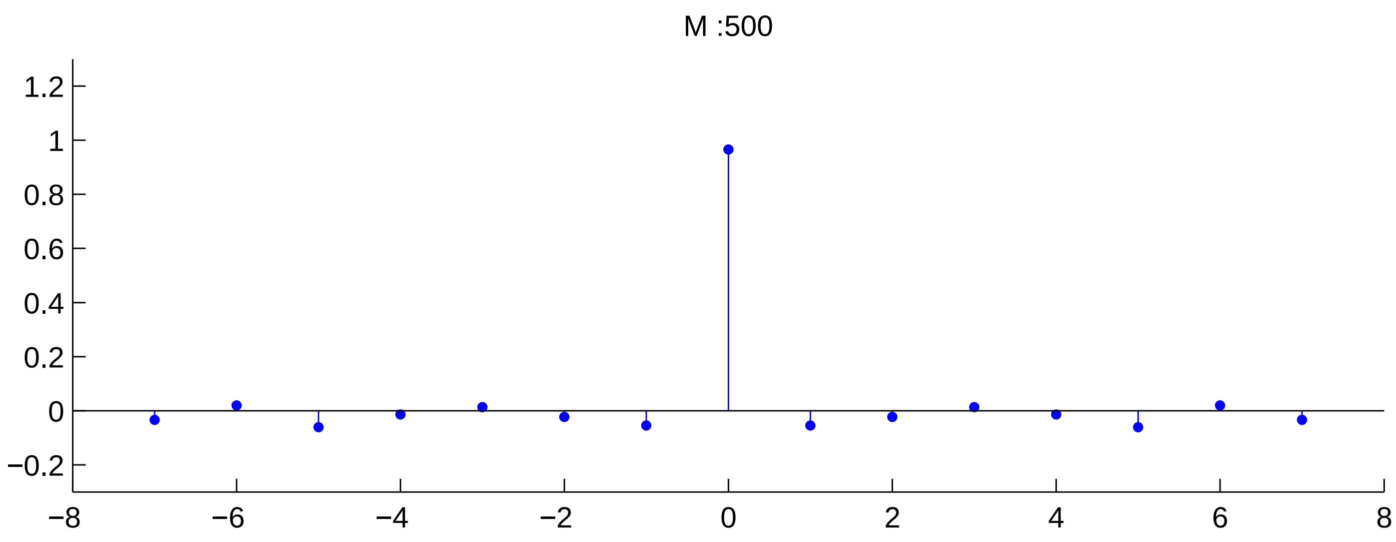
BYM 510E – Biomedical Signals Processing

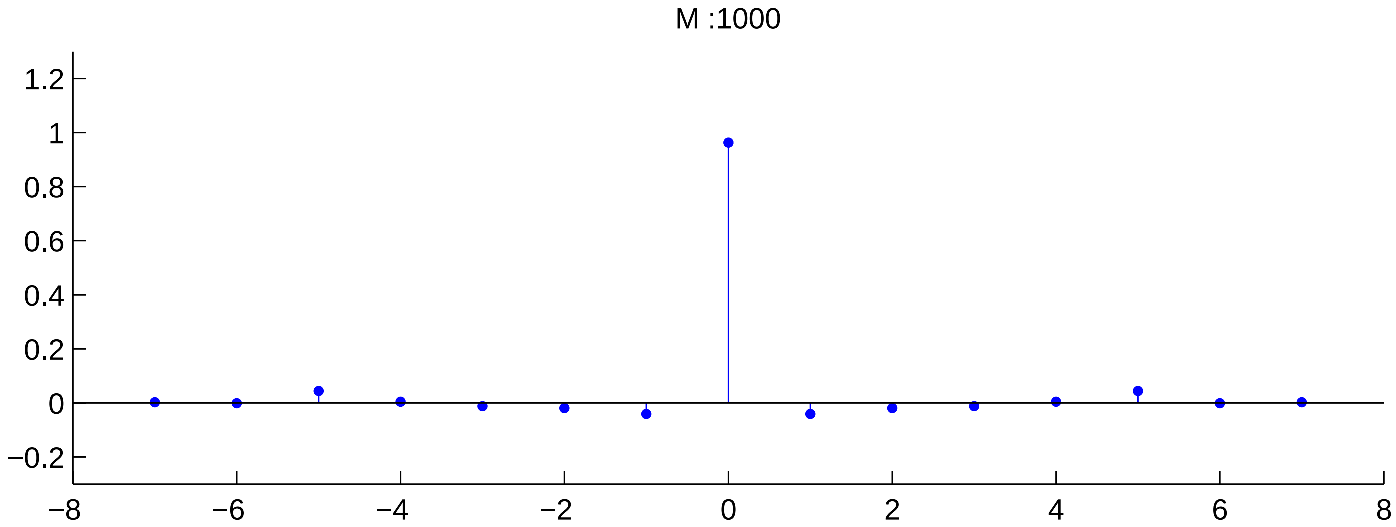
İlker Bayram

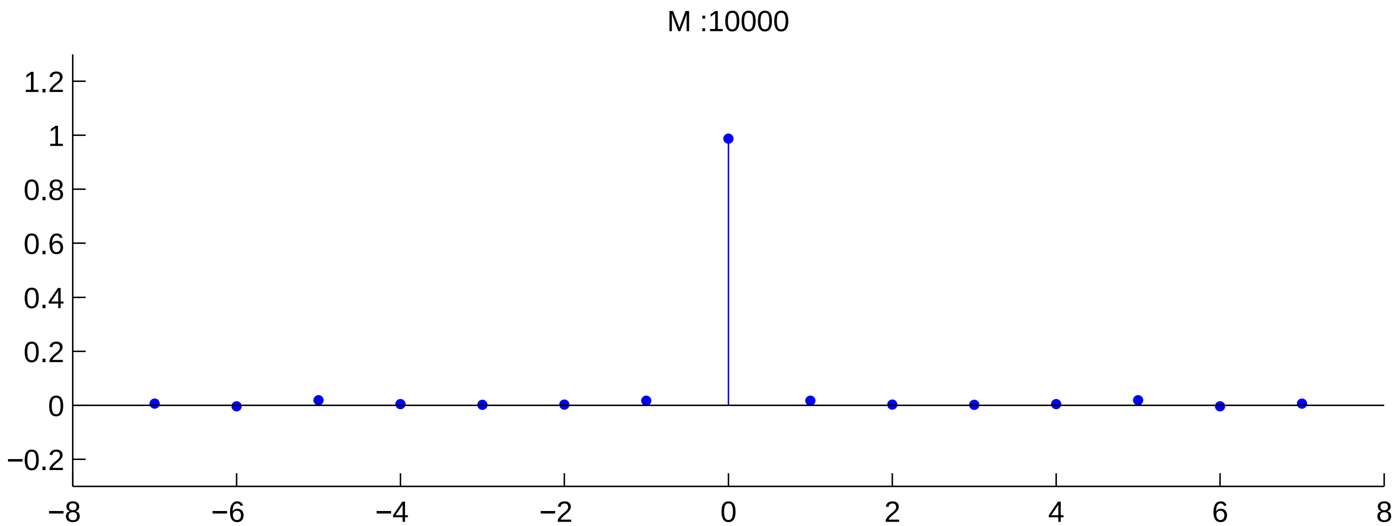
Autocorrelation Function

```
%create signal
M = 500; % length of the signal
x = randn(1,M); %Gaussian white noise with unit variance

ax = xcorr(x) / M; % the biased estimate of the autocorrelation function
L = 10; % determines the size of spectrum estimate
est = ax((end+1)/2 - L :(end+1)/2 + L);
stem(est,'.');
```



