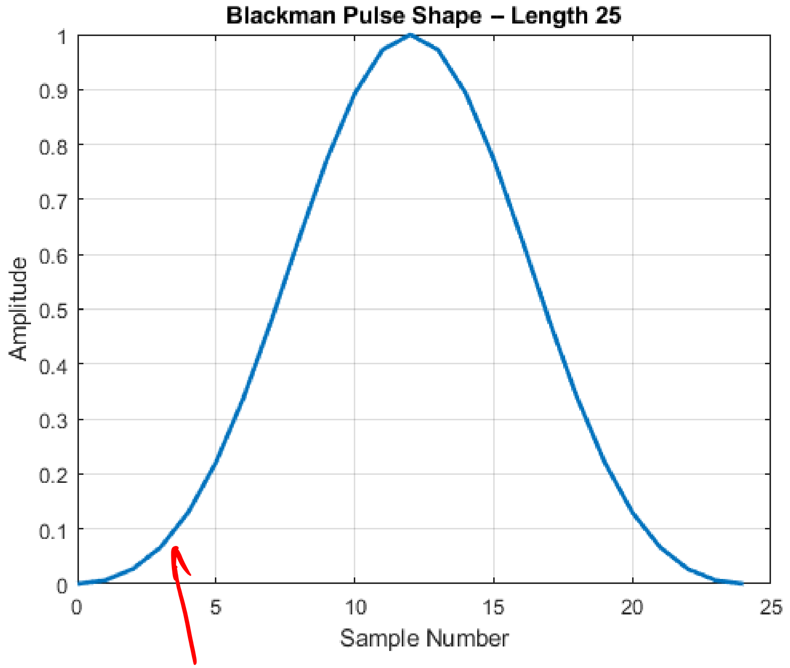
The Gamma Ray Detector Example

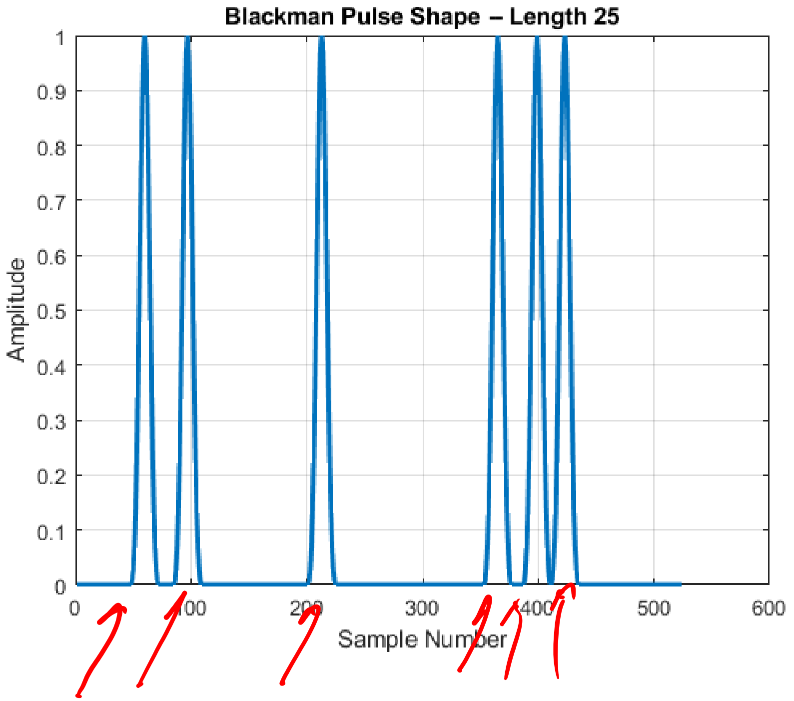
- Ideally we would like to find the inverse filter and retrieve the original impulses but this is not possible
- Let's try to retrieve pulses that are narrower and could be distinguished if close in time

- Use a Blackman window as the pulse shape
- Set its width to ~1/3 of the length of the gamma ray impulse response.

