

Lecture 6

Table of Contents

Power Spectral Density (PSD).....	1
Definition of PSD.....	2
Quiz: Compare the PSD of two sinusoids.....	2
Computing PSD of Example Signals.....	3
White Noise.....	3
Random Sinusoid.....	4
Coloured Noise.....	4
Average output of a filter.....	4
Example: Power Spectrum Analyser.....	5
Quiz: How do we know if a system in the real-world is linear and time-invariant?.....	6
Cross Power Spectrum Density.....	6
Coherence Function.....	7
Coherence Estimates in MATLAB.....	8
High-Pass IIR filter.....	8
High-pass filter with noise at the output.....	10
High-pass filter with noise at the input.....	12
Nonlinear filter.....	13
Nonlinear filter via clipping.....	15
Comb Filter.....	17
Properties of White Noise.....	21
Quiz: white noise cubed.....	22
Regular processes.....	27
Wold decomposition.....	27

Power Spectral Density (PSD)

Since we cannot have a mathematical model of random signals, so instead we have a statistical model.

The top half of figure 13.12 shows us how to compute the autocorrelation of the signal coming out, if we have the autocorrelation of the input signal.

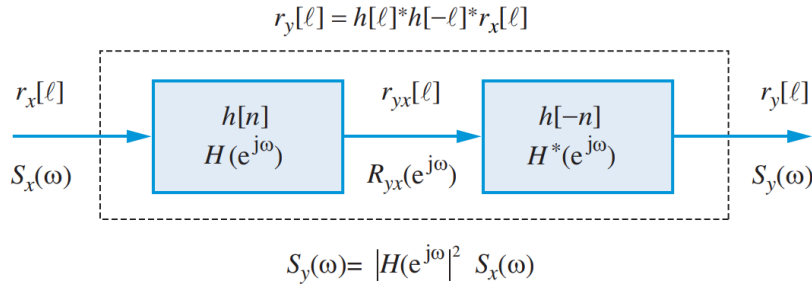


Figure 13.12 The ACRS and the PSD of the output process can be thought of as “filtered” by an LTI system with impulse response $r_{hh}[n] = h[n] * h[-n]$. Therefore, although LTI systems process individual sequences, they have the same effect on all sequences with the same mean and ACRS.

The bottom half of the figure has to do with the power spectral density.

Definition of PSD

Power spectral density is used to characterise stationary random processes in the frequency domain. Power Spectral Density is defined as follows:

$$S_{xx}(\omega) = R_{xx}(e^{j\omega}) = \sum_{\ell=-\infty}^{\infty} r_{xx}(\ell) e^{-j\ell\omega}$$

Essentially, Power Spectral Density is the discrete-time Fourier Transform of the auto-correlation sequence $r_{xx}(\ell)$.

Notice that phases disappear when we compute the spectrum.

SIDE NOTE: If we have a signal with some oscillation, the oscillation is observed in the autocorrelation. Taking the Fourier Transform of the same autocorrelation, we will observe that there is energy at that particular frequency.

We can go the opposite direction. Given the PSD, we can compute the auto-correlation sequence by the inverse Fourier Transform as follows:

$$r_{xx}(\ell) = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{xx}(\omega) e^{j\omega\ell} d\omega$$

Quiz: Compare the PSD of two sinusoids

Consider the following two signals

$$x_1(k) = A_1 \cos(\omega k + \phi) + A_2 \cos(2\omega k + \phi)$$

$$x_2(k) = A_1 \cos(\omega k + \phi) + A_2 \cos(2\omega k + \phi + \Delta\theta)$$

Where ϕ is uniformly distributed on $[0 : 2\pi]$ and $\Delta\theta$ is a fixed number.

Do the two signals have the same power density spectrum?

- A: Yes
- B: No
- C: Not enough information given

The answer is A; the two signals have the power density spectrum.

In this quiz, we need to compare the PSD of the two signals; $x_1(k)$ and $x_2(k)$

The only difference between the two signals is that $x_2(k)$ is a fixed phase shift of $\Delta\theta$

In order to compare compute the PSD, we need to compute the autocorrelation of the signal.

Last week (see problem ADSI Problem 4.4.3) we computed the autocorrelation of a generic real sinusoid $x(n) = A\cos(\omega n + \phi)$, which was:

$$r_{xx}(\ell) = \frac{A^2}{2} \cos(\omega \ell)$$

From this result, we see that the phase does not matter. Therefore, the autocorrelation of $x_1(k)$ and $x_2(k)$ are the same.

This means that the corresponding PSDs are also the same.

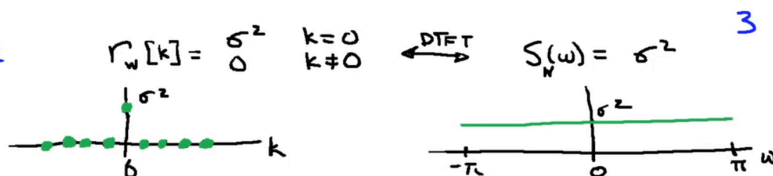
Lesson: when you compute the PSD, we only know the amount of energy at each given frequency. But we have no clue about the phase, which is essentially lost.

Computing PSD of Example Signals

White Noise

Examples

1) white noise



The power spectral density is constant which means that we have the same power across all frequencies. This signal is called white noise because the power is equally distributed across the entire spectrum.

Random Sinusoid

2) Random Sinusoid

$$s[n] = A \cos(\omega_0 n + \phi)$$

ϕ : uniform $[0, 2\pi]$

A : Gaussian, $E\{A^2\} = \sigma_A^2$

$$r_s[k] = \frac{\sigma_A^2}{2} \cos(\omega_0 k) \xleftrightarrow{\text{DTFT}} S_s(\omega) = \frac{\sigma_A^2}{4} \delta(\omega + \omega_0) + \frac{\sigma_A^2}{4} \delta(\omega - \omega_0)$$

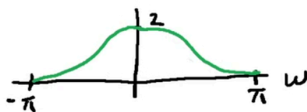


The power spectral density shows that the power is concentrated at $\pm\omega_0$. The area under these concentrations is $\frac{1}{4}\sigma_A^2$.

Coloured Noise

3) Colored Noise

$$r_c[k] = \begin{cases} 1 & k=0 \\ 1/2 & k=\pm 1 \\ 0 & \text{otherwise} \end{cases} \xleftrightarrow{\text{DTFT}} S_c(\omega) = 1 + \cos(\omega)$$



Average output of a filter

Suppose we want to know the average output of the filter i.e., $E[y^2(n)]$

The average output of a filter is actually just the value of the autocorrelation at lag 0. How? We can see this if look at the expression from a different perspective:

$$E[y^2(n)] = E[y(n)y(n-0)] = r_y(0)$$

Using the inverse definition of the PSD, we can compute $r_y(0)$

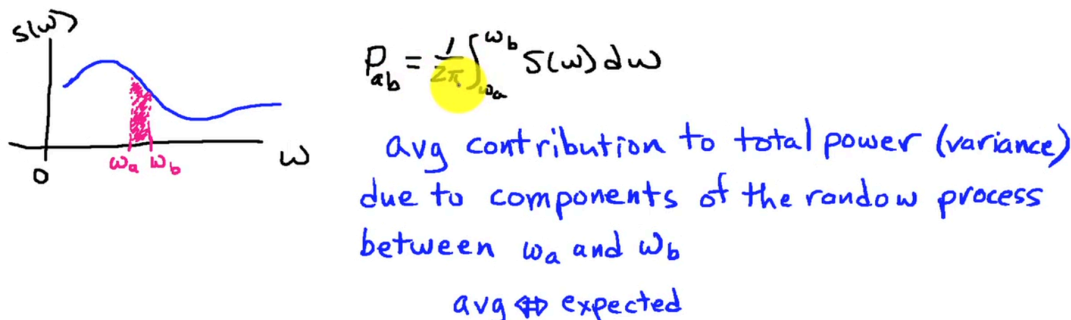
$$r_y(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{xx}(\omega) e^{j\omega \cdot 0} d\omega$$

$$r_y(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{xx}(\omega) d\omega$$

If we integrate the PSD from $-\pi$ to π , the resulting quantity is equal to the power in the time-domain. This means that **energy in the time-domain is equal to energy in the frequency domain**.

Another perspective:

Let P_{ab} be a normalised integration of the PSD from the interval ω_a to ω_b . The quantity P_{ab} is the expected (or average) contribution to the total power (or variance) that is due to the components of the random process between ω_a and ω_b . In other words, the area under the curve between ω_a and ω_b is the power that that portion of the spectrum is expected to contribute to the random process. It tells us how power is distributed in a frequency spectrum.