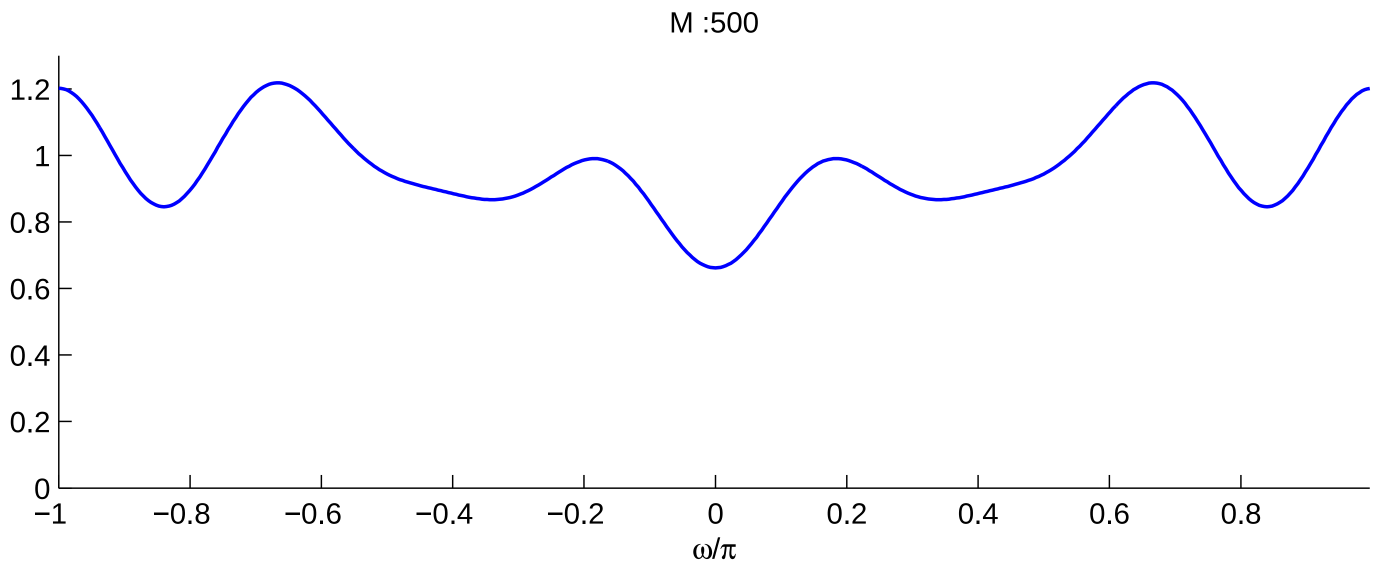
Autocorrelation Function



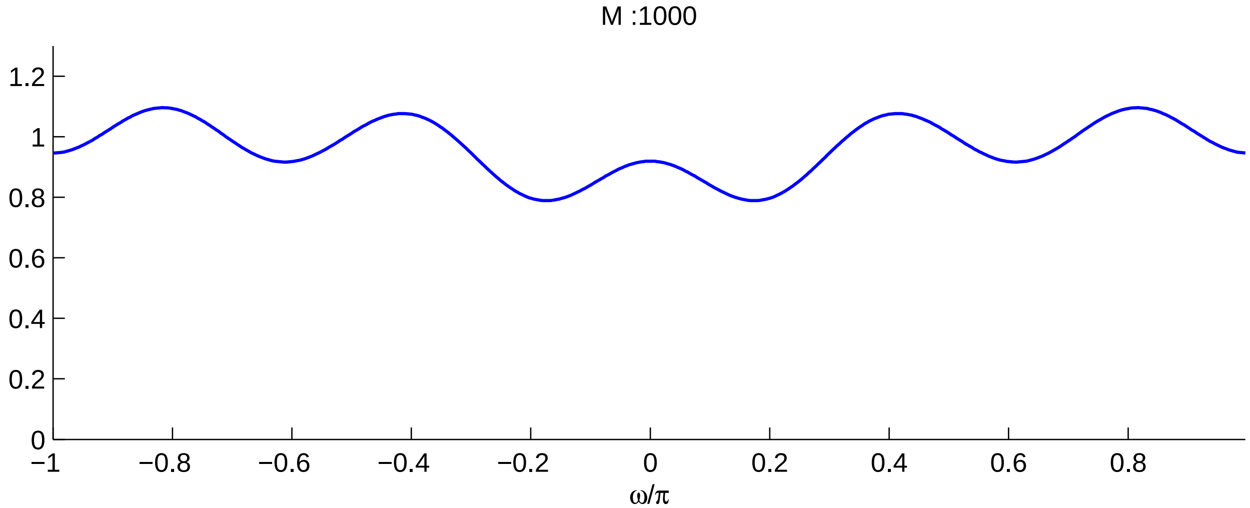


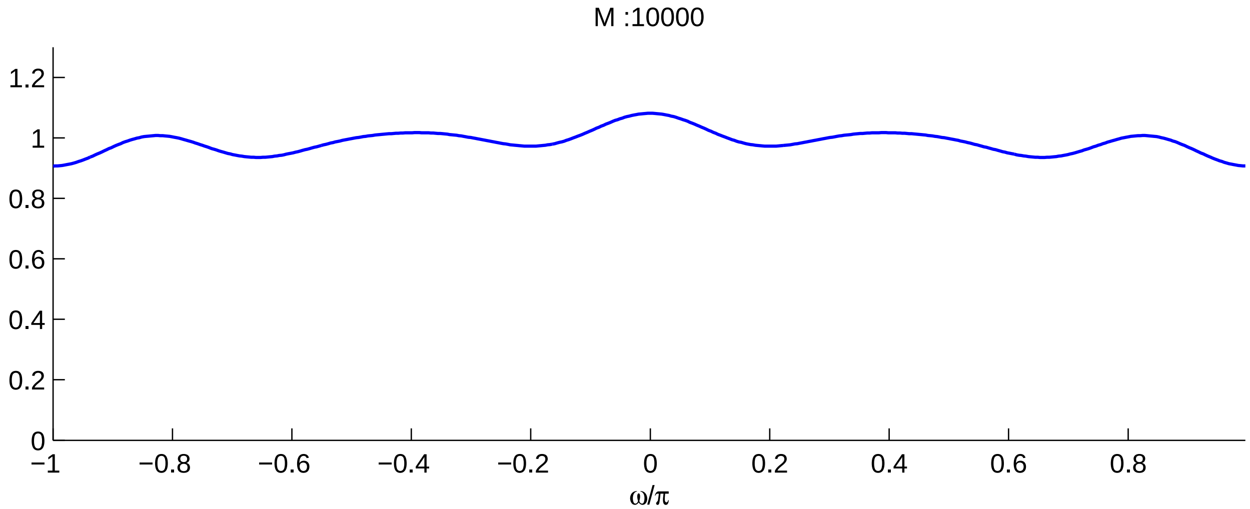
Spectrum Estimation : The Correlogram

```
Sx = fft(est,512); % this is the Correlogram
Sx = fftshift(abs(Sx));Sx = Sx(:);
w = 2*(0:511)/512 - 1; w = w(:);
plot(w,Sx);
```



Spectrum Estimation : The Correlogram





Spectrum Estimation

Colored Noise :

$$x \longrightarrow \boxed{H} \longrightarrow y$$

Relation between the autocorrelation functions :

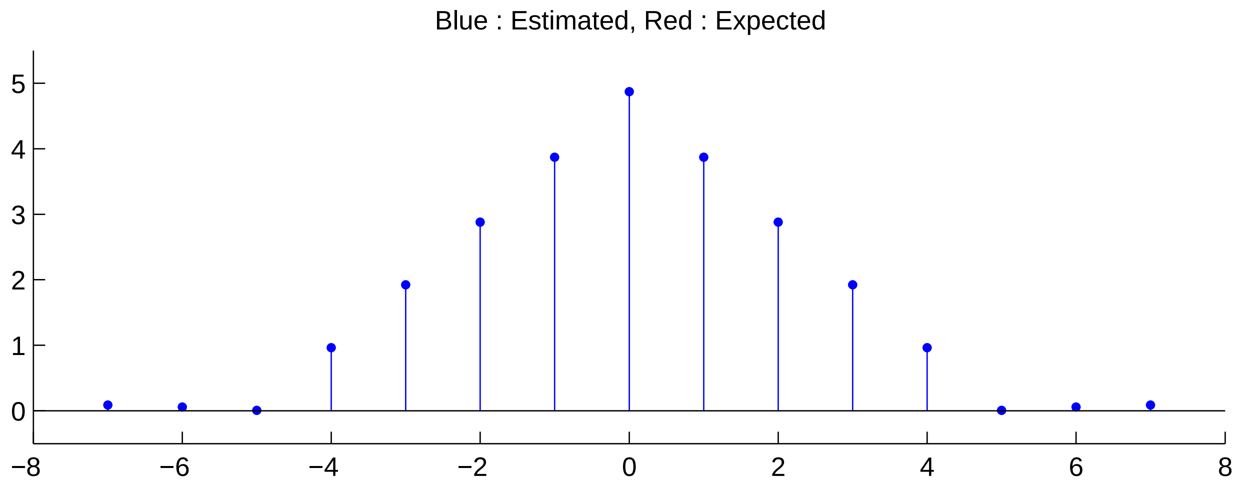
$$R_y(n) = h(n) * h(-n) * R_x(n)$$

or, in the Fourier Domain :

$$S_y(e^{j\omega}) = |H(e^{j\omega})|^2 \, S_x(e^{j\omega})$$

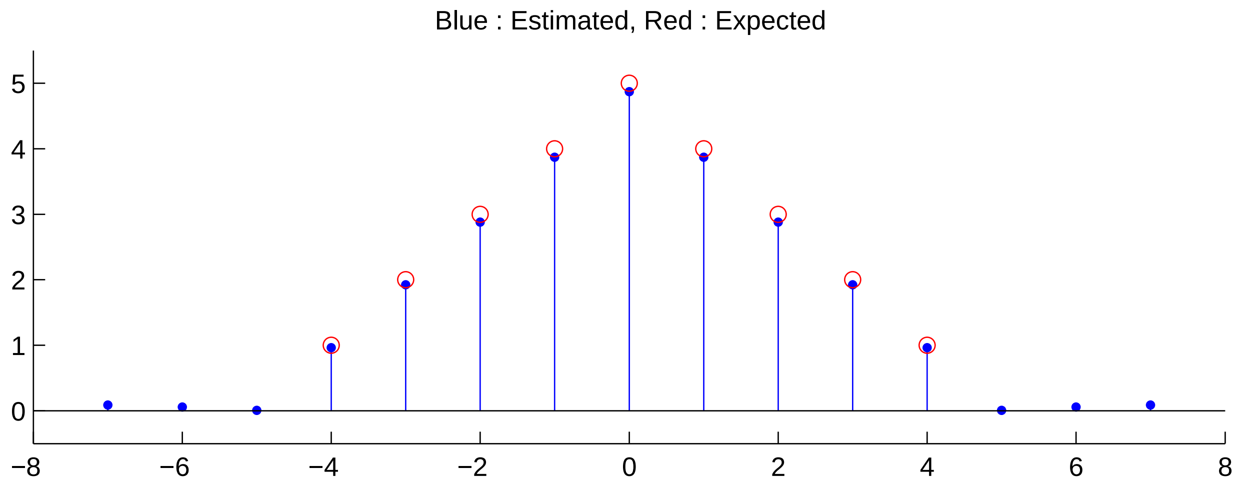
Spectrum Estimation

```
M = 5000; x = randn(1,M); % input Gaussian white noise
K = 5; h = ones(1,K); % filter
y = conv(x,h);

ax = xcorr(y) / M;
L = 7; est = ax((end+1)/2 - L :(end+1)/2 + L); % estimate

hh = xcorr(h); % this is the expected autocorrelation function of y
```