The Gamma Ray Detector Example

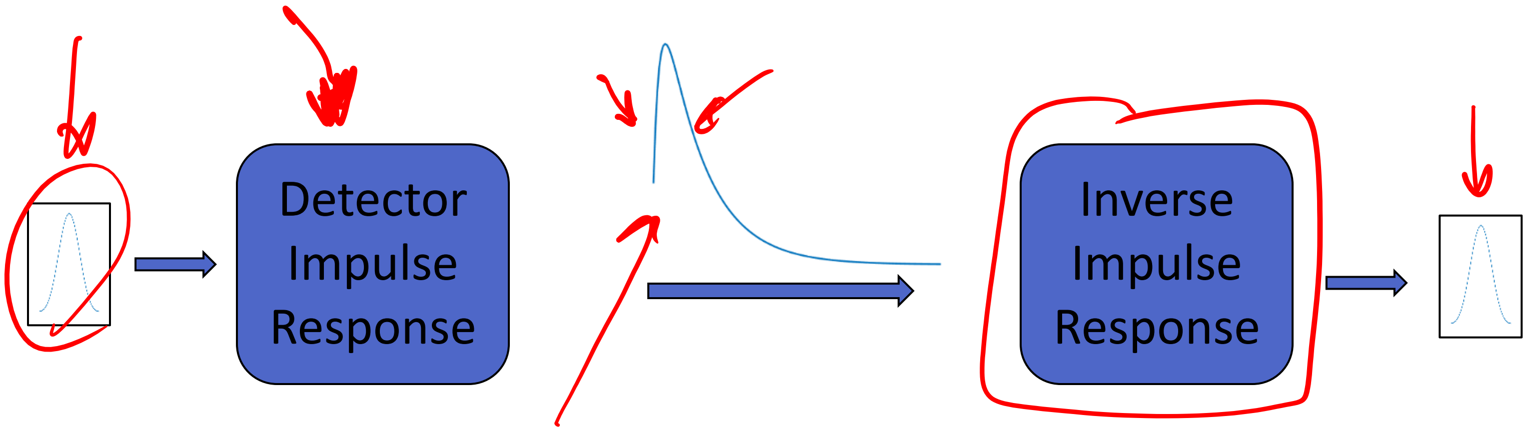
- The desired output is pulses that are distinguishable from one another





Finding the Inverse Impulse Response

- If the impulse response of the system is known then find the frequency response using the FFT
- The desired pulse shape is also known and we can find its frequency response using the FFT



Finding the Inverse Impulse Response

- Using frequency domain convolution to find the inverse filter frequency response

Input b → Detector Impulse Response f[n] → Output d

Time Domain

$b[n] * f[n] = d[n]$

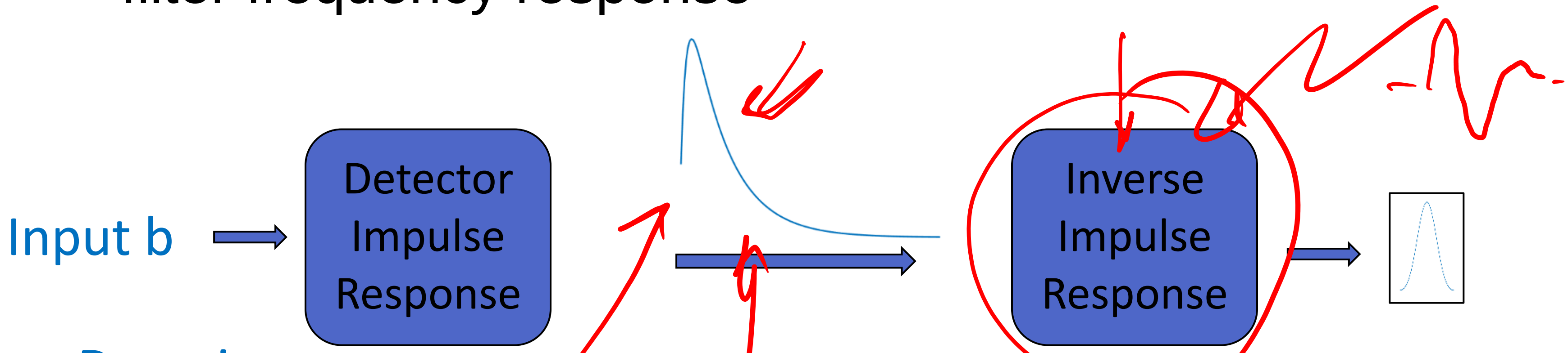
Frequency Domain

$B[k] \times F[k] = D[k]$

Find the inverse of $F[k]$ from $F^{-1}[k] = B[k]/D[k]$

Finding the Inverse Impulse Response

- Using frequency domain convolution to find the inverse filter frequency response



