

01-lecture

ch_allpass

ch_arma

ch_autocorr

ch_hilbert

ch_karhunen_loeve

ch_lattice

ch_linear_predictors

ch_matched_filters

ch_multirate

ch_noise

ch_pisarenko

ch_probability

ch_psd

ch_random_processes

ch_signal_modelling

ch_wiener_filters

matlab_cheatsheet

Lecture 1: Distortion of signals passing through LTI systems

In [1]:

```
import utils as utils
utils.load_custom_styles()
```

Stylesheet "styles.css" loaded.

Recap

Typically, we think LTI systems just as filters.

$$y[n] = \sum_{k=0}^{M-1} h[n]x[n - k]$$

where $h[n]$ is the impulse response and M is the number of weights.

The output of a filter is the linear combination of previous samples of the input signal $x[n]$ multiplied by the weights $h[n]$.

This is the time domain. Signals in the real world occurs in the time domain. However, it is more convenient and easier to analyse signals in the z -domain or the in the frequency domain.

If we take the z -transform of the impulse response, we get the system function or the transfer function:

$$H(z) = \sum_{n=-\infty}^{\infty} h[n]z^{-n}$$

In practice, we typically have a finite-length impulse response of length M . This means that most of the terms in the infinite sum will be zero and disappear.

We can go from the system function (which is in the z -domain) to the Fourier domain just by evaluating the system function for $z = e^{j\omega}$:

$$H(e^{j\omega}) = H(z) \Big|_{z=e^{j\omega}} = \sum_{n=-\infty}^{\infty} h[n]e^{-j\omega n}$$

where $H(e^{j\omega})$ is called the frequency response. This describes the impulse function in the frequency domain. The quantity $e^{j\omega}$ is a complex function of ω .



In other words, the frequency response is equal to a system function evaluated on the unit circle.

Distortionless systems

Basically, a distortionless system does not have any kind of distortion as we send signal through the system. Why are focusing on these system? Because there is a different way of analysing this kind of system than we have seen previously.

Let us describe a distortionless system. To have a distortion-free system, the "shape" of the output signal (shape in the time domain) to be identical to the shape the input signal:

$$y[n] = Gx[n - n_d]$$

where G is a positive number i.e. $G > 0$. This system is simply scaling the system by G and delaying the signal.

For example, if we put through a square-wave signal to this system, the output is also a square-waved signal. However, the output signal may be delayed by n_d and the amplitude may be larger or smaller depending on G .

Frequency Analysis

In order to better understand distortionless system, let us describe the transfer function such system. To do this, we need to perform a Fourier transform:

$$Y(e^{j\omega}) = Ge^{-j\omega n_d} X(e^{j\omega})$$

What have we done?

- The Fourier transform is a linear operator so if we multiply by a number in the time-domain, the same number is multiplied in the frequency domain.
- A delay by n_d in the time-domain $x[n - n_d]$ corresponds to $e^{-j\omega n_d} X(e^{j\omega})$.

Table 4.4 provides more operations:

Table 4.4 Operational properties of the DTFT.

Property	Sequence	Transform
	$x[n]$	$\mathcal{F}\{x[n]\}$
1. Linearity	$a_1x_1[n] + a_2x_2[n]$	$a_1X_1(e^{j\omega}) + a_2X_2(e^{j\omega})$
2. Time shifting	$x[n - k]$	$e^{-jk\omega}X(e^{j\omega})$
3. Frequency shifting	$e^{j\omega_0 n}x[n]$	$X[e^{j(\omega-\omega_0)}]$
4. Modulation	$x[n] \cos \omega_0 n$	$\frac{1}{2}X[e^{j(\omega+\omega_0)}] + \frac{1}{2}X[e^{j(\omega-\omega_0)}]$
5. Folding	$x[-n]$	$X(e^{-j\omega})$
6. Conjugation	$x^*[n]$	$X^*(e^{-j\omega})$
7. Differentiation	$nx[n]$	$-j\frac{dX(e^{j\omega})}{d\omega}$
8. Convolution	$x[n] * h[n]$	$X(e^{j\omega})H(e^{j\omega})$
9. Windowing	$x[n]w[n]$	$\frac{1}{2\pi} \int_{2\pi} X(e^{j\theta})W[e^{j(\omega-\theta)}]d\theta$
10. Parseval's theorem	$\sum_{n=-\infty}^{\infty} x_1[n]x_2^*[n] =$	$\frac{1}{2\pi} \int_{2\pi} X_1(e^{j\omega})X_2^*(e^{j\omega})d\omega$
11. Parseval's relation	$\sum_{n=-\infty}^{\infty} x[n] ^2 =$	$\frac{1}{2\pi} \int_{2\pi} X(e^{j\omega}) ^2 d\omega$