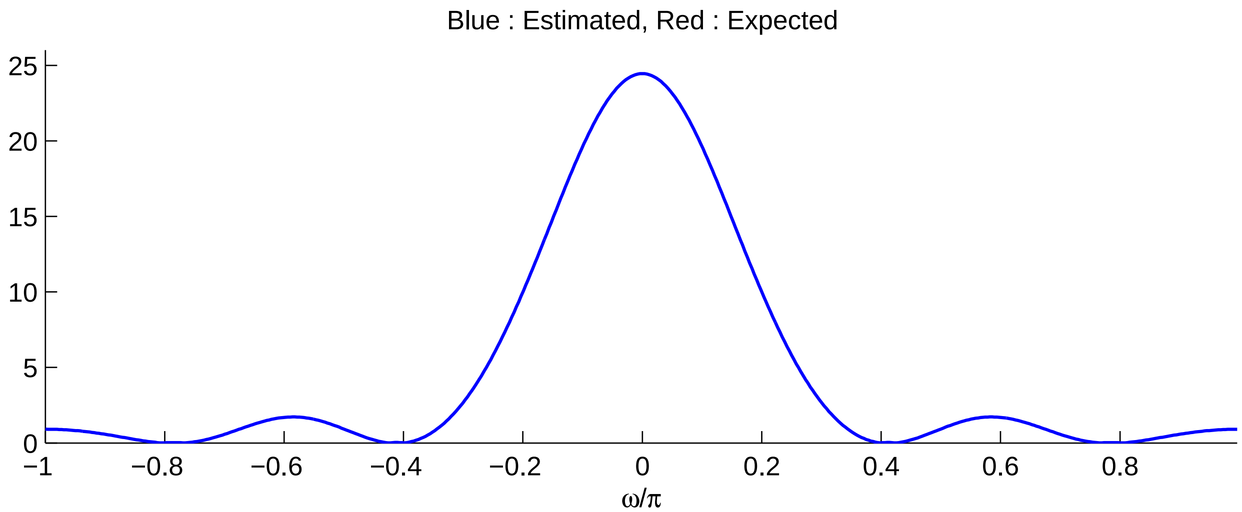
Spectrum Estimation

```
M = 5000; x = randn(1,M); % input Gaussian white noise
K = 5; h = ones(1,K); % filter
y = conv(x,h);

ax = xcorr(y) / M;
L = 7; est = ax((end+1)/2 - L :(end+1)/2 + L); % estimate

hh = xcorr(h); % this is the expected autocorrelation function of y
```



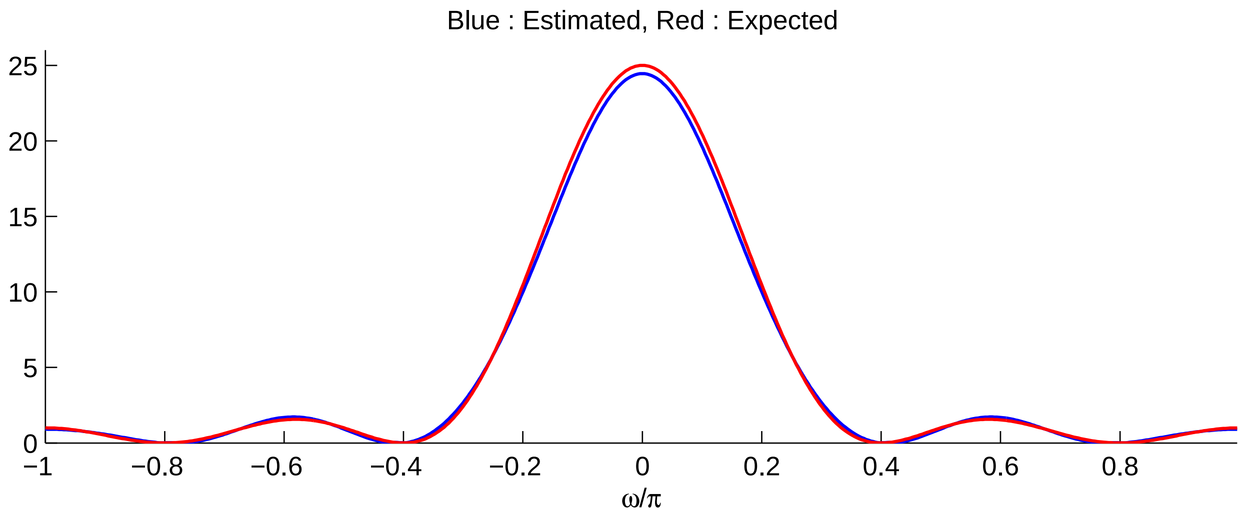
Spectrum Estimation

```
Sx = fft(est,512);Sx = fftshift(abs(Sx)); % this is the estimated spectrum (i.e. the correlogram)

Exp = fft(hh,512);Exp = fftshift(abs(Exp)); % the expected spectrum
```



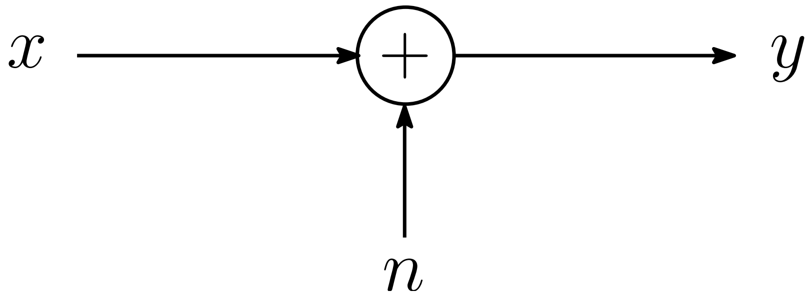
Spectrum Estimation

```
Sx = fft(est,512);Sx = fftshift(abs(Sx)); % this is the estimated spectrum (i.e. the correlogram)

Exp = fft(hh,512);Exp = fftshift(abs(Exp)); % the expected spectrum
```



The Wiener Filter

Denoising Scenario :

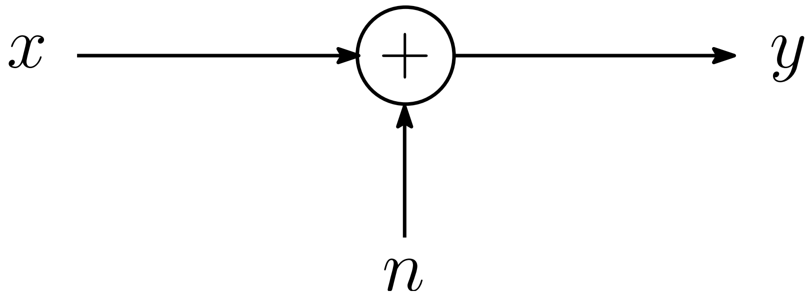


Wiener Filter :

$$W(e^{j\omega}) = \frac{S_x(e^{j\omega})}{S_x(e^{j\omega}) + S_n(e^{j\omega})}$$

Wiener Filter

Denoising Scenario :



Construct the signals

```
g = rand(1,1000); g = sqrt(12)*(g - mean(g)); %uniformly distributed white noise with unit variance

h = hamming(20); % a lowpass filter
x = conv(g,h); % filter with a lowpass to produce 'colored' noise (this is the 'clean signal')

sig = 1;
n = sig*randn(size(y)); % noise

y = x + h; % observation
```