Time Domain

$$\boxed{b[n] * f[n]} = d[n] \qquad d[n] * f^{-1}[n] = b[n]$$

$$B[k] \times F[k] = D[k] \qquad D[k] \times \boxed{F^{-1}[k]} = B[k]$$

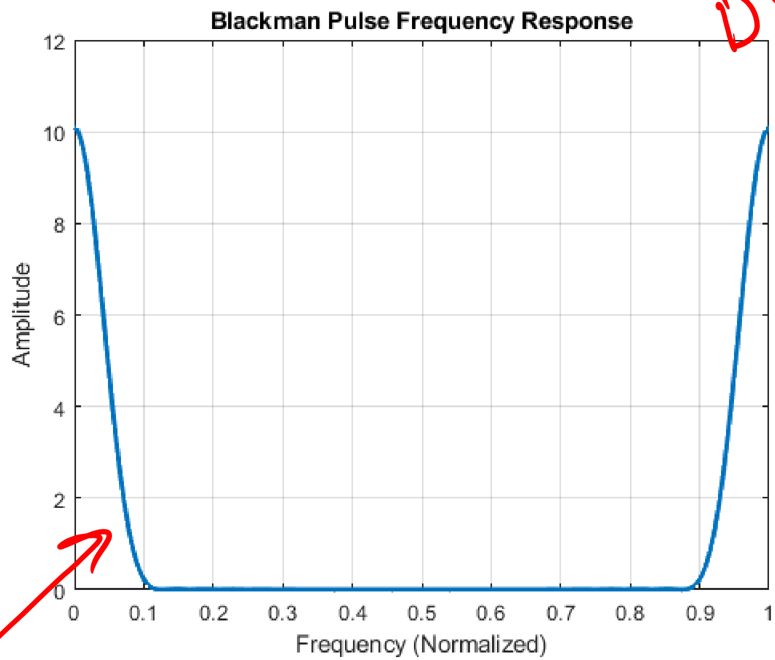
Frequency Domain

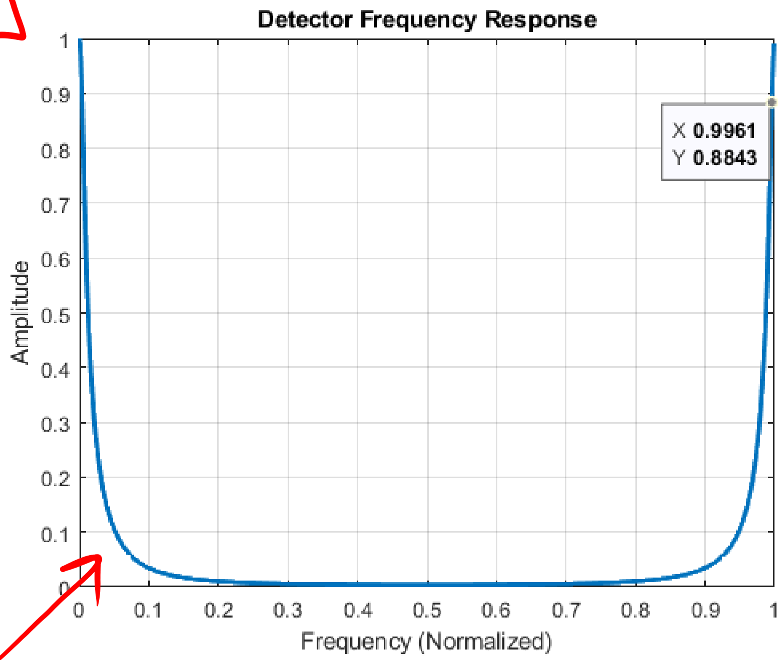
Find the inverse of $F[k]$ from $F^{-1}[k] = B[k]/D[k]$

Pulse and Detector Frequency Responses

- Individual frequency response B[k] and D[K]