Example: Power Spectrum Analyser

Suppose we want to know the power spectrum of some input signal $x[n]$. We can pass that signal to an ideal bandpass filter with very narrow band and compute the power at the output. We define the bandpass filter to have the unit response in the vicinity of ω_c and the band $\Delta\omega$ is very small. Our bandpass filter rejects frequencies outside $\Delta\omega$ and pass any frequency inside this band.

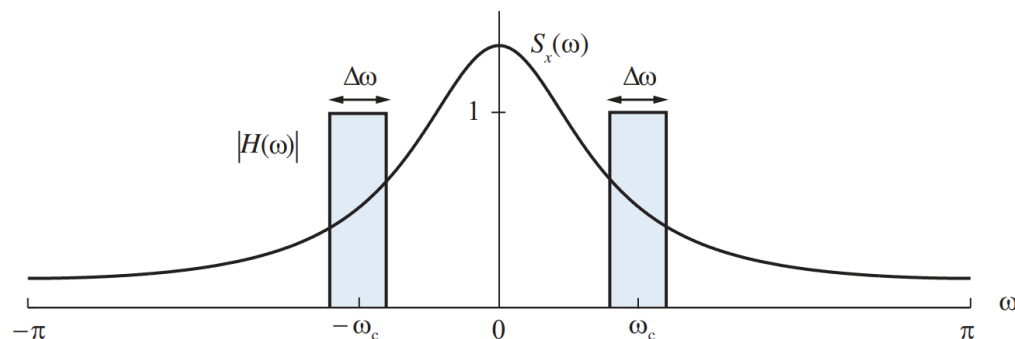


Figure 13.13 Physical interpretation of power spectrum density as power at the output of a narrowband LTI system.

Now, the question is how to do we compute the power spectrum of our ideal narrow bandpass filter when the signal is random?

The answer is to compute the the average output of the filter.

$$E[y^2(n)] = 2 \frac{1}{2\pi} \int_{\omega_c - \Delta\omega/2}^{\omega_c + \Delta\omega/2} S_{xx}(\omega) d\omega \approx \frac{1}{\pi} S_{xx}(\omega_c) \Delta\omega$$

Since the spectrum is symmetric around zero, we get power contribution from both sides of zero. This is called a double-sided spectrum. Therefore, we have $2 \frac{1}{2\pi}$ in the equation above.

Quiz: How do we know if a system in the real-world is linear and time-invariant?

In this class, we always make the assumptions that our systems are LTI system, but how do we actually know if a real-world system is LTI.

How do we measure if a system is linear and time-invariant?

We can measure if a system is linear by checking the two linearity properties:

1) $T(u + v) = T(u) + T(v)$

- Feed the system with the sum of two signals $x_1(n) + x_2(n)$ and observe the output $y(n)$.
- Feed the system with $x_1(n)$ to record the output $y_1(n)$.
- Feed the system with $x_2(n)$ to record the output $y_2(n)$.
- Now, the output $y(n)$ should be the same as $y_1(n) + y_2(n)$

2) $T(A v) = A T(v)$

- Feed the system with a signal with certain amplitude A e.g. $A = 42$. Record the output $y(n)$
- Feed the system with the same signal but where the amplitude is 1. Record the output $z(n)$
- The two outputs should have following relations: $y(n) = 42 \cdot z(n)$

We can measure if a system is time-invariant as follows:

- Feed the system with a known signal $x(n)$ and measure the output $y(n)$
- Feed the system with the same signal but shifted $x(n - k)$ the output should also be shifted $y(n - k)$

A more elegant approach is to compute the coherence function:

$$|\gamma_{yx}(\omega)|^2 = \frac{|S_{yx}(\omega)|^2}{S_{yy}(\omega)S_{xx}(\omega)} \quad \text{with } 0 \leq |\gamma_{yx}(\omega)|^2 \leq 1$$

For a linear system, the coherence function should be 1 or very close to it:

$$|\gamma_{yx}(\omega)|^2 = 1$$

Before talking about the coherence function, we need to describe the cross power spectrum density.

Cross Power Spectrum Density

Cross power spectrum density $S_{yx}(\omega)$ is the Fourier Transform of the cross-correlation:

$$S_{yx}(\omega) = \sum_{\ell=-\infty}^{\infty} r_{yx}(\ell) e^{-j\omega \ell}$$

The cross power spectrum compares two signals and computes how much the two signals oscillate in phase. If we get a large value at a given frequency ω_0 , then this means the two signals are oscillating together at ω_0 . If $S_{yx}(\omega_0)$ is small then the two signals are not in phase at ω_0 .

Why do we care? The cross power spectrum density allows us to define a coherence function.

Coherence Function

The (magnitude squared) coherence function allows us to determine in an elegant way when a system is linear.

Formally, the (magnitude squared) coherence function is defined:

$$|\gamma_{yx}(\omega)|^2 = \frac{|S_{yx}(\omega)|^2}{S_{yy}(\omega)S_{xx}(\omega)}$$

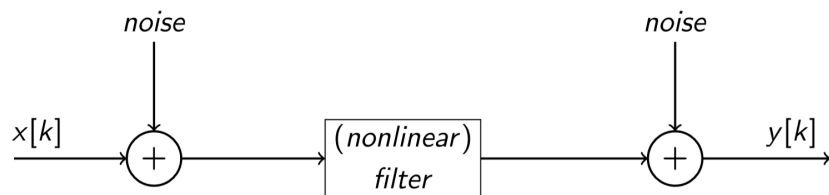
where $0 \leq |\gamma_{yx}(\omega)|^2 \leq 1$ i.e., the function yields a number between zero and one.

The coherence function of a perfect linear system is equal to 1.

However, in the real-world we don't always get a coherence of 1.

There are three primal factors that causes the coherence function to fall below 1.

Suppose we want to determine whether a system or a filter is linear:



- Factor 1: After we measure $x(n)$, some unaccounted noise creeps into the signal before going into the system.
- Factor 2: Some unaccounted noise creeps into the output signal before we measure it.
- Factor 3: The filter contains some nonlinear components.

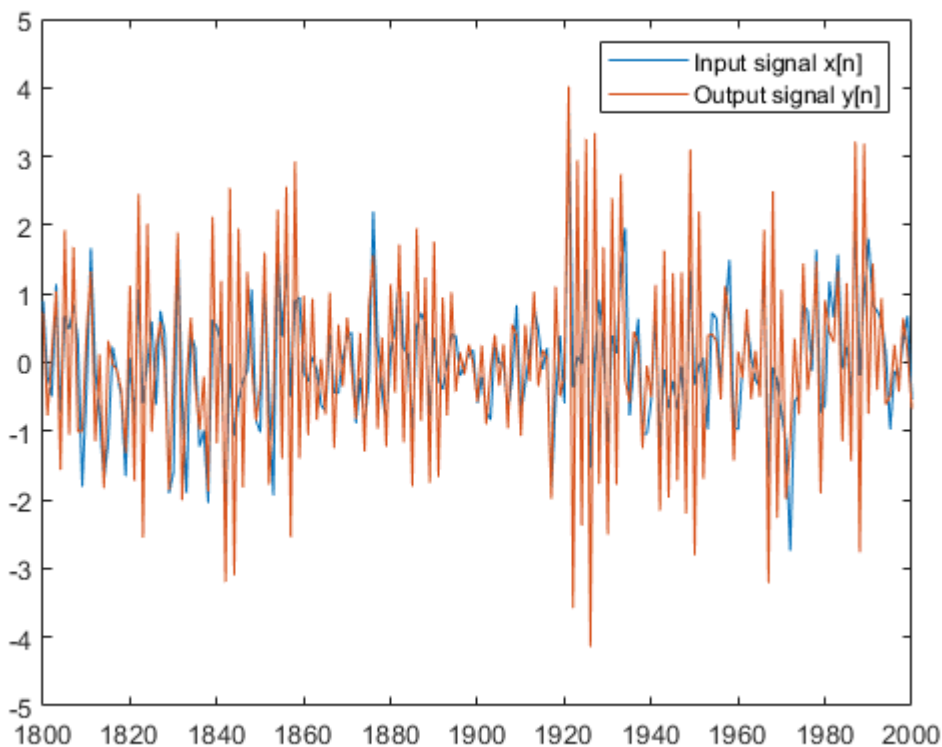
Important lesson: The coherence function is a tool that we can use to measure whether any given system is actually linear. Since the coherence function measures how close a system is linear, we don't know whether a low coherence is due to noise or a non-linear filter.