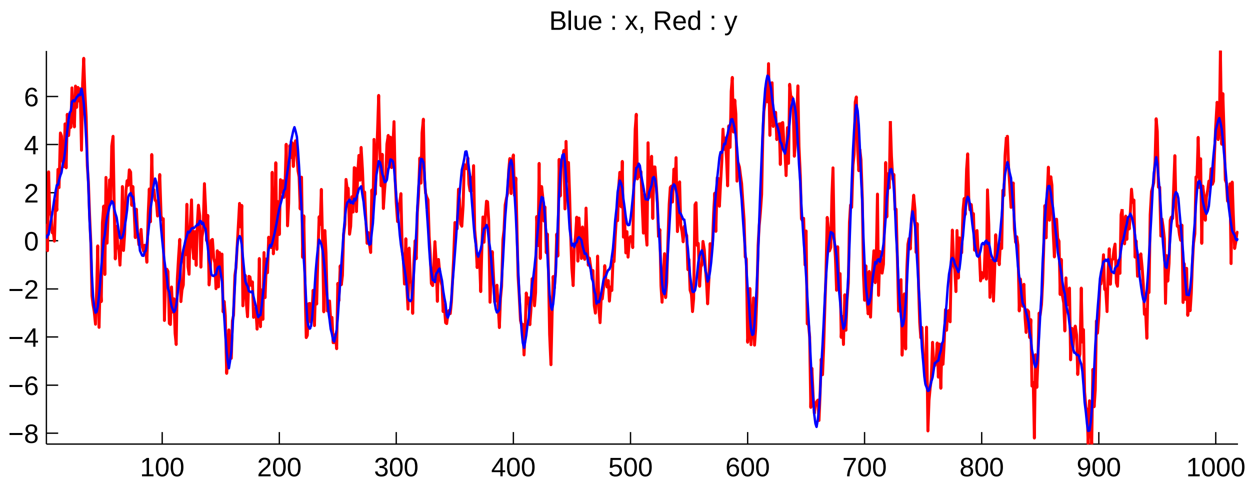
Wiener Filter – Known Spectra

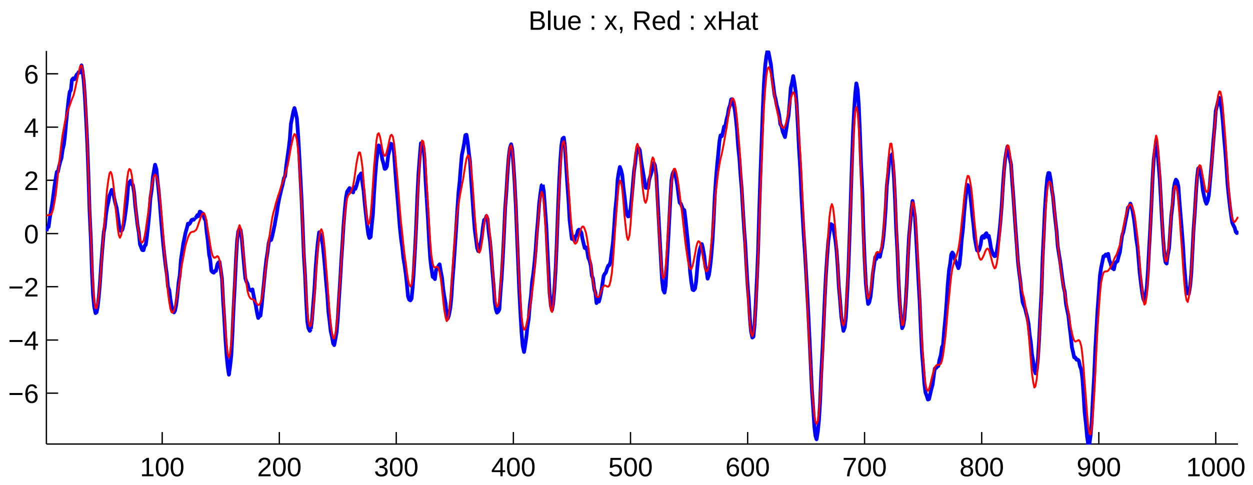
The Wiener filter is easy to perform if one knows the spectra of x and n.

$$W(e^{j\omega}) = \frac{S_x(e^{j\omega})}{S_x(e^{j\omega}) + S_n(e^{j\omega})}$$

```
hh = xcorr(h);
Sx = fft(hh,length(y)); Sx = abs(Sx); % this is the spectrum of x.
Sn = sig*ones(length(y),1); % noise spectrum

filt = Sx./(Sx + Sn); % the Wiener filter

Y = fft(y); Y = Y.';
XHat = Y .* filt ; % FT of the denoised signal
xhat = ifft(XHat); % this is the estimate
```

Wiener Filter – Known Spectra





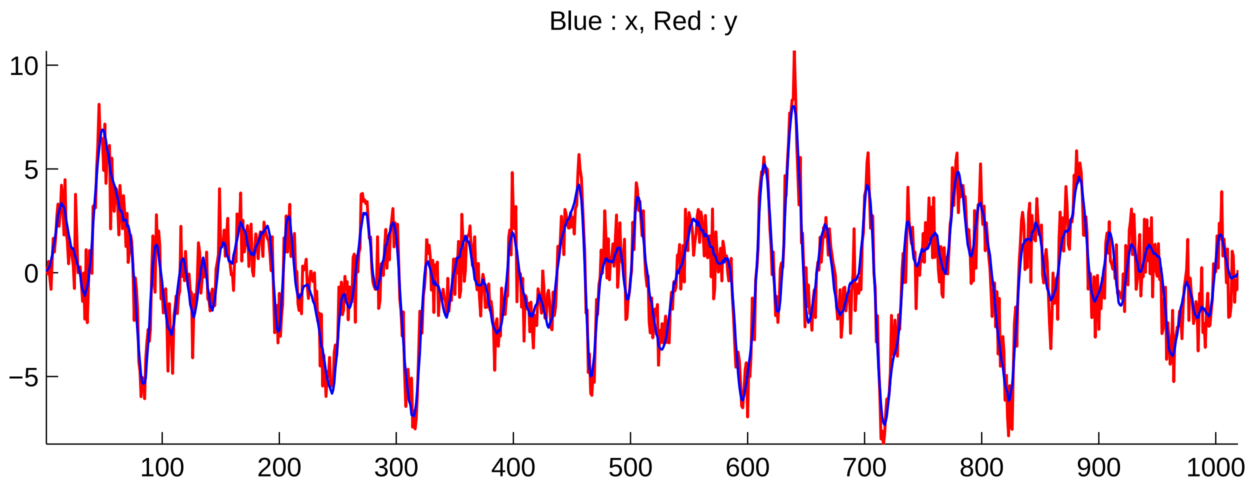
Wiener Filter – Unknown Input Spectrum

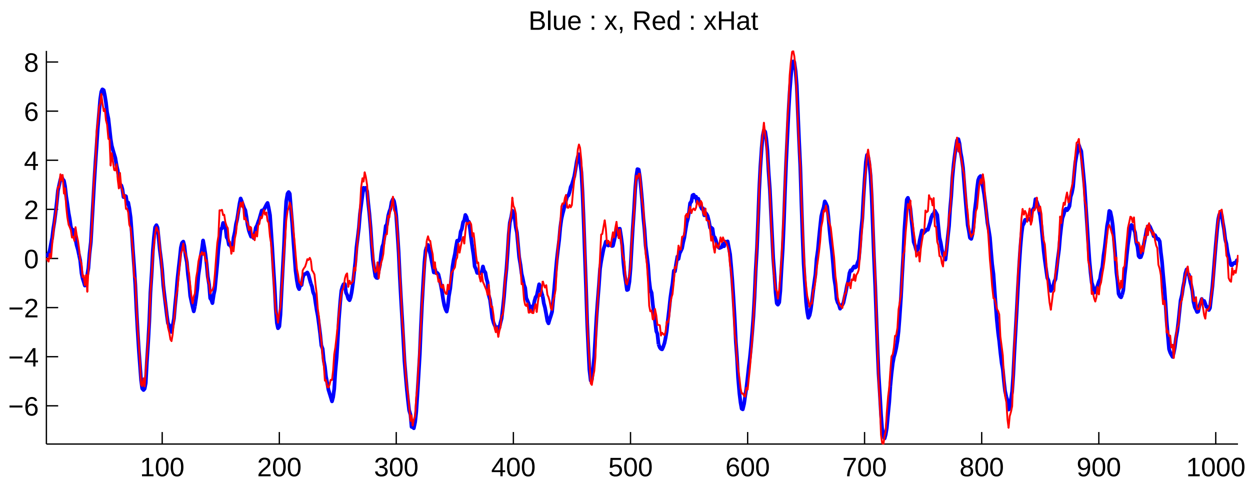
When the spectrum of x is unknown, it can be estimated from the spectrum of y.

$$W(e^{j\omega}) = \frac{S_x(e^{j\omega})}{S_x(e^{j\omega}) + S_n(e^{j\omega})}$$

```
ax = xcorr(y) / length(y);
L = 25; % determines the size of the spectrum estimate
est = ax((end+1)/2 - L :(end+1)/2 + L);hh = xcorr(h);
Sy = fft(est,length(y)); Sy = abs(Sy); % this is the estimated spectrum of y.
Sn = sig*ones(length(y),1); Sn = Sn.';% noise spectrum

filt = (max(Sy-Sn,0))./(Sy); % the Wiener filter in the Fourier domain
```