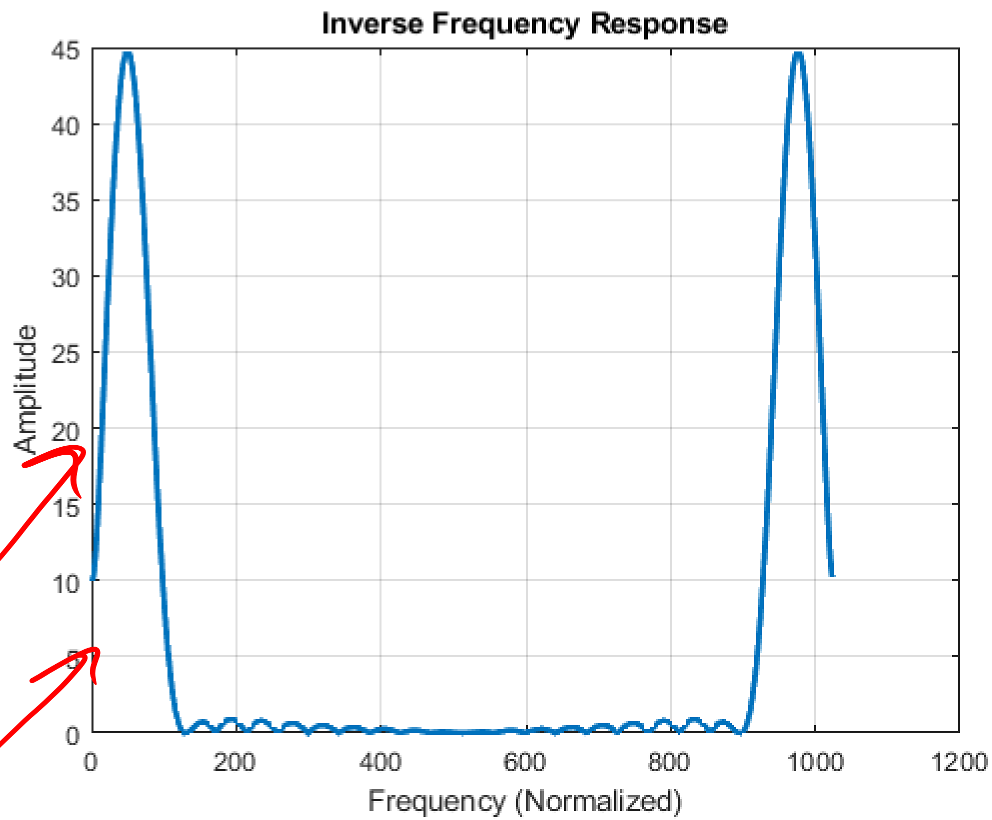
$$\frac{B[k]}{D[k]}$$





Inverse Filter Frequency Response

- Perform point by point division of the two frequency responses



$$B[k] \times F[k] = D[k]$$

$$F^{-1}[k] = B[k]/D[k]$$

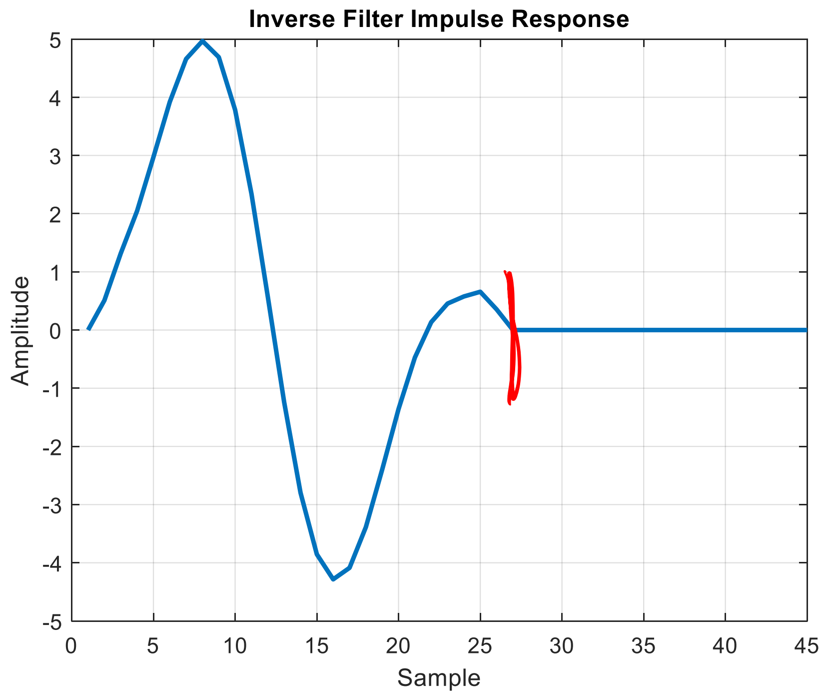
```
deconvFFT = pulseFFT ./ detFFT;
```

Use ./ in MATLAB

Inverse Filter Impulse Response

- Find the impulse response of the inverse filter by taking the IFFT as when making a custom filter

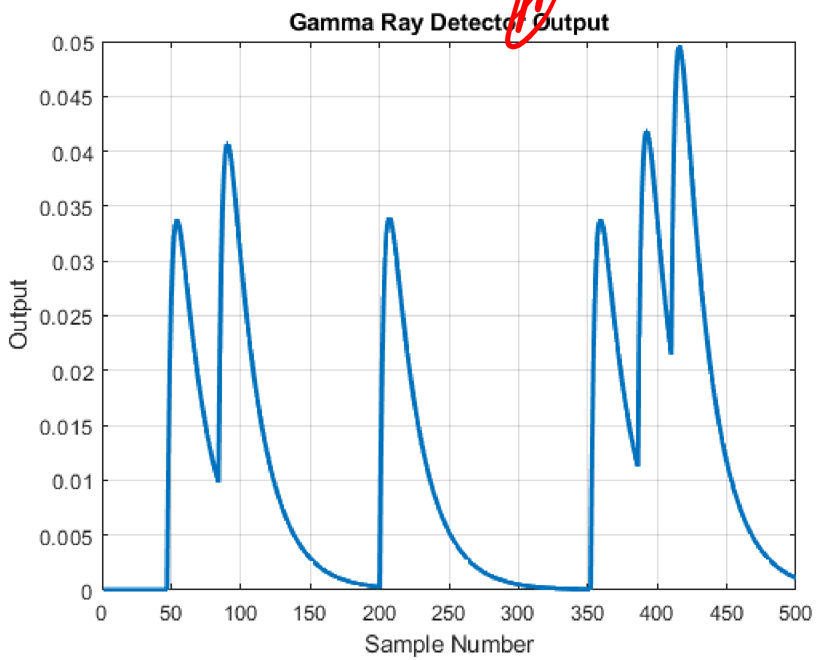
- Shift, truncate and window the response