Coherence Estimates in MATLAB

Let us try it in MATLAB:

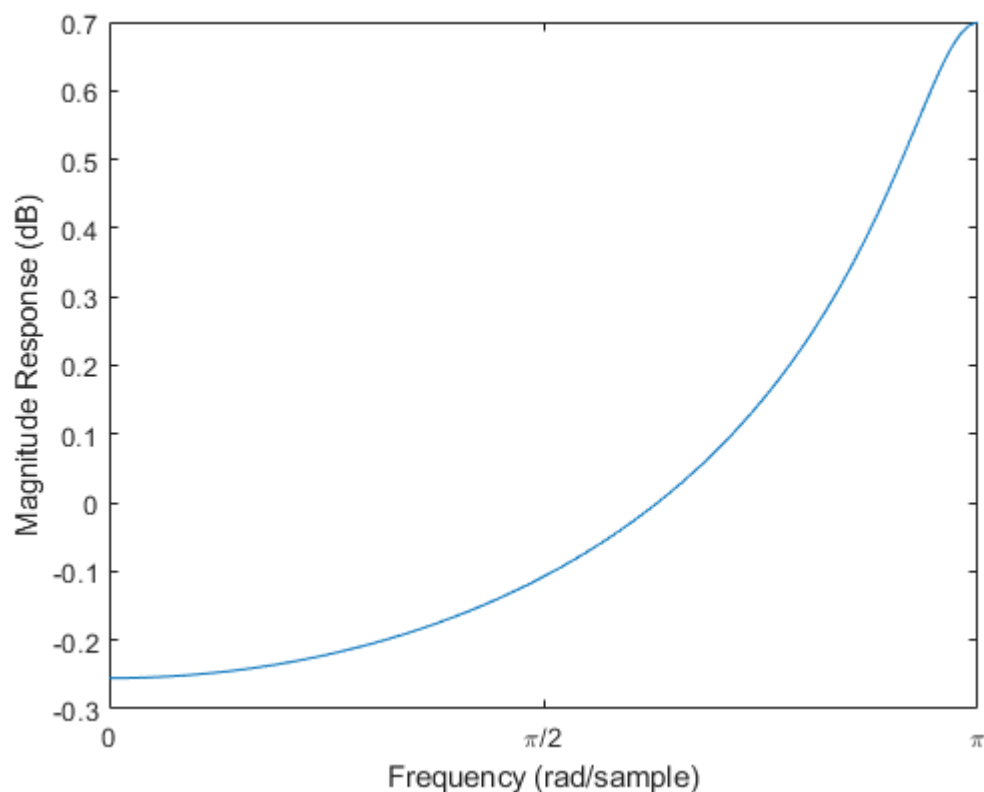
High-Pass IIR filter

```
N = 2000;  
n = 1800:N;  
  
% Create an input signal of random numbers  
x = randn(N, 1);  
  
% Define a high pass IIR filter that attenuates low  
% frequencies and amplifies high frequencies  
b = 1;  
a = [1, 0.8];  
  
y = filter(b, a, x);  
plot(n, x(n), n, y(n));  
legend('Input signal x[n]', 'Output signal y[n]')
```

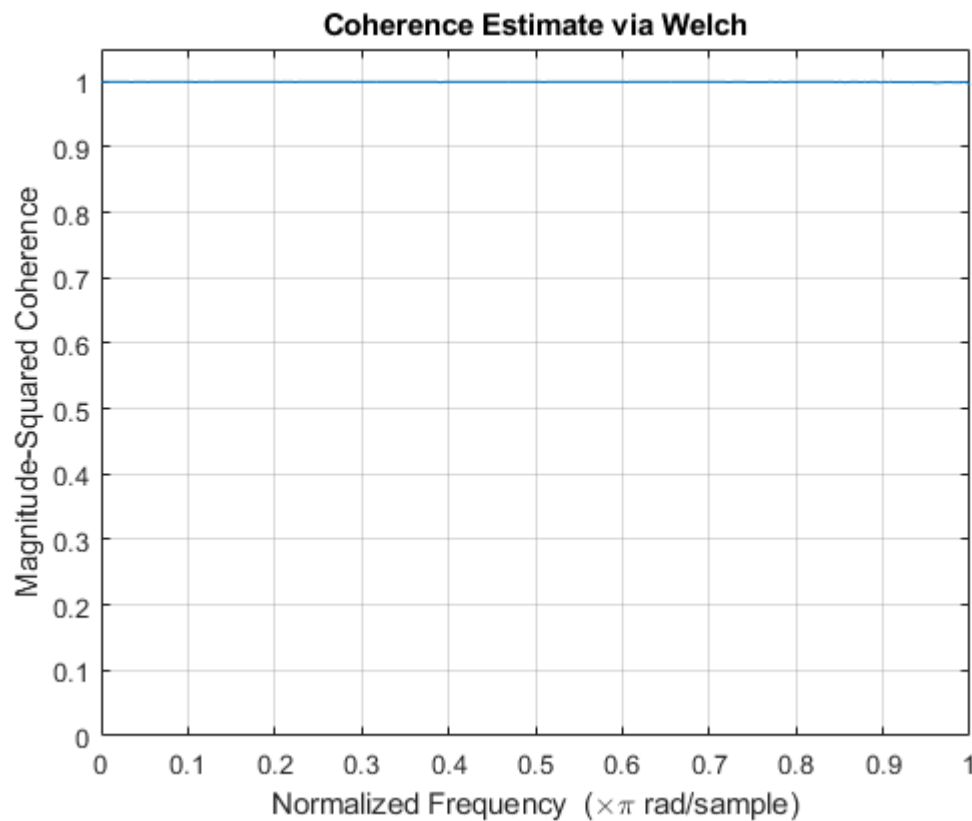


We can plot the magnitude response to see which frequencies the filter is attenuating and which frequency components it is amplifying:

```
[H, w] = freqz(b, a, 'whole');  
plot(w, log10(abs(H)));  
set(gca, 'XTick', 0:pi/2:2*pi)  
set(gca, 'XTickLabel', {'0', '\pi/2', '\pi', '3\pi/2', '2\pi'})  
xlabel('Frequency (rad/sample)')  
ylabel('Magnitude Response (dB)')  
xlim([0, pi]);
```



```
% Compute and plot the coherence estimate  
mscohere(y, x);  
ylim([0, 1.05]);
```

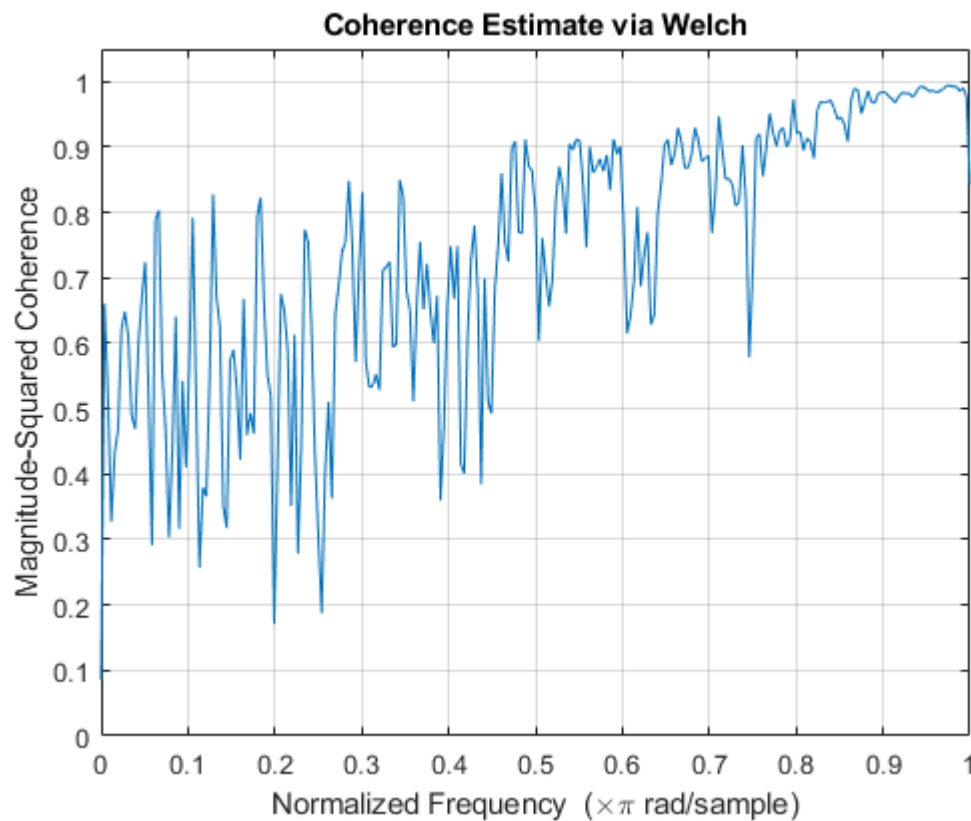


Notice that the estimate coherence function is 1. This is not a surprise because we are sending the input signal through a LTI system.