Wiener Filter – Unknown Input Spectrum





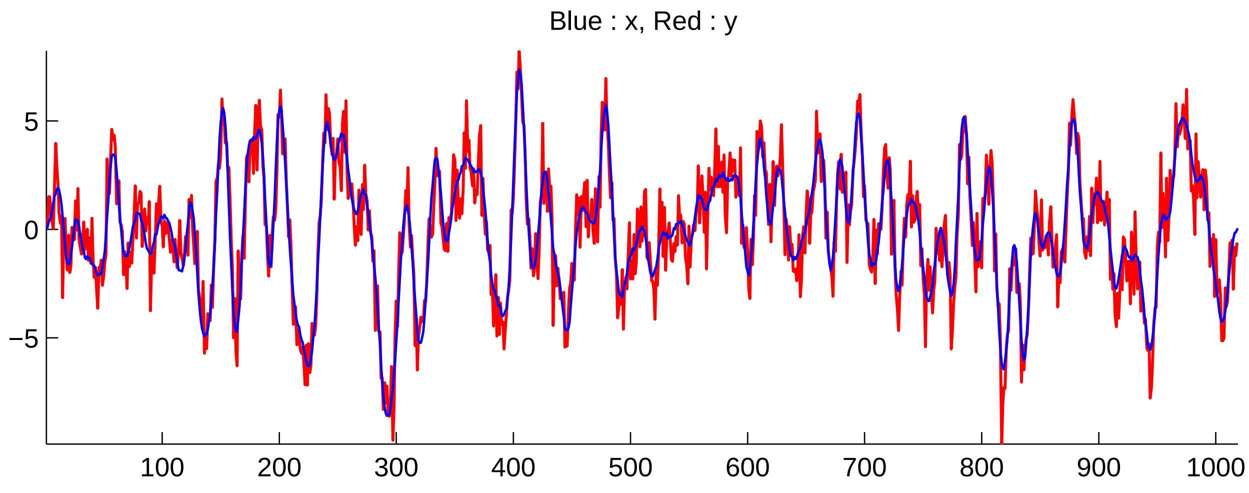
Wiener Filter – Unknown Spectra

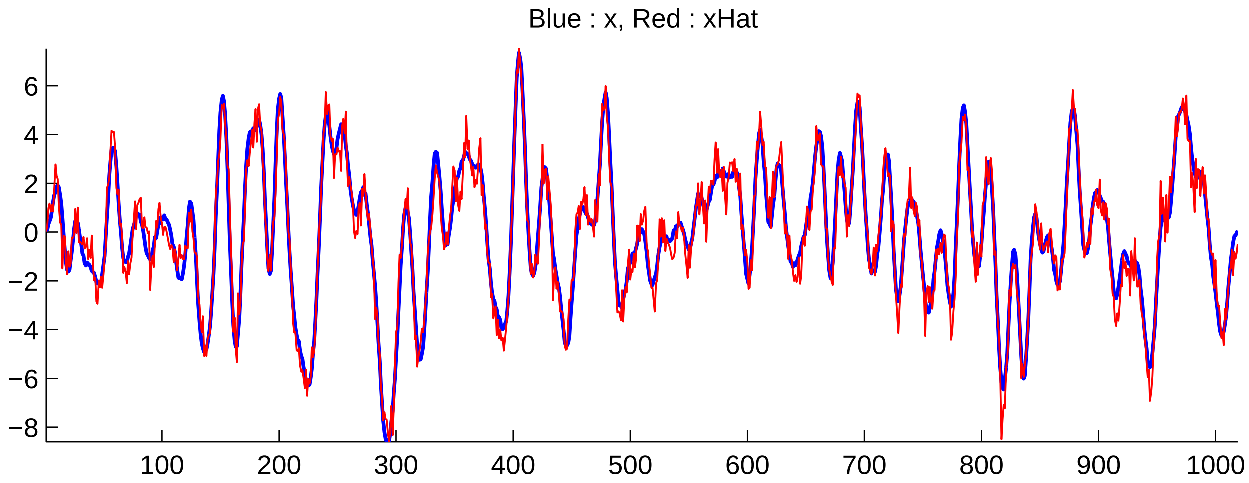
If the only knowledge is that n is white noise, the spectra of x and n may be estimated from the spectrum of y.

$$W(e^{j\omega}) = \frac{S_x(e^{j\omega})}{S_x(e^{j\omega}) + S_n(e^{j\omega})}$$

```
ax = xcorr(y) / length(y);
L = 25; % determines the size of the spectrum estimate
est = ax((end+1)/2 - L :(end+1)/2 + L);hh = xcorr(h);
Sy = fft(est,length(y)); Sy = abs(Sy); % this is the spectrum of y.
Sn = ones(size(Sy))*min(Sy); % estimate of the noise spectrum
filt = (Sy-Sn)./(Sy); % the Wiener filter
```

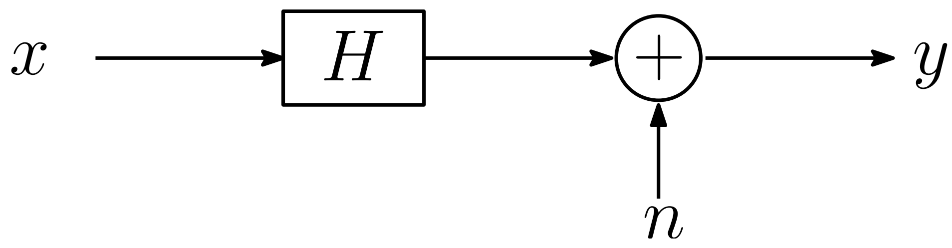
Wiener Filter – Unknown Spectra





Wiener Deconvolution

Deconvolution Scenario :

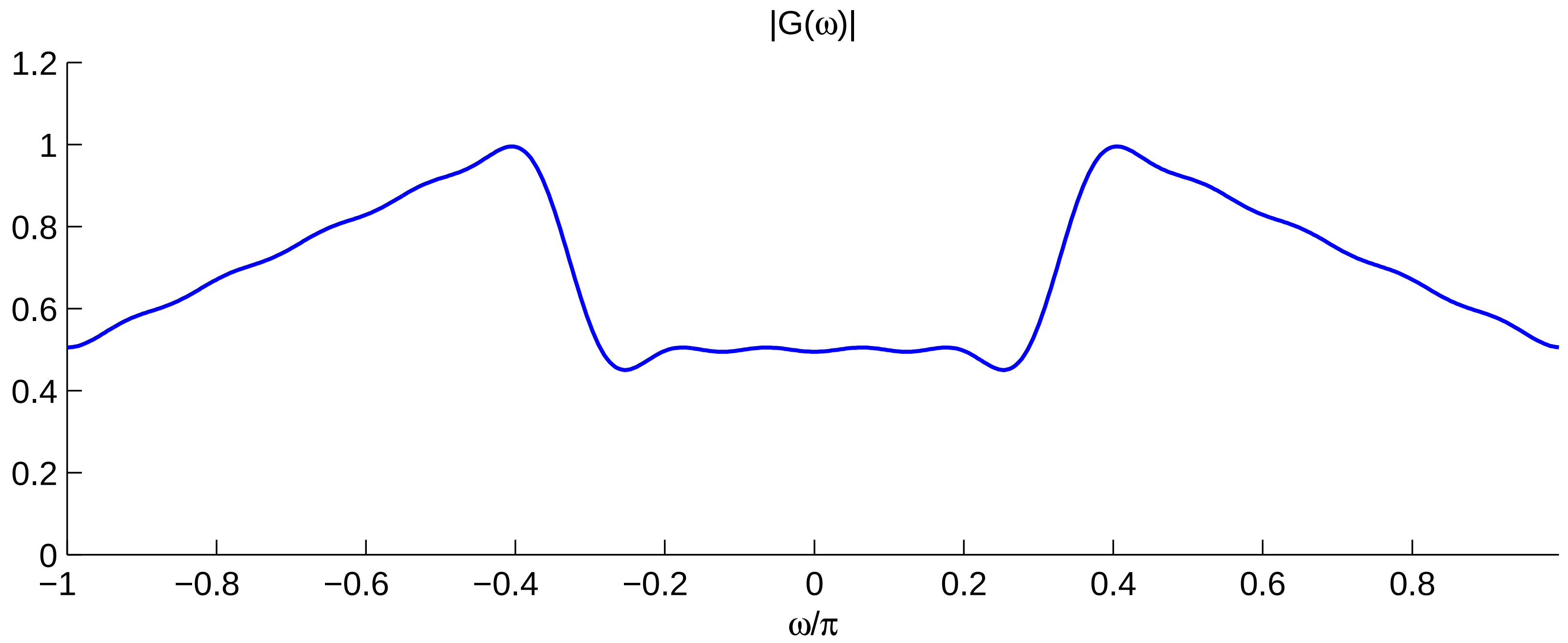


Wiener Filter :

$$W(e^{j\omega}) = \frac{G^*(e^{j\omega})\, S_x(e^{j\omega})}{|G(e^{j\omega})|^2\, S_x(e^{j\omega}) + S_n(e^{j\omega})}$$

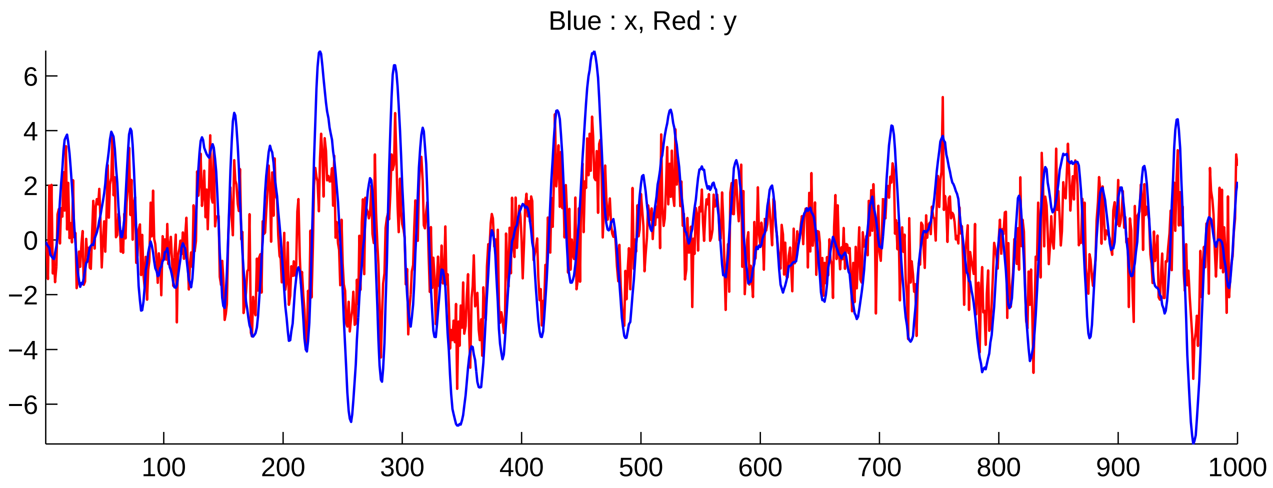
Wiener Deconvolution

```
% the blurring filter
N = 15;
g = firpm(2*(N-1),[0 .1 .2 .5]*2,[0.5 1 0.5 0.25]);
```