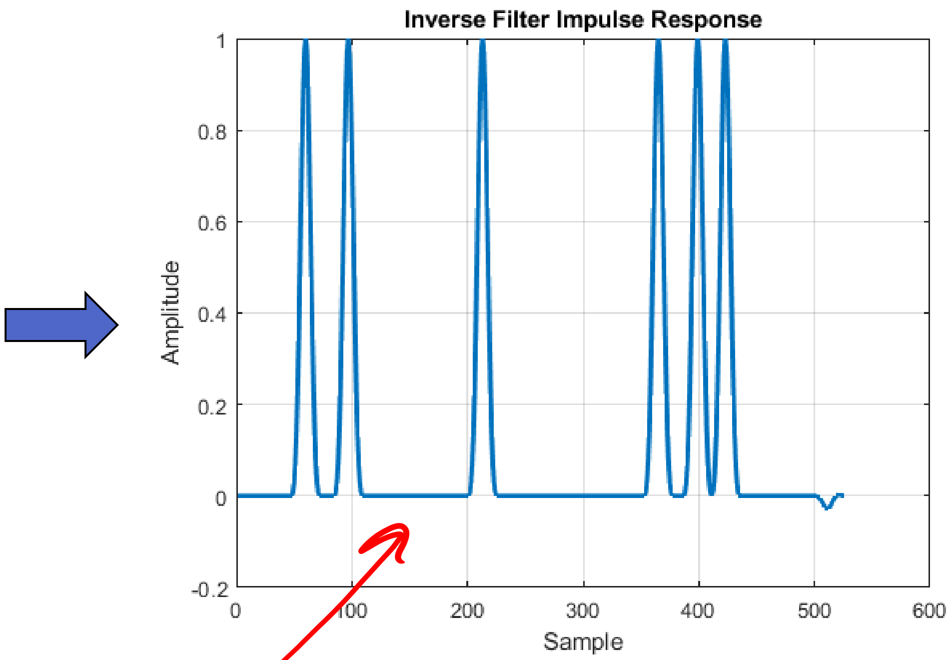


It wasn't necessary to shift or window this response as it was already shifted and had no abrupt transitions

Deconvolved Result

- Deconvolution has improved the pulse response





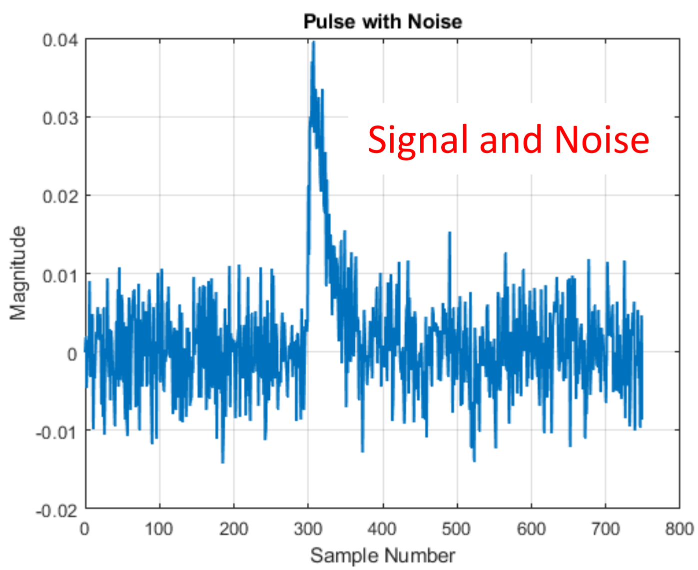
Blind Deconvolution

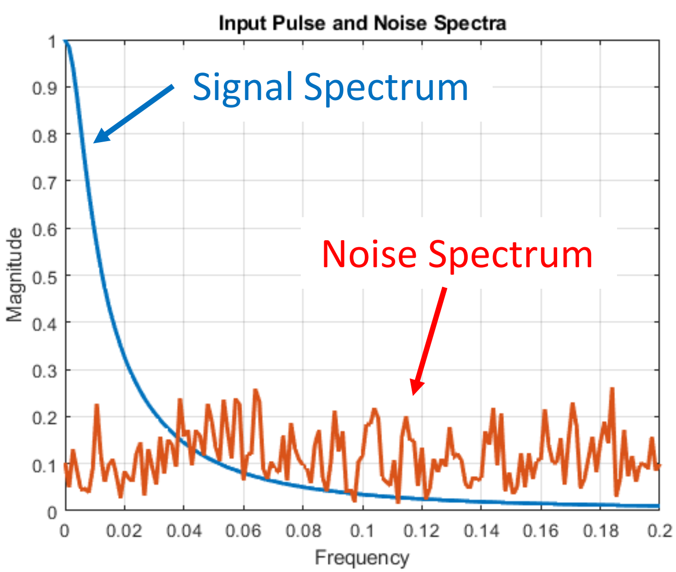
- In some cases, the exact impulse response that transforms the signal is not known.

- In these cases, the de-convolution process is “blind”. It must operate based on a best guess as to what the undesired convolution was, and then try to remove its effect.
- Example – A room’s echo may have affected a recording. If the room is no longer available, there is no way to measure its frequency response.
- Instead, the frequency response of the room must be estimated from the original recording and perhaps other recordings of the same music in a room with no echoes.