High-pass filter with noise at the output

```
% Define a high pass IIR filter that attenuates low
% frequencies and amplifies high frequencies
noise = 0.5;
y = filter(b, a, x);
y = y + noise*randn(N, 1);

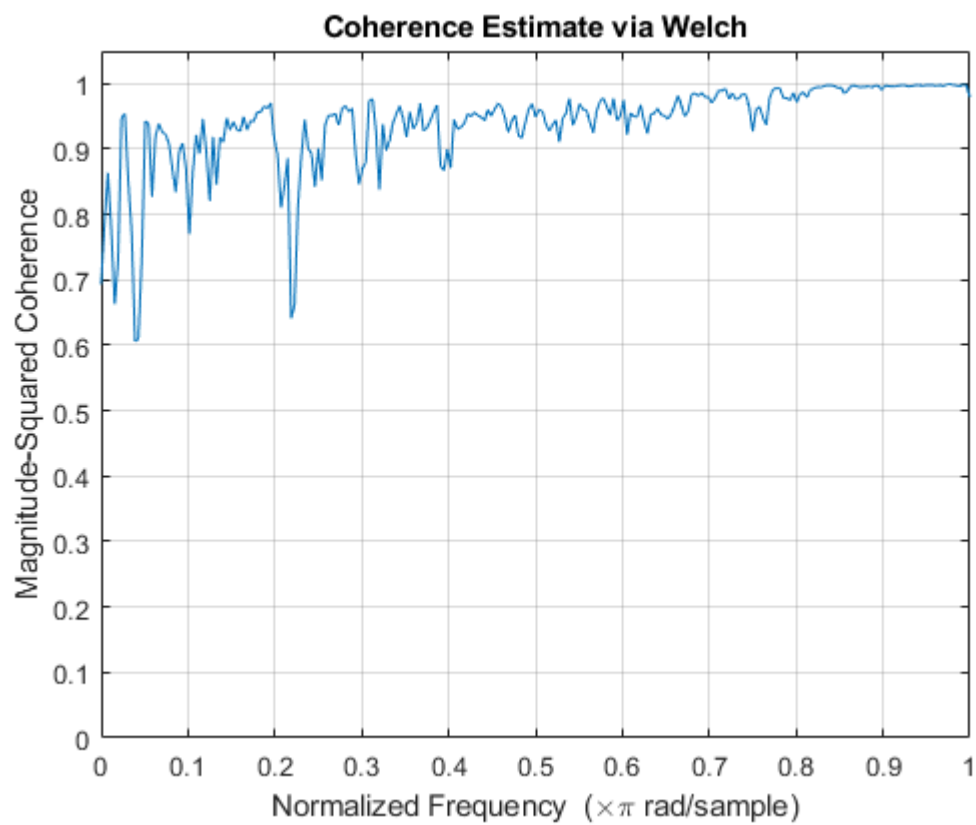
% Compute and plot the coherence estimate
mscohere(y, x);
ylim([0, 1.05]);
```



We observe that the coherence falls below 1 although we have a LTI filter because we have added a lot of noise to the output signal. Notice that there are low coherence at the low frequencies and high coherence at the high frequencies. This is because the high-pass filter attenuates low frequencies, and amplifies high frequencies.

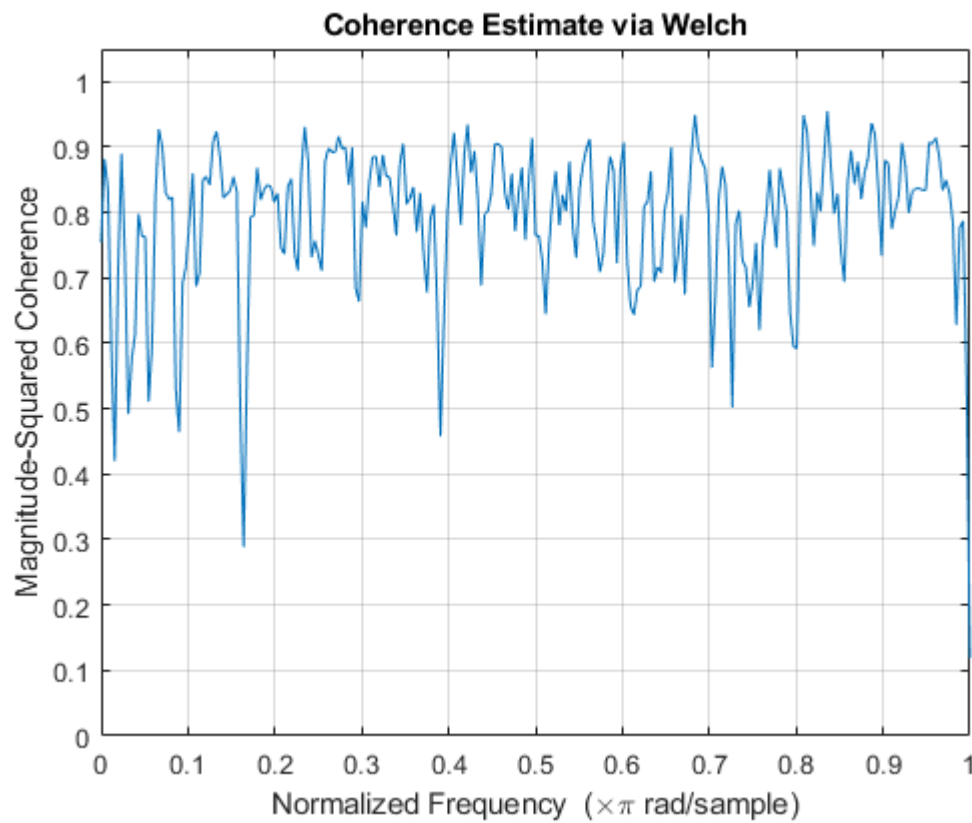
If the noise is reduced, we get close to 1:

```
noise = 0.2;  
y = filter(b, a, x);  
y = y + noise*randn(N, 1);  
  
% Compute and plot the coherence estimate  
mscohere(y, x);  
ylim([0, 1.05]);
```



High-pass filter with noise at the input

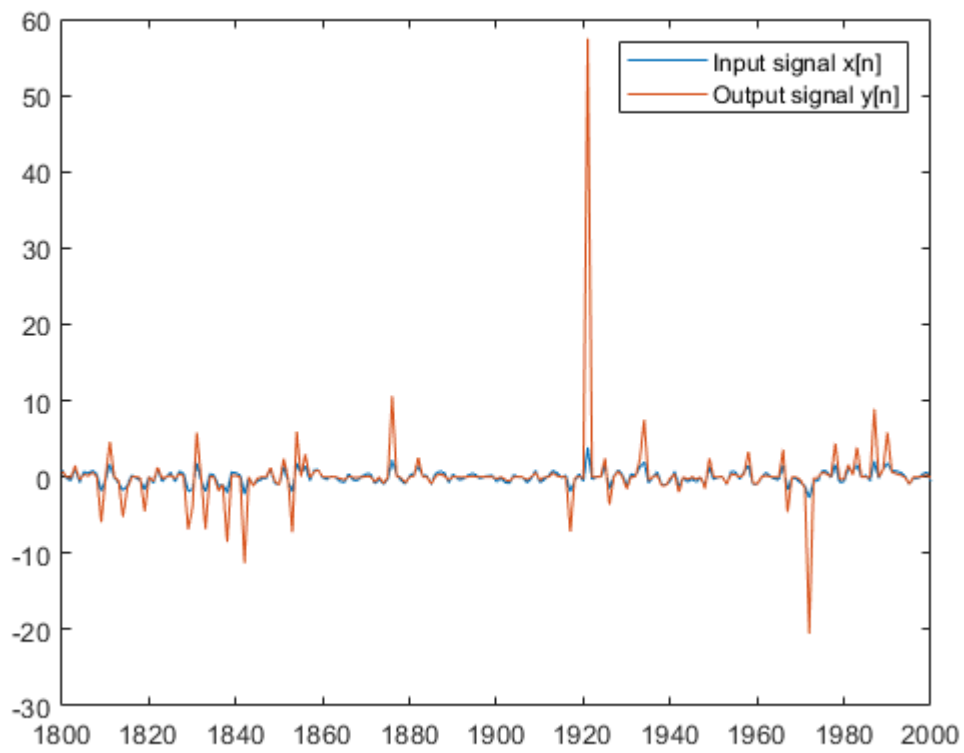
```
noise = 0.5;  
y = filter(b, a, x + noise*randn(N, 1));  
mscohere(y, x);  
ylim([0, 1.05]);
```