System Function of Distortionless Systems

Since we have the relationship between the input and the output in the Fourier domain, we can now compute the system function or the transfer function of a distortionless system:

$$H(e^{j\omega}) = \frac{Y(e^{j\omega})}{X(e^{j\omega})} = Ge^{-j\omega n_d}$$

Let us now consider what are the properties of distortionless systems. To do this, we need to compute the manitude response and phase response of the system.

Magnitude response (gain)

The quantity $|H(e^{j\omega})|$ is known as the **magnitude response** or the **gain** of the system. The gain describes how much is the specific frequency ω of the input signal amplified or attenuated.

When the gain $|H(e^{j\omega})|$ is very small (close to zero) at a given frequency $\omega = \omega_0$ then that frequency component is filtered out (removed) from the input signal.



For this reason, LTI systems are often called filters. However, it is more appropriate to use the term filter for LTI systems designed to remove some frequency components from the input signal.

Computing Magnitude Response

So how do we go about computing the magnitude response?

First, we need to represent the transfer function into the frequency domain. This can be done in two ways depending on the given signal:

- Given a signal in the time-domain $h[n]$, we perform Fourier analysis
- Given a signal in the z -domain $H(z)$, we can evaluate the signal for $z = e^{j\omega}$.

The magnitude response of our distortionless system is G :

$$|H(e^{j\omega})| = G$$

This means that in order to have a distortionless system, all frequency components must be amplified or attenuated by the same amount.

Magnitude distortion

An LTI system is said to introduce **magnitude distortion** if the system applies unequal amount of amplification or attenuation to the different frequency components present in the input signal. For example, if a system attenuates any frequency below 1 KHz by a some factor k and frequencies above by a different factor $5k$ then the system is said to introduce magnitude distortion.

Formally, we say that a system introduces magnitude distoration if

$$|H(e^{j\omega})| \neq G$$

where G is a constant.

Phase response

The quantity $\angle H(e^{j\omega})$ is called the **phase response** of the system. The phase response describes the phase shift that the system applies to each frequency component of the input signal.



Given an LTI system's frequency response $H(e^{j\omega})$, we can plot the magnitude response and the phase response in order to observe how the system changes the amplitude and the phase of input signals at different frequencies.

The phase response of our distortionless system is:

$$\angle H(e^{j\omega}) = -n_d \omega$$

The phase response of the distortionless system is a **linear function** of the frequency ω . Plotting the function, we will observe that the line passes through the origin and the slope is $-n_d$.



Basically, the more the system delays the input signal, the larger is the phase that is added to the output signal. As long as the phase is a linear function, the system will not distort the input signal.

Characterising Phase Response

There are several ways of characterising the phase response of LTI systems. The frequency response of an LTI system describes how the system acts as a function of frequency:

$$H(e^{j\omega}) = |H(e^{j\omega})| e^{j\angle H(e^{j\omega})}$$

In other words, if a sinusoid of a certain frequency comes in, a sinusoid goes out at the same frequency but its amplitude is changed by $|H(e^{j\omega})|$ and phase is shifted by $\angle H(e^{j\omega})$

1) We can represent the phase response as the quantity $\angle H(e^{j\omega})$. However, this quantity is not uniquely defined.

We can write the phase as:

$$e^{j\angle H(e^{j\omega})}$$

A sinusoidal signal shifted by a multiple of 2π is the same as the original signal:

$$e^{j\angle H(e^{j\omega})} = e^{j(\angle H(e^{j\omega}) + m2\pi)}$$

where m is an integer.