Application – Optimal Filtering

- A common problem is to try to extract a signal that is buried in noise.
- If the spectrum of the noise and the spectrum of the signal do not overlap, then this can be easily done in the frequency domain
- However, if the spectrum of the noise and signal do overlap, the problem is more difficult.
- In this case the noise is white and the signal is an exponential pulse.





Application – Optimal Filtering

- Three types of filter are discussed
  - Moving Average Filter
  - Matched Filter
  - Wiener filter

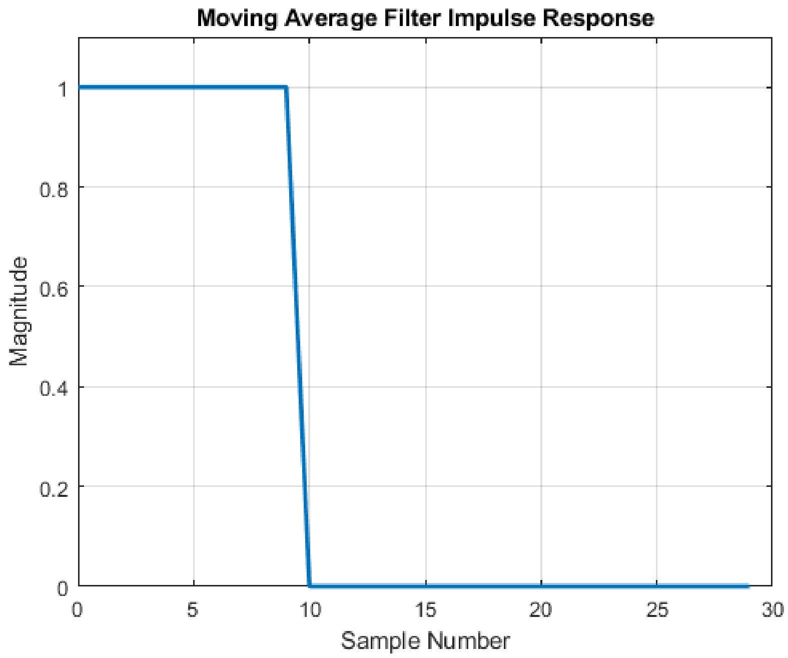
- Moving average filter provides the fastest step response for a given amount of random noise reduction.

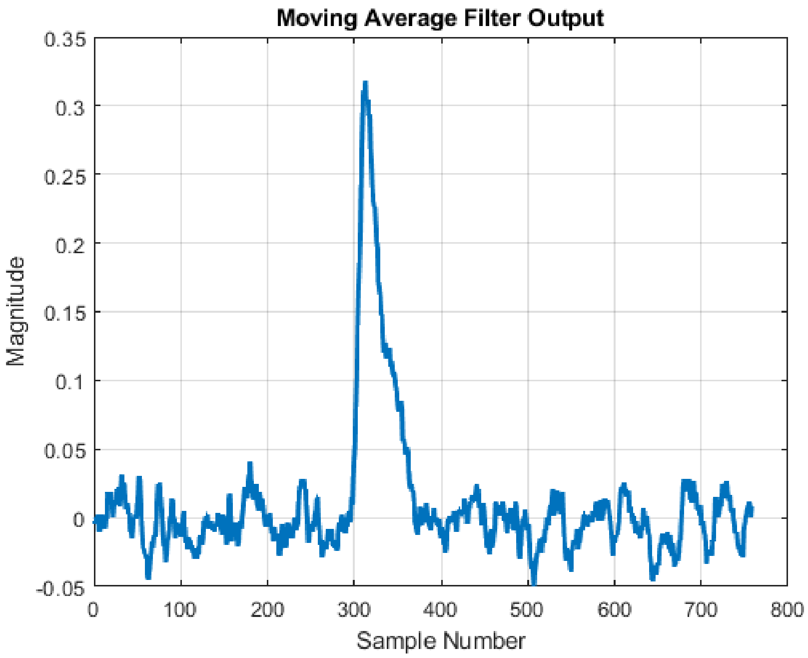
- The Matched Filter provides the highest peak response above the noise.

- The Wiener provides the highest signal to noise ratio at each frequency by varying the gain at each frequency