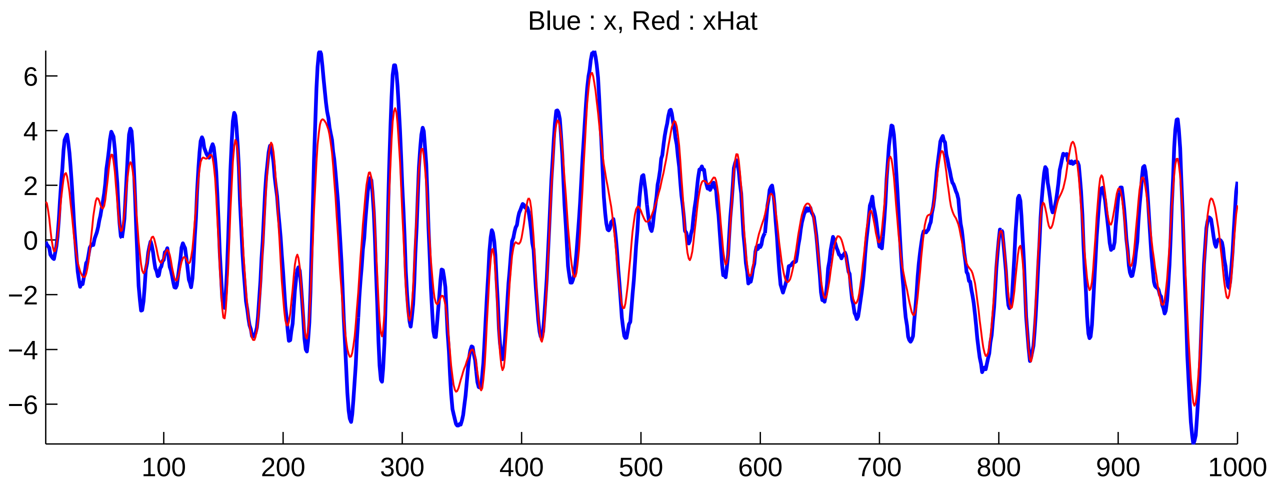
Wiener Deconvolution

```
M = 1000;
z = rand(1,M); z = sqrt(12)*(z - mean(z)); %this is uniformly distributed white noise with unit variance
h = hamming(20); % filter with a lowpass to produce 'colored' noise
x = conv(z,h); % colored noise (the clean signal)
x = x(1:M);

sig = 1;
n = sig*randn(size(x)); % noise

y = conv(x,g);
y = y(N:N+M-1) + n; % observation
```



Wiener Deconvolution

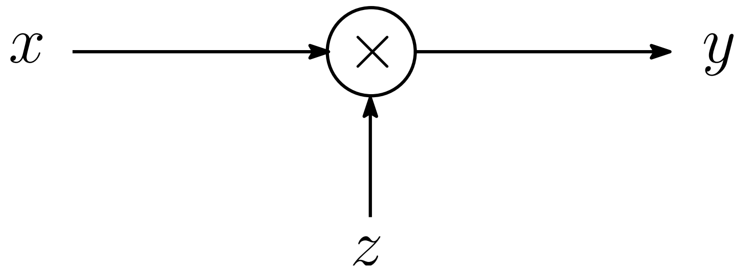
```
G = fft(ifftshift(g),M);G = G.';
hh = xcorr(h);
Sx = fft(hh,M); Sx = abs(Sx); % this is the spectrum of x.
Sn = (sig^2)*ones(M,1); % noise spectrum

filt = conj(G).*Sx./(Sx.*G.*conj(G) + Sn); % the Wiener filter

Y = fft(y); Y = Y.';
XHat = Y .* filt ; % FT of the denoised signal

xhat = ifft(XHat); % this is the estimate
```



Homomorphic Filtering

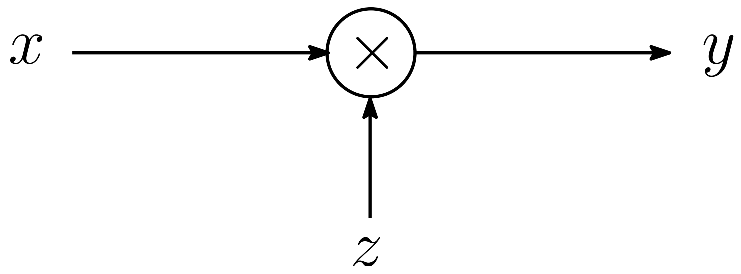
We have so far considered additive distortion.

What if the distortion is multiplicative?



Even if z is a lowpass function, the effect of z cannot be undone by an LTI system.

Homomorphic Filtering



Homomorphic filtering idea :

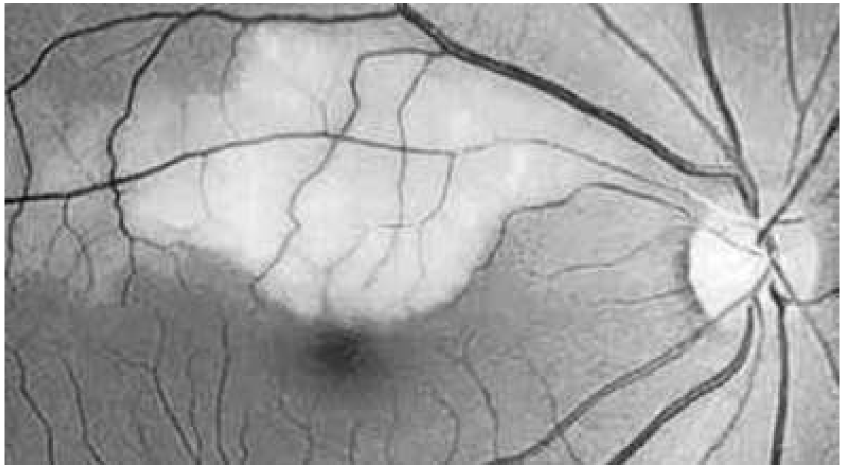
- If z is lowpass, so is $\ln z$.
- Let $d = \ln y = \ln z + \ln x$
- Highpass filter d to obtain $c = d * h$
- $\hat{x} = \exp c$

Homomorphic Filtering

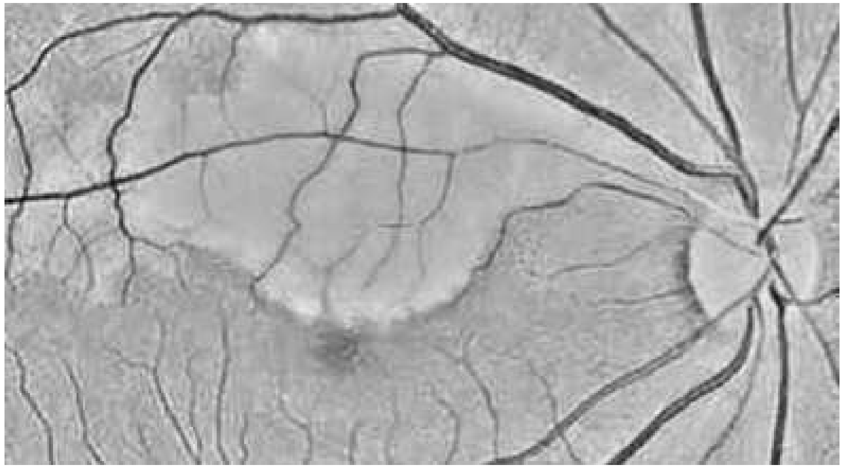
```
y = imread('retinalgri2.png');
b = log( double (y) ); % logarithm of the input image
K = 25;h = hamming(2*K+1); h = h/sum(h); h = h*h'; % lowpass filter
c = conv2(b,h); c = c(K+1:end-K,K+1:end-K); % crop the borders
d = b - c; % this is an approximation to ln x

xHat = exp(d); % this is the 'x' component (i.e. hatx)

zHat = exp(c); % this is the 'z' component
```