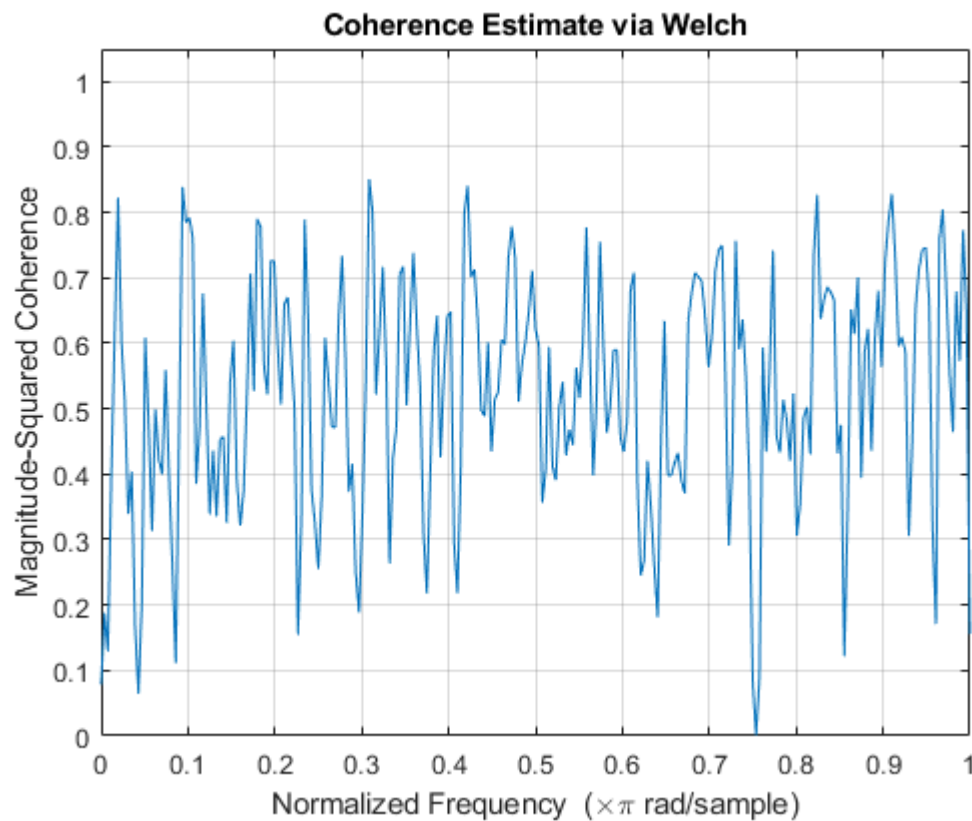
The additional noise is also being filtered so we have equal amount.

Nonlinear filter

```
% Take each sample and cube it
y = x.^3;
plot(n, x(n), n, y(n));
legend('Input signal x[n]', 'Output signal y[n]')
```



```
mscohere(y, x);  
ylim([0, 1.05]);
```



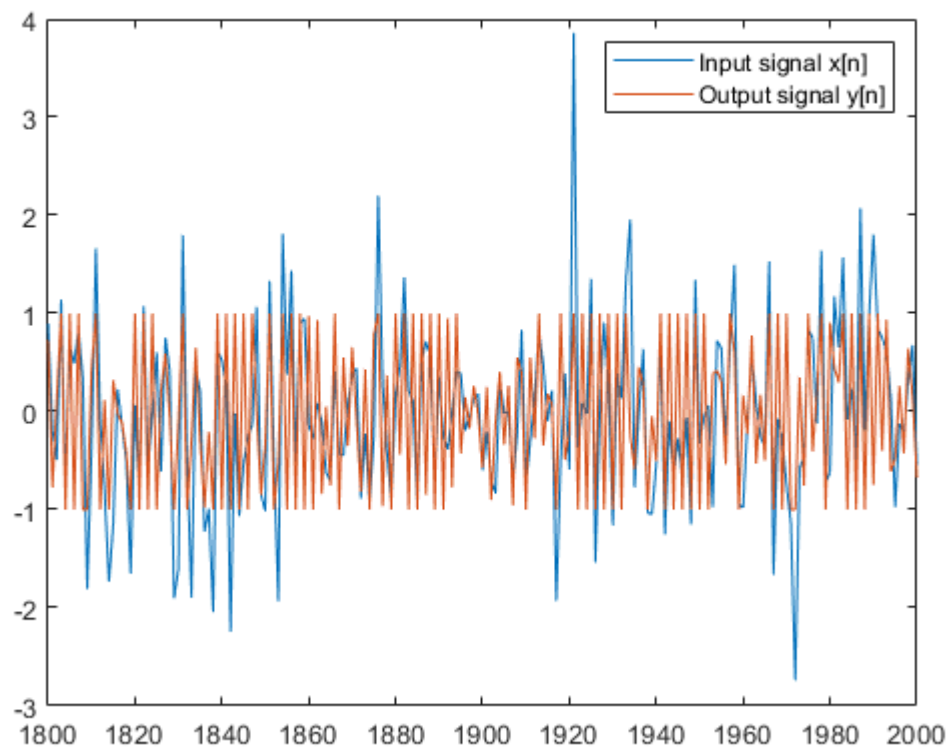
The coherence is all over the place.

Lesson: Since the coherence function measures how close a system is linear, we don't know whether a low coherence is due to noise or a non-linear filter.

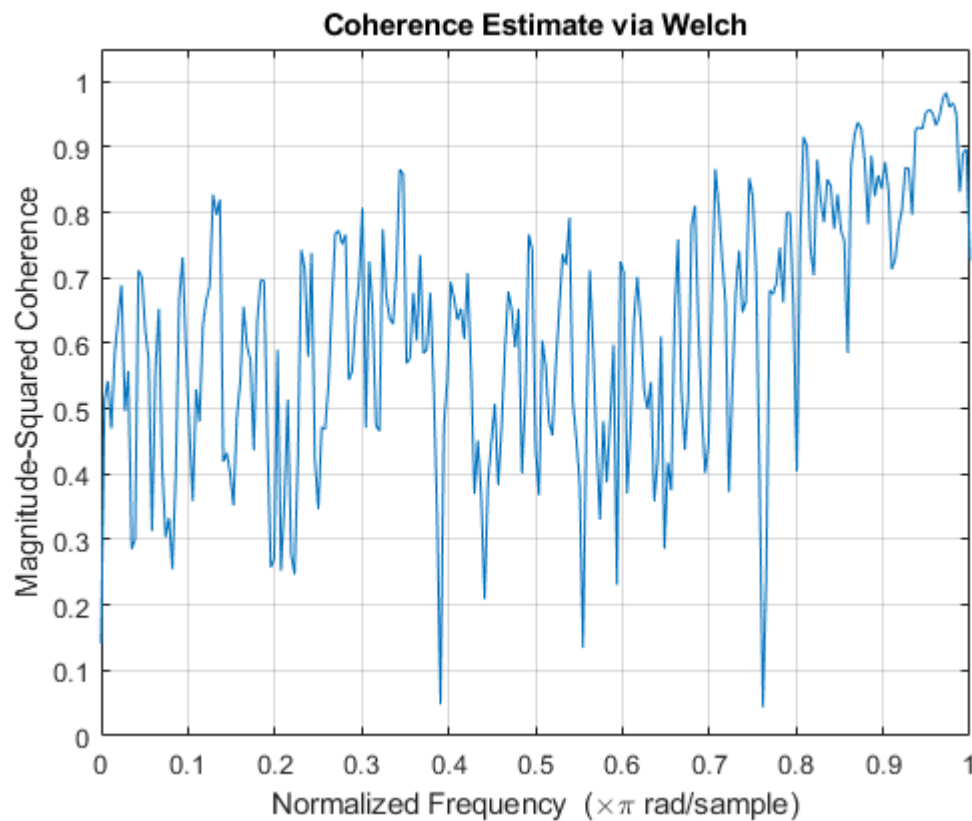
In practice, the coherence function is close to 1. The difference $1 - \text{mscohere}(y, x)$ tells us how well we can remove noise.