Optimal Filter
Moving Average Filter

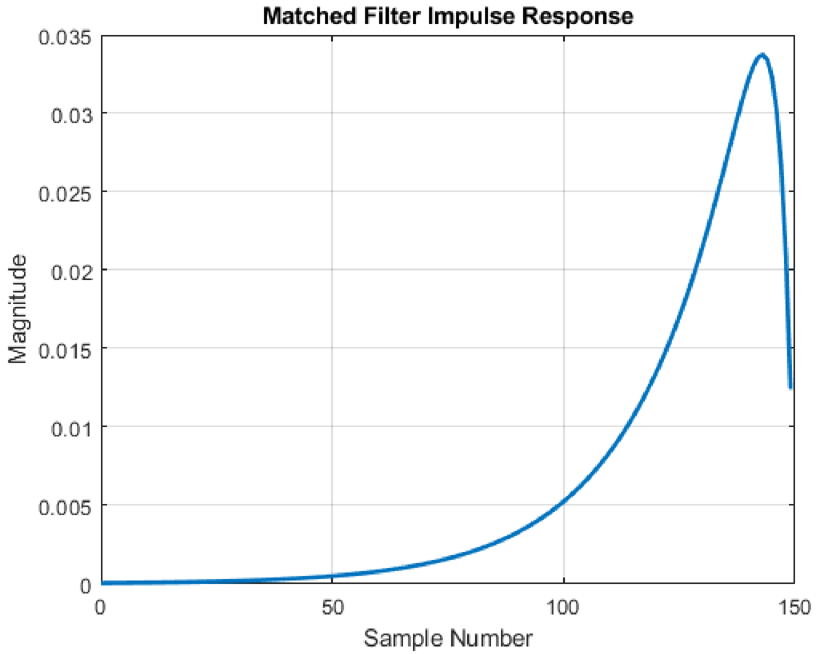
- Impulse response is all 1's
- Fastest step response for given noise reduction
- Easy to implement. Can be fast, recursive

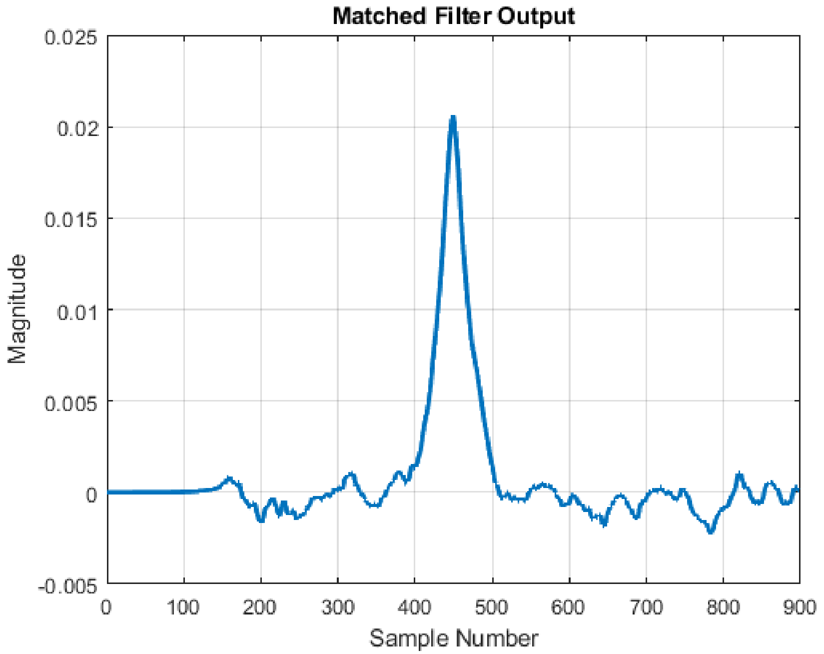




Matched Filter

- A matched filter is made by time reversing the shape of the pulse. This is the filter impulse response
- Peak is highest above the residual noise