Homomorphic Filtering

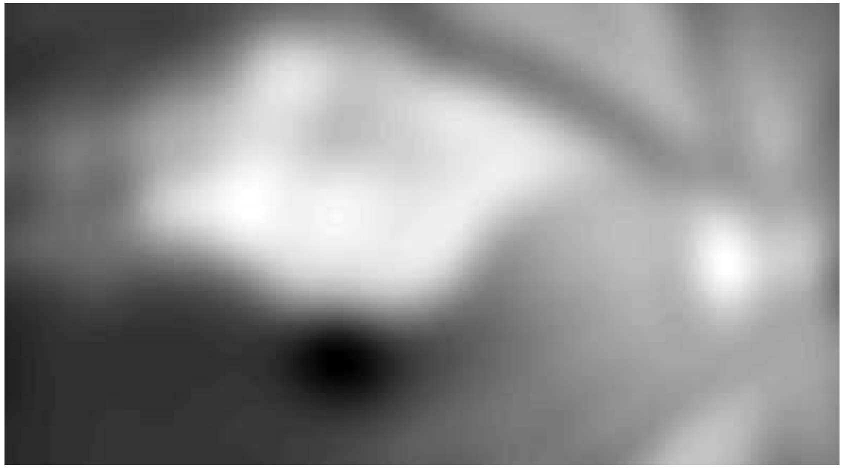
Input Image : $x \times z$



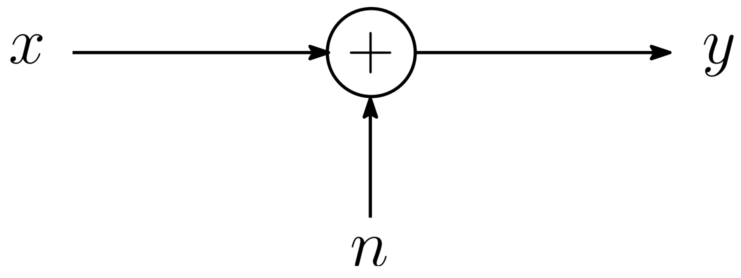
$\hat{x}$



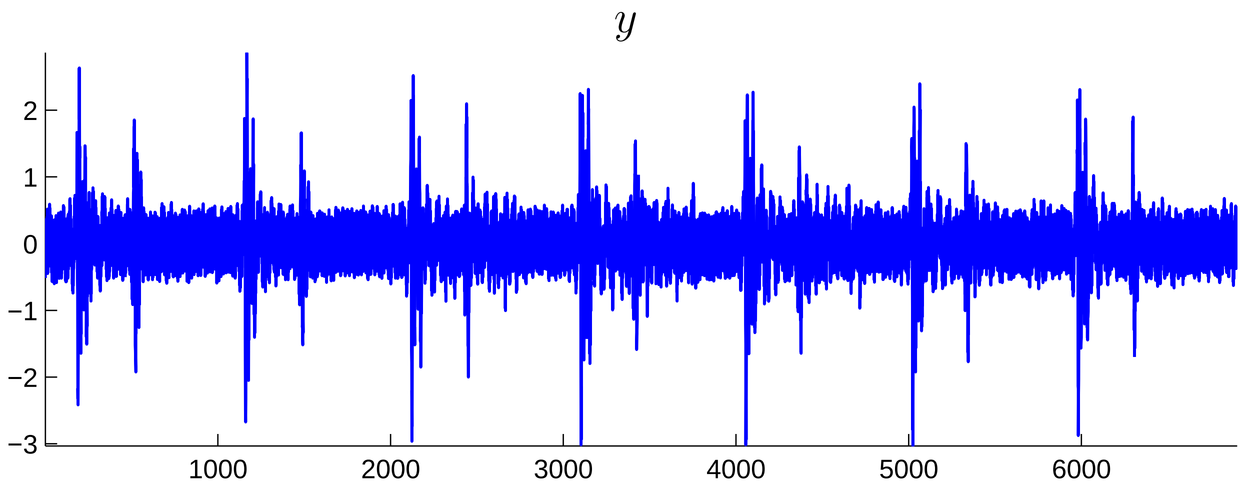
$\hat{z}$



Notch Filter



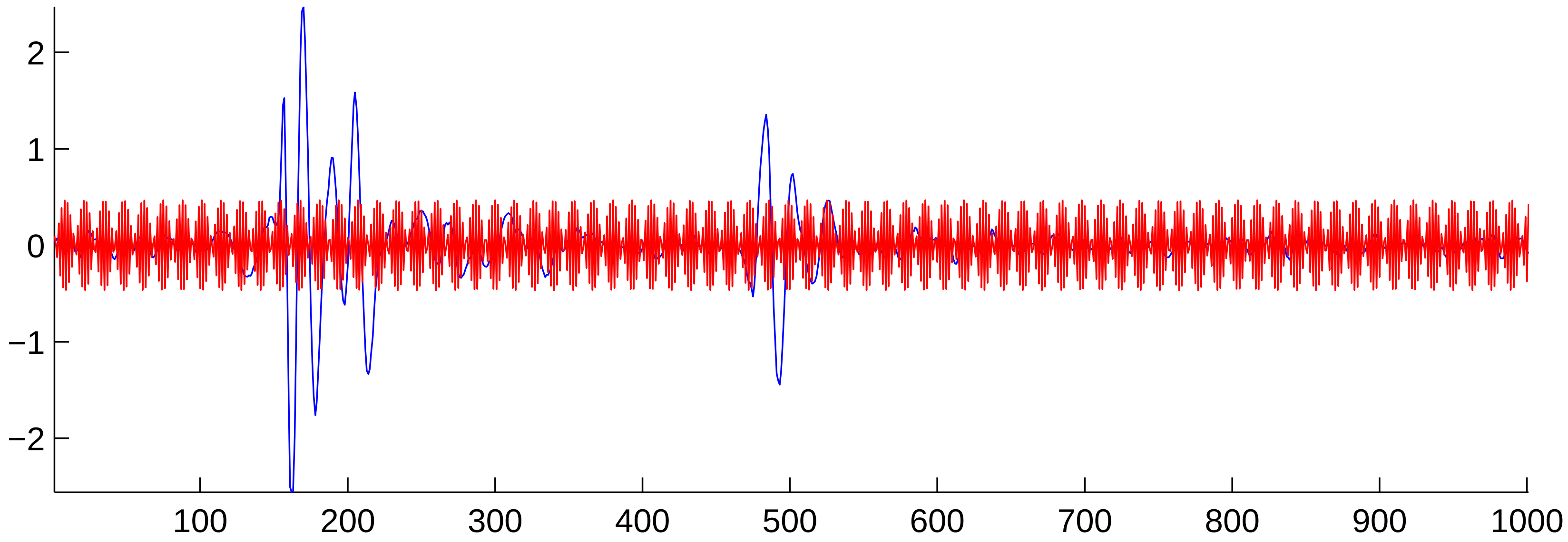
Assume that noise is at a certain frequency only (typically 50 Hz is considered).



Notch Filter

```
% x is the input signal

a = 0.5 + 0.5*rand(1,1);
b = 0.5*rand(1,1);
t = 1:length(x);
n = b * cos( a * pi * t ); % noise component with unknown frequency and amplitude

y = x + n; % observation signal
```

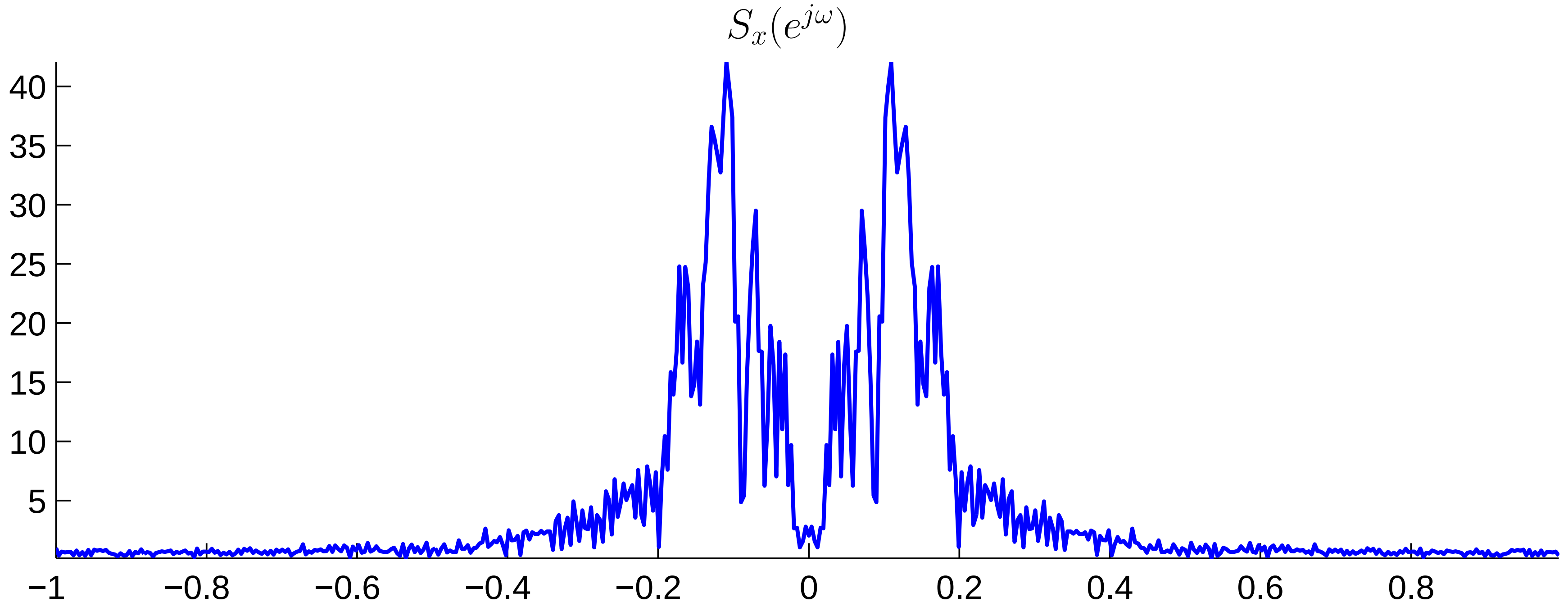
Blue : x, Red : $x - y$



Notch Filter

How do we determine the frequency of the unwanted component?