Nonlinear filter via clipping

```
% Non-linear filter via clipping
y = filter(b, a, x);
for k=1:N
    if y(k)>1
        y(k)=1;
    end
    if y(k)<-1
        y(k)=-1;
    end
end
plot(n, x(n), n, y(n));
legend('Input signal x[n]', 'Output signal y[n]')
```



```
mscohere(y, x);  
ylim([0, 1.05]);
```

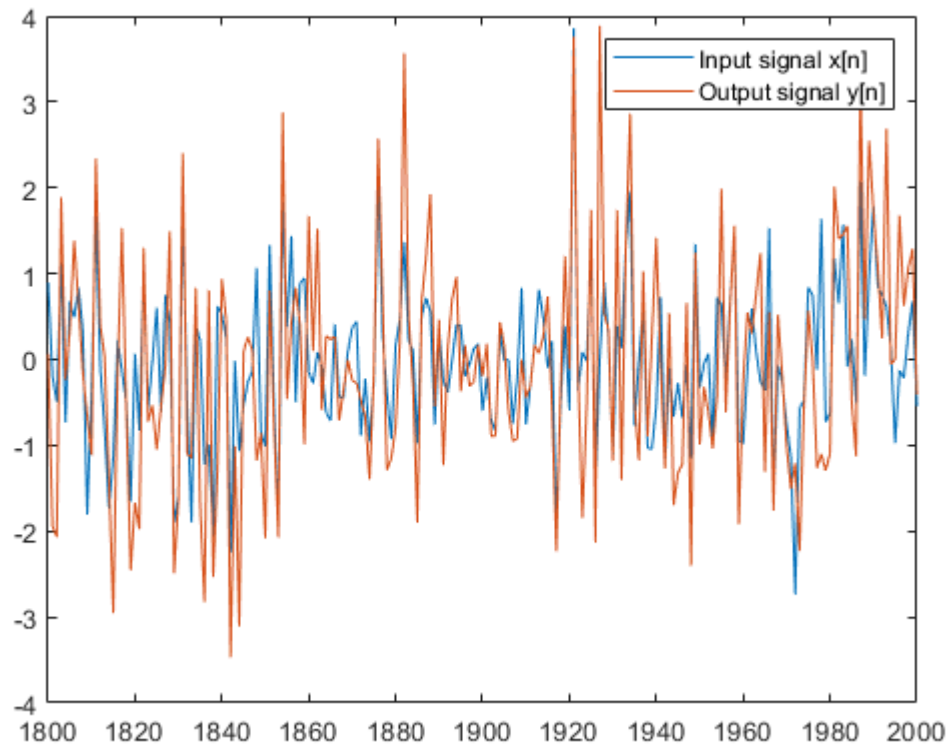
We see more coherence in the high frequencies than the low frequencies. Research why?

Comb Filter

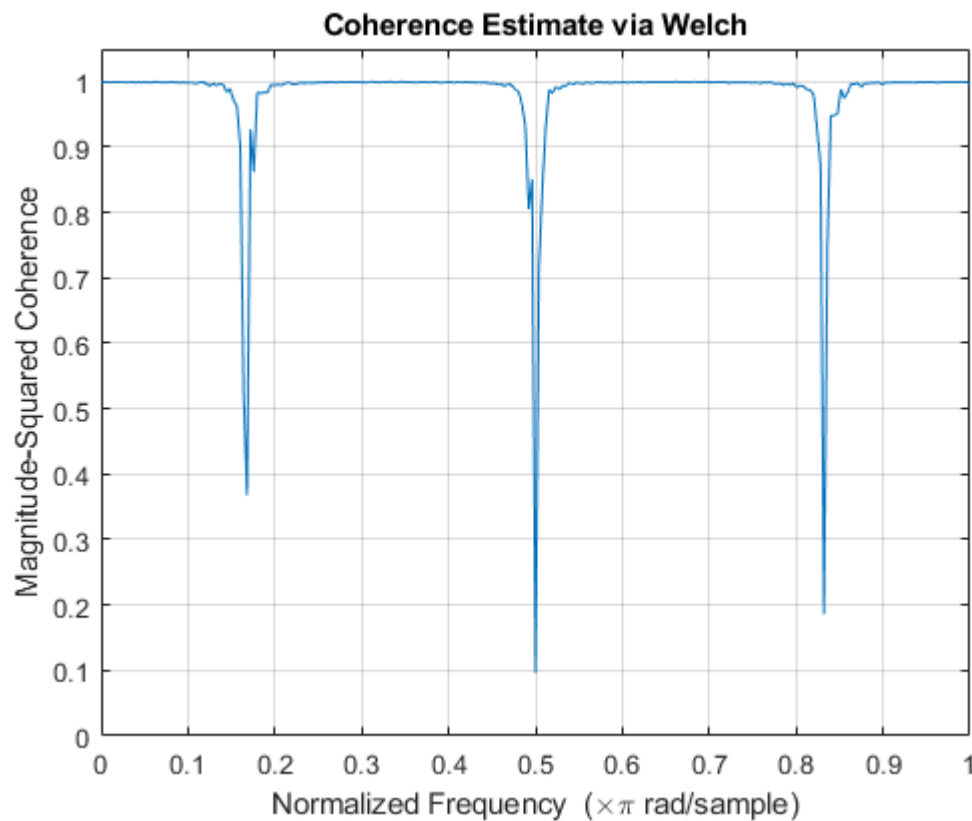
Let us defined a comb filter which has the difference equation:

$$y[n] = b_0x[n] + b_Mx[n - M]$$

```
% Define a comb filter
b = [1 0 0 0 0 0 1];
a = 1;
y = filter(b, a, x);
plot(n, x(n), n, y(n));
legend('Input signal x[n]', 'Output signal y[n]')
```



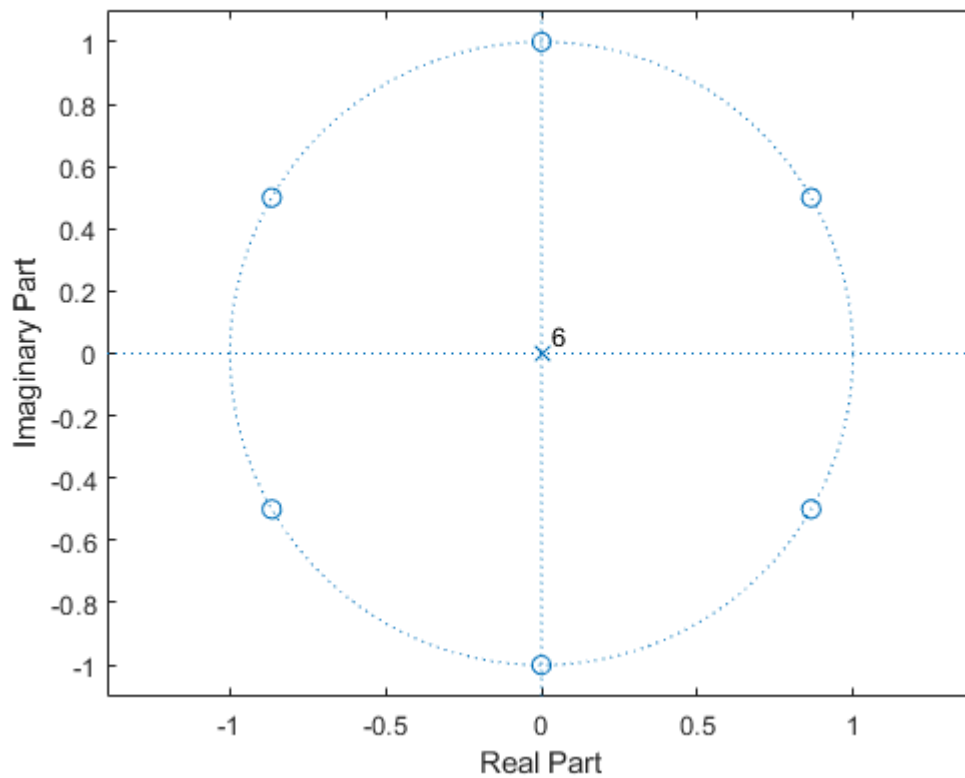
```
% Compute and plot the coherence estimate  
mscohere(y, x);  
ylim([0, 1.05]);
```