This means that adding an integer multiple of 2π results in the same phase. So the phases are not unique.

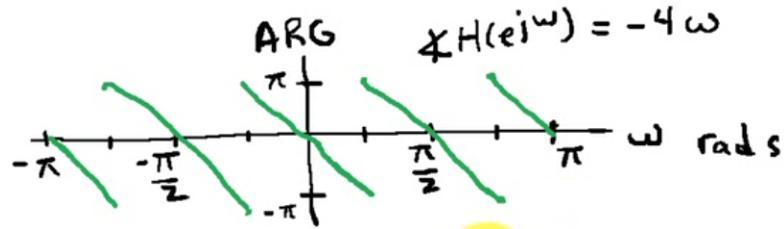
2) Principle value of the phase is defined as being within the following range:

\$\$

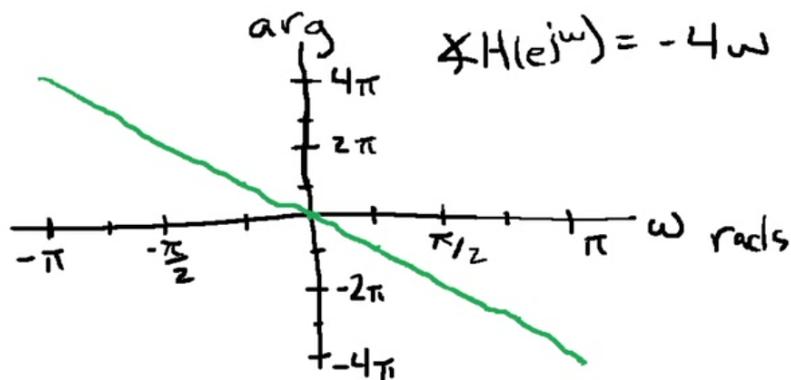
- $\pi < \text{ARG} [H(e^{j\omega})] < \pi$

The value $\text{ARG}[H(e^{j\omega})]$ is a value that is computed numerically. If the phase response exceeds the above limit, the function $\text{ARG}[H(e^{j\omega})]$ is discontinuous.

If we take the principal value of the phase, we get the green curves:



3) Unwrapped phase denoted $\arg[H(e^{j\omega})]$



Notice that the unwrapped phase represents the phase as continuous function where the jumps in the principal values are removed. In other words, by unwrapping the phase, we removes the jumps of 2π .

Phase or delay distortion

An LTI system is said to introduce **phase distortion** if the system applies unequal amount of time delays to the different frequency components present in the input signal.

Formally, we say that a system introduces phase/delay distortion if the phase response is not a linear function of the frequency ω :

$$\angle H(e^{j\omega}) \neq -n_d \omega$$

where n_d is a constant.

A nonlinear phase response may lead to the "shape" of the signal being changed significantly.



Although, we have discussed magnitude and phase distortions separately, it is almost impossible to separate the effects of magnitude distortions and phase distortions.

Summary of Distortionless Systems

Basically, a distortionless system does not have any kind of distortion as we send signal through the system. Why are focusing on these systems? Because there is a different way of analysing this kind of system than we have seen previously.

An LTI system is said to have distortionless response if the system applies the same amount of amplification or attenuation *and* phase shifts to the different frequency components present in the input signal.

Formally, an LTI system is distortionless if:

(1) the magnitude response $|H(e^{j\omega})|$ is constant

$$|H(e^{j\omega})| = G$$

where G is constant.

(2) the phase response $\angle H(e^{j\omega})$ is a linear function of ω with slope $-n_d$ that passes through the origin $\omega = 0$:

$$\angle H(e^{j\omega}) = -n_d \omega$$

where n_d are constants.

Note the slope n_d can be obtained by (phase delay):

$$n_d = \frac{-\angle H(e^{j\omega})}{\omega}$$

Assuming that the phase response is continuous function, we can obtain the slope n_d by derivative (group delay):

$$n_d = -\frac{d\angle H(e^{j\omega})}{d\omega}$$



This means that for distortionless system the phase delay and the group delay are the same.

Phase Delay

The phase response $\angle H(e^{j\omega})$ describes the phase shift (in radians) applied to each sinusoidal component of the input signal. Sometimes, it is more meaningful to use the **phase delay**.