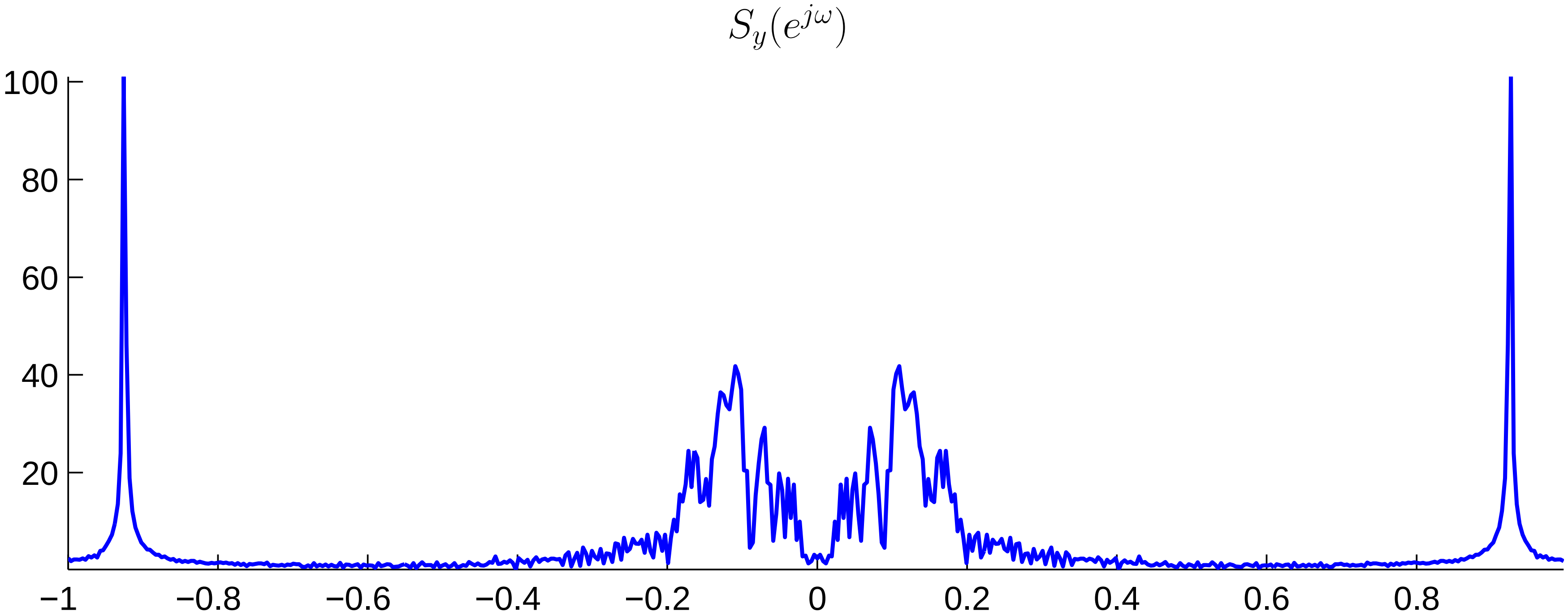
Idea : Take a look at the Spectrum



Notch Filter

How do we determine the frequency of the unwanted component?

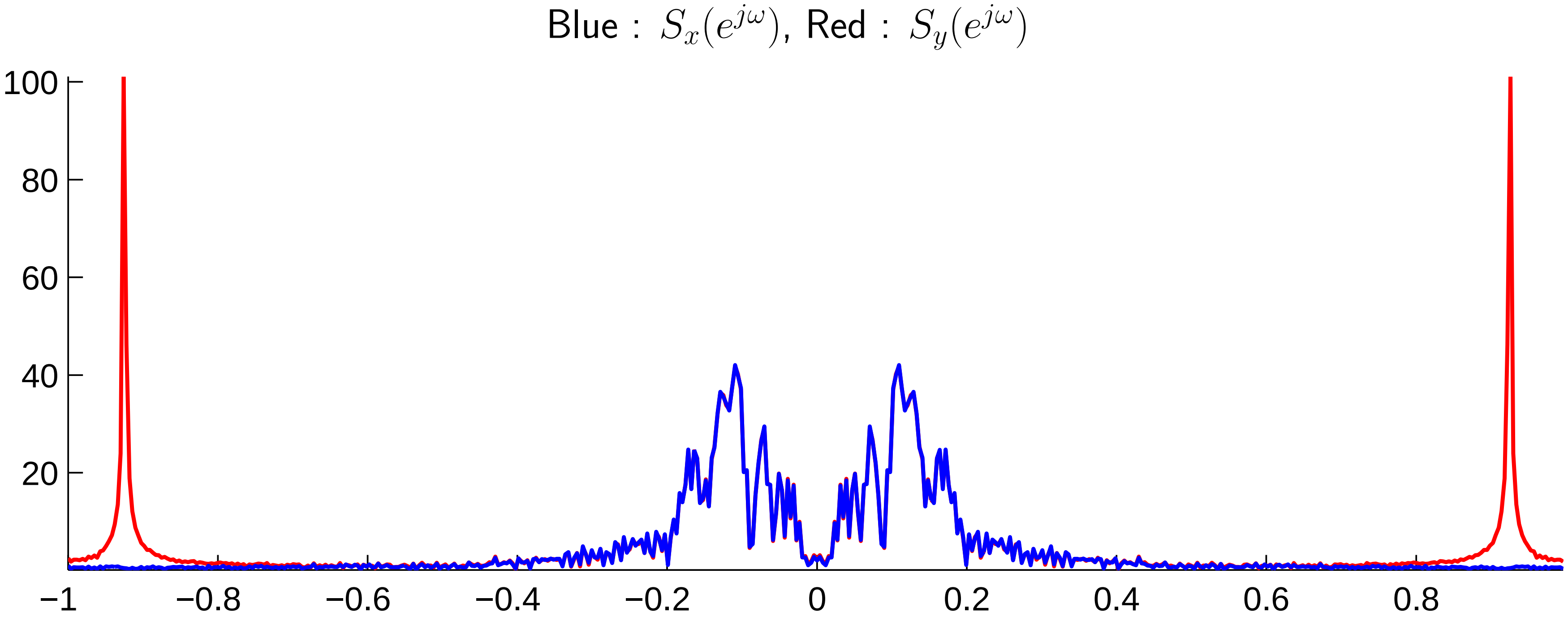
Idea : Take a look at the Spectrum



Notch Filter

How do we determine the frequency of the unwanted component?

Idea : Take a look at the Spectrum

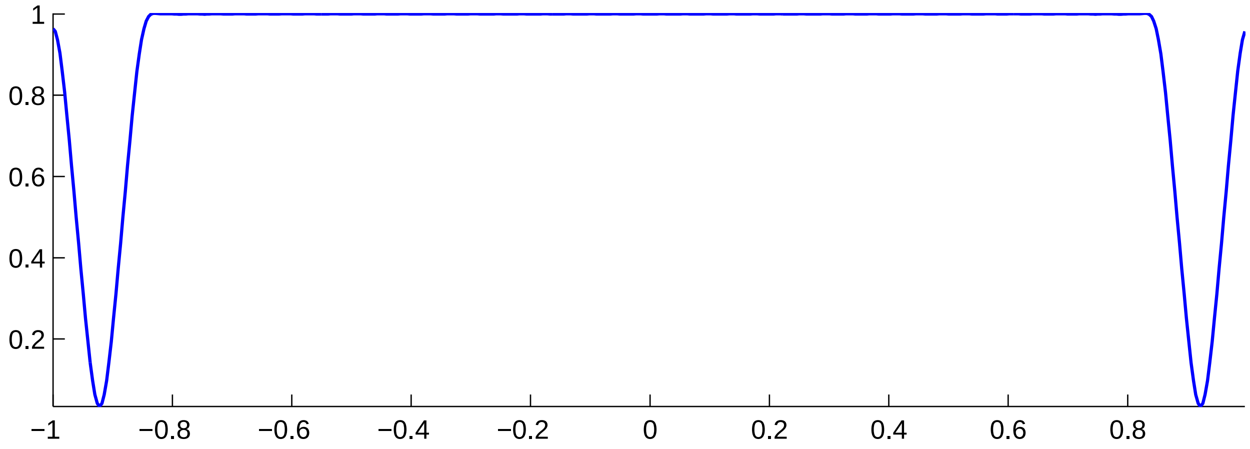


Notch Filter

Design a filter to suppress the frequency around which the unwanted component is concentrated.

```
% filter design by windowing
T = 128; % 'ideal filter' length
t = ones(1,T);
n1 = round(a*T/2);
M = 2; t(n1-M:n1+M)=0; t(T-n1-M:T-n1+M)=0; %suppres frequencies around 'a' radians

%windowing
g = ifft(t);
N = 50; win = hamming(2*N+1); win = win';
g2 = [g(end-N+1:end) g(1:N+1)] .* win;
```

The filter frequency response



Notch Filter – Results