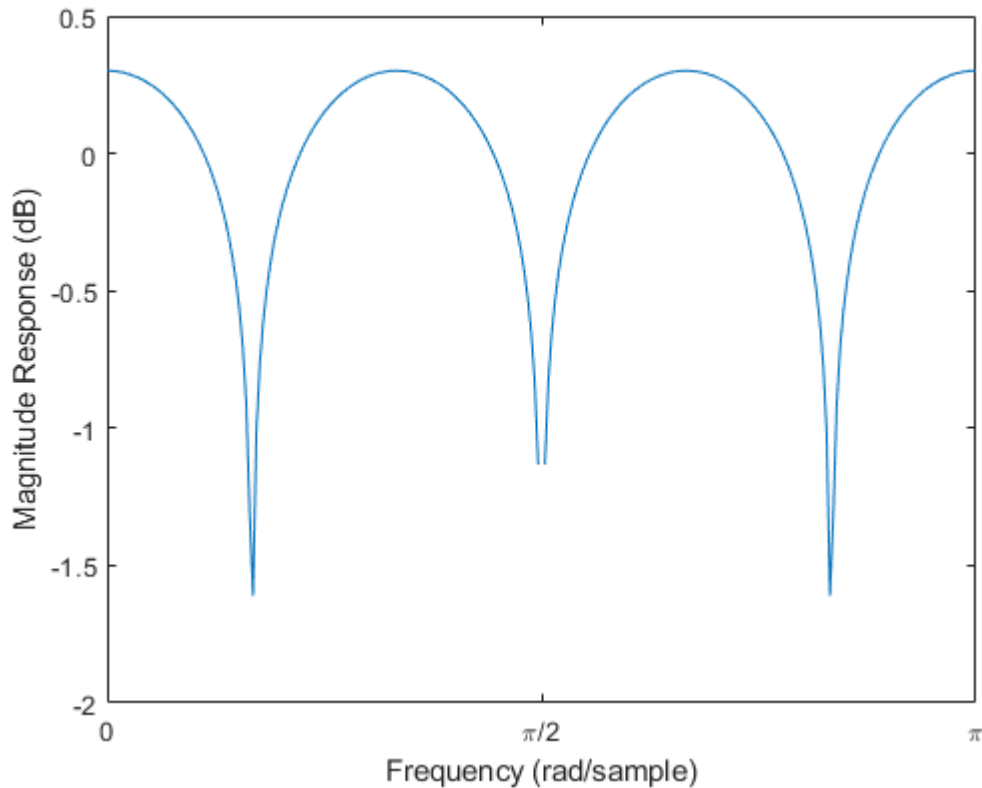
Notice that the coherence function dips at certain frequencies. This is because the comb filter removes certain frequency components of the input signal. Therefore, there is no coherence between the input signal and the output signal at those specific frequencies since the output signal does not contain these filtered frequencies.

The filter has 6 zeros on the unit circle, which means that it is removing $6/2=3$ frequency components:

```
zplane(b, a)
```



```
% Plot the magnitude response to see which frequency
% components are being attenuated
[H, w] = freqz(b, a, 'whole');
plot(w, log10(abs(H)));
set(gca, 'XTick', 0:pi/2:2*pi)
set(gca, 'XTickLabel', {'0', '\pi/2', '\pi', '3\pi/2', '2\pi'})
xlabel('Frequency (rad/sample)')
ylabel('Magnitude Response (dB)')
xlim([0, pi]);
```



Properties of White Noise

White noise is important for random signal modelling.

INSIGHT: we can use white noise to find the impulse response of a system because the autocorrelation of white noise is the same the delta signal.

Suppose we have a random process that produces perfect random noise.

Let $w(n)$ be a random signal from this process.

The **expected value** of the signal is zero because there are no patterns in white noise:

$$E[w(n)] = 0$$

The **autocorrelation of white noise** generates one peak at $\ell = 0$ because that is the only time when there is any correlation of the signal. One peak at $\ell = 0$ can be modelled by delta signal:

$$r_{ww}(\ell) = E[w(n)w(n - \ell)] = \sigma_w^2 \delta(\ell)$$

where σ_w^2 is the variance of the signal which can be computed as follow:

$$\sigma_w^2 = E[w^2(n)] - E[w(n)]^2$$

$$\sigma_w^2 = E[w^2(n)] - 0 \quad (\text{by definition } E[w(n)] = 0)$$

$$\sigma_w^2 = E[w^2(n)]$$

The **power spectral density of white noise** is:

$$S_w(\omega) = \sum_{\ell=-\infty}^{\infty} r_{ww}(\ell) e^{-j\omega\ell}$$

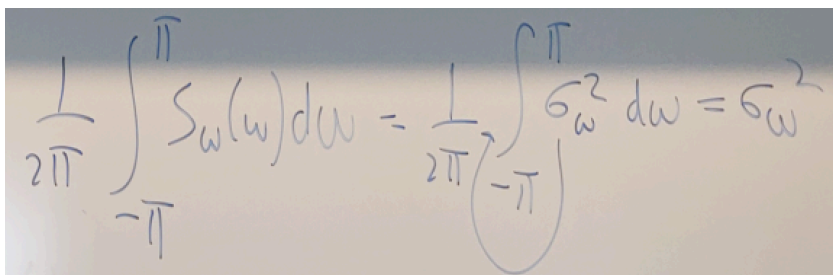
$$S_w(\omega) = \sum_{\ell=-\infty}^{\infty} \sigma_w^2 \delta(\ell) e^{-j\omega\ell}$$

All terms where $\ell \neq 0$ become zero so we left with one term when $\ell = 0$

$$S_w(\omega) = \sigma_w^2 \delta(0) e^{-j\omega \cdot 0}$$

$$S_w(\omega) = \sigma_w^2 \cdot 1 \cdot 1$$

$$S_w(\omega) = \sigma_w^2$$



$$\frac{1}{2\pi} \int_{-\pi}^{\pi} S_w(\omega) d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} \sigma_w^2 d\omega = \sigma_w^2$$

Quiz: white noise cubed

We are used to white Gaussian noise

$$x[n] \sim \text{WGN}(0, \sigma_x^2) \quad \text{with} \quad r_{xx}(l) = \sigma_x^2 \delta(l)$$

What happens if we play with white Gaussian noise

```
x=randn(10000,1);  
y=x.^3;  
[ryy,lags]=xcorr(y,20,'biased');  
stem(lags,ryy)
```

Is $y[n]$ also white noise?

A: Yes

B: No

So the question is this: suppose we have a white noise signal $x(n)$. If we take each element of this signal and cube it, would the resulting signal also be white noise?

The answer is yes! The output is also white noise because we are not introducing any correlation between different samples.

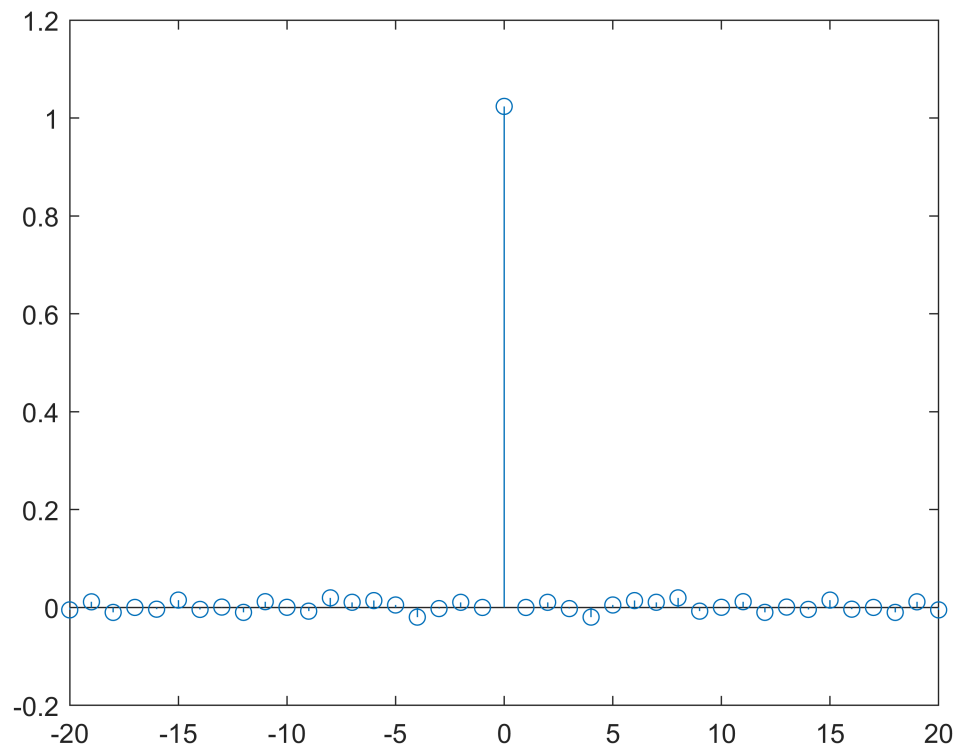
Let us prove it in MATLAB.