The phase delay is a function of frequency which is defined as:

$$\tau_{pd}(\omega) \equiv \frac{-\angle H(e^{j\omega})}{\omega}$$

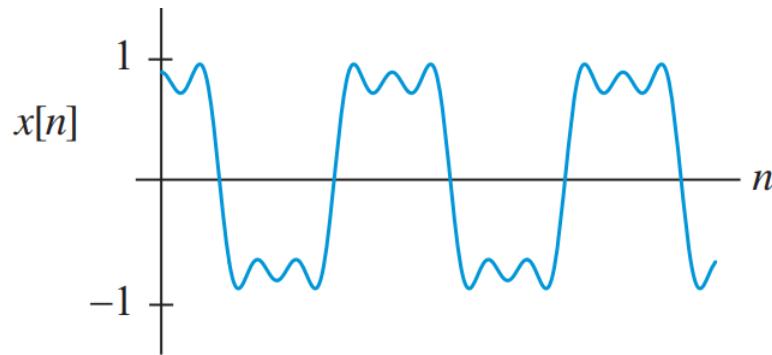
This function allows us to compute the phase delay of any given frequency.

Example

The following signal approximates a rectangular pulse:

$$x[n] = \cos(\omega_0 n) - \frac{1}{3}\cos(3\omega_0 n) + \frac{1}{5}\cos(5\omega_0 n)$$

The plot of the signal:



Now, let us assume that we have a system that applies a phase shift that is linear to the frequency. Look at system $y_4[n]$ in the table below:

Signal	c_1	c_2	c_3	ϕ_1	ϕ_2	ϕ_3	Phase shift
$x[n]$	1	-1/3	1/5	0	0	0	zero
$y_3[n]$	1	-1/3	1/5	$\pi/6$	$\pi/6$	$\pi/6$	constant
$y_4[n]$	1	-1/3	1/5	$-\pi/4$	$-3\pi/4$	$-5\pi/4$	linear
$y_5[n]$	1	-1/3	1/5	$-\pi/3$	$\pi/4$	$\pi/7$	nonlinear

For example, let us compute the phase delay of three different frequencies ω_0 , $3\omega_0$ and $5\omega_0$ where $\omega_0 = 0.004\pi$:

$$\tau_{pd}(\omega_0) = \frac{-(-\pi/4)}{0.004\pi} = 62.5 \text{ samples}$$

$$\tau_{pd}(3\omega_0) = \frac{-(-3\pi/4)}{3 \cdot 0.004\pi} = 62.5 \text{ samples}$$

$$\tau_{pd}(5\omega_0) = \frac{-(-5\pi/4)}{5 \cdot 0.004\pi} = 62.5 \text{ samples}$$

If the system applies a phase shift that is linearly proportional to the frequency of each the input component, the computing the phase delay for each frequency will result in the same value.

A system that applies constant or nonlinear phase shift to each frequency component will not yield the same phase delay of each frequency. If the phase delay is different for each frequency, then the output signal will be distorted i.e., the shape of the output will be different than the input signal.