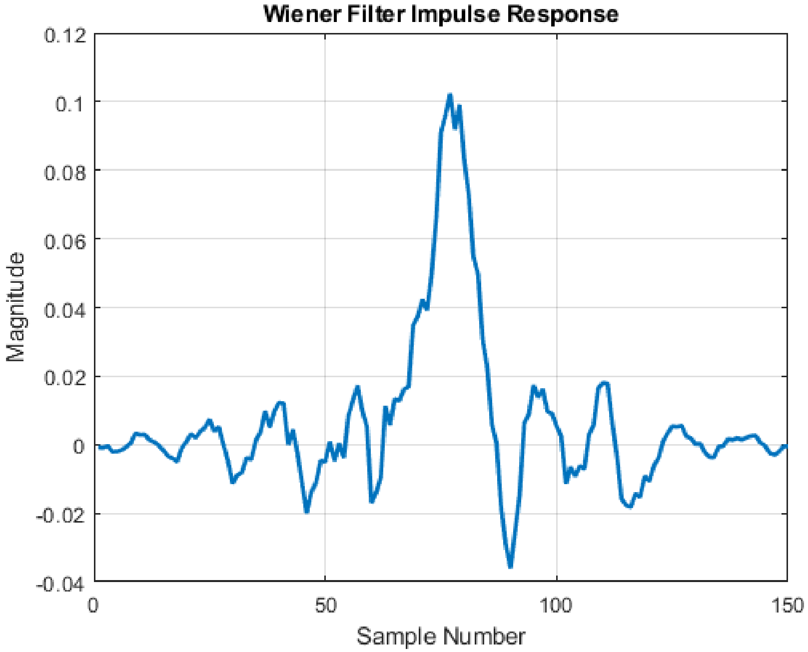


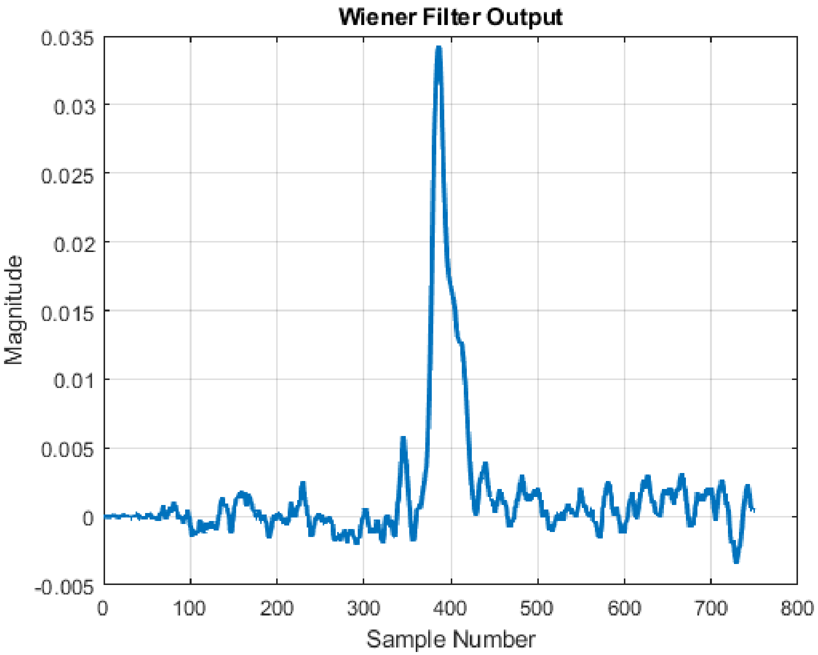


Wiener Filter

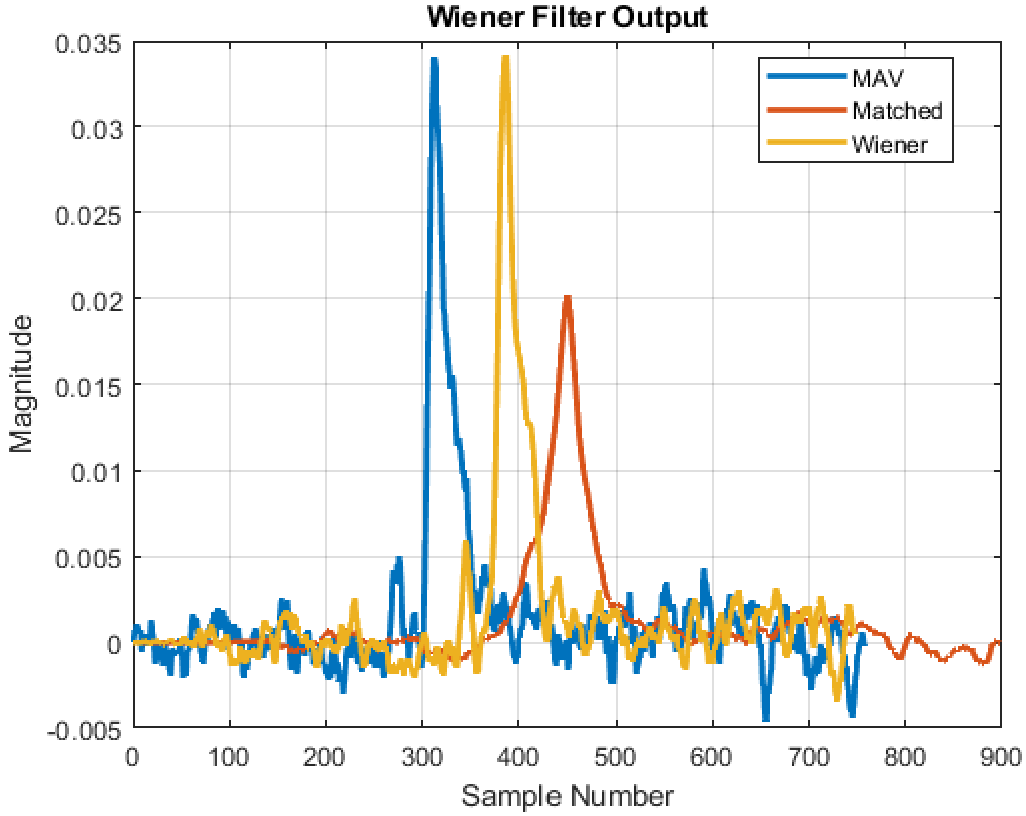
- The Wiener filter optimizes the SNR
- Computed using a reference signal and the input signal with noise

$$H[f] = \frac{S[f]^2}{S[f]^2 + N[f]^2}$$





Filter Comparison



Optimal Filtering Comments

- The difference in performance may be small between filters.
- The computation times /cost/ complexity can be significantly different.
- In certain cases, the use of an optimal filter may provide sufficient benefit to make it worth considering, particularly when there are no other options to boost signal to noise ratio.