If we plot the autocorrelation of white noise, we should observe a signal similar to delta signal with amplitude of σ_x^2 because:

$$\text{if } x(n) \sim \text{WGN}(0, \sigma_x^2) \text{ then } r_{xx}(\ell) = \sigma_x^2 \delta(\ell)$$

First, we plot $r_{xx}(\ell)$

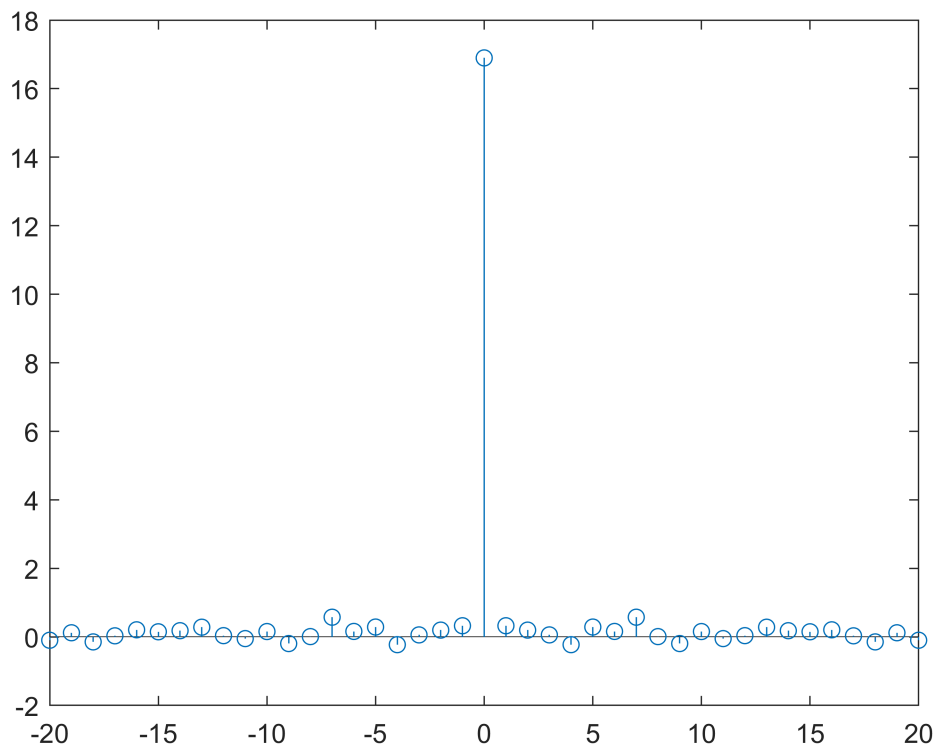
```
N = 10000;  
n = [1:N];  
x = wgn(N,1,0);  
  
[r_xx, lags] = xcorr(x, 20, 'biased');  
stem(lags, r_xx);
```



This looks like what have expected.

Now, let us plot the autocorrelation of $y(n) = x^3(n)$

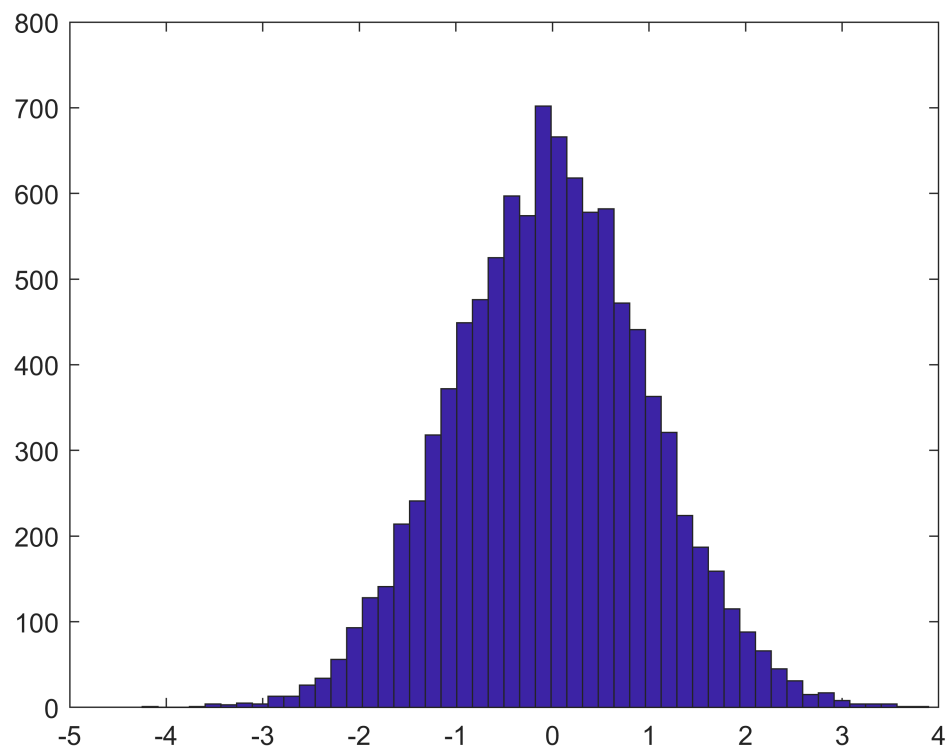
```
y = x.^3;  
[r_yy, lags] = xcorr(y, 20, 'biased');  
stem(lags, r_yy);
```

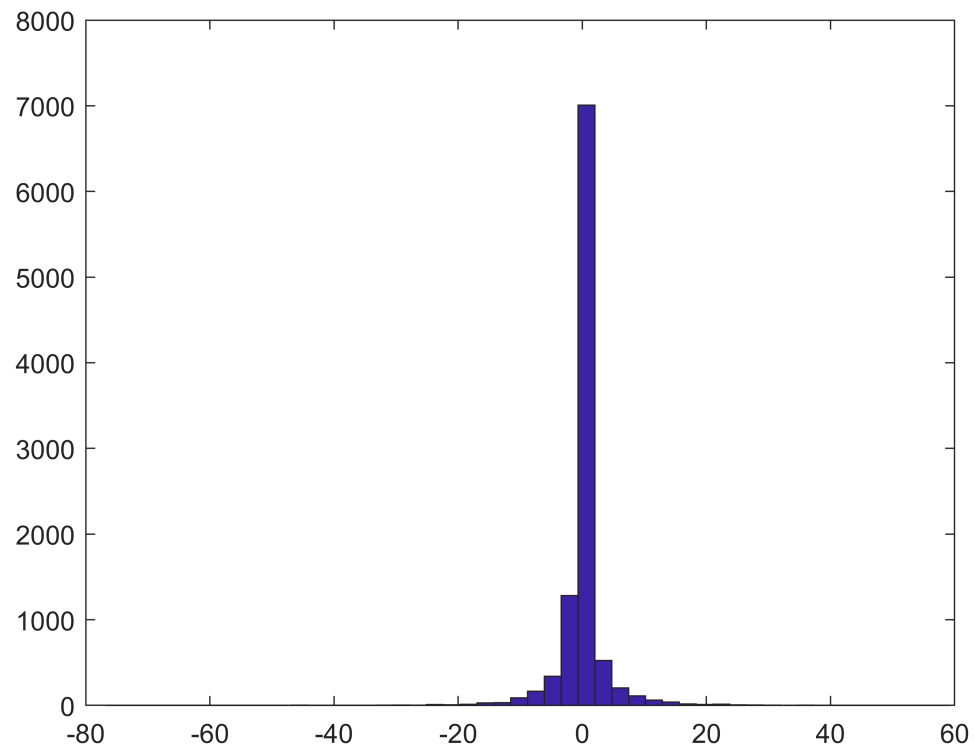
This proves that the answer to the quiz is Yes!

INSIGHT: Noise does not necessarily have to be Gaussian in order to be white. White noise can have any probability density function. Noise is said to be white when there is no correlation between samples.

```
hist(x, 50)
```



```
hist(y, 50)
```



Regular processes

A regular process is a filter

Coloring filter:

Whitening filter:

Analysis filter:

Wold decomposition