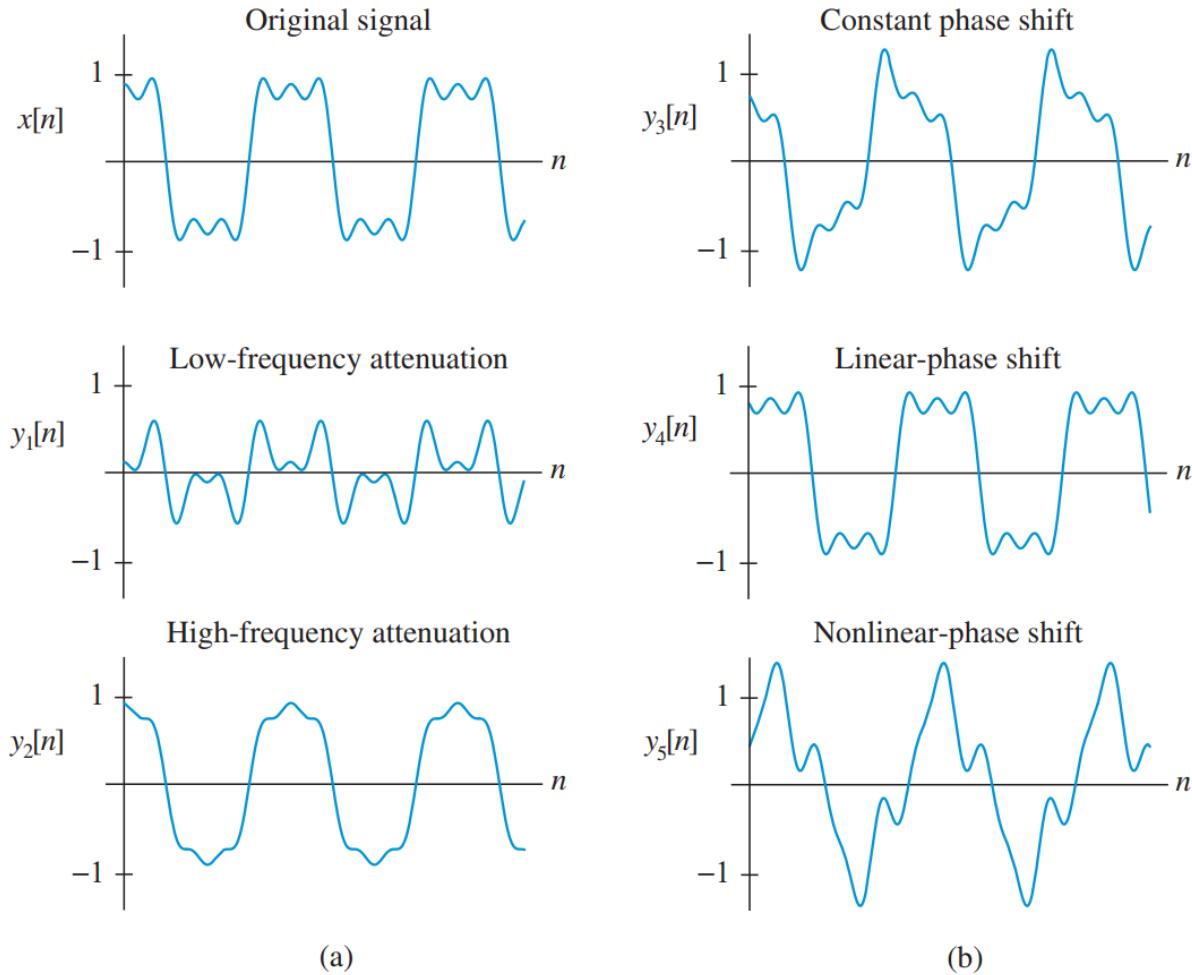


Figure 5.6 Magnitude (a) and phase (b) distortions. Clearly, it is difficult to distinguish the effects of magnitude and phase distortion.

Group Delay

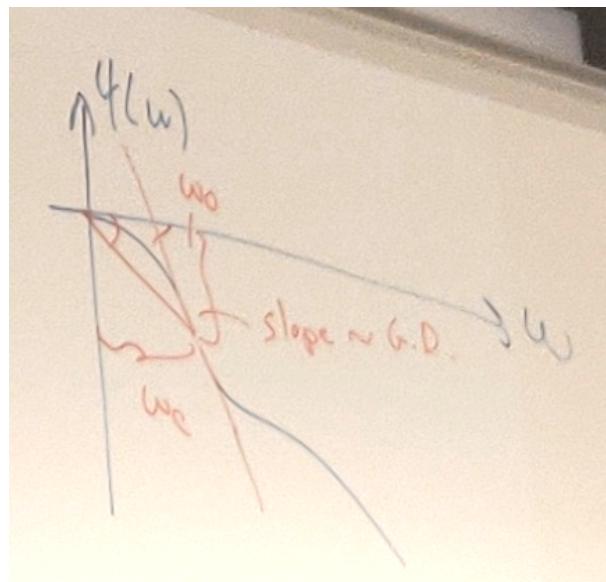
A convenient way to check the linearity of phase response is to use the **group delay**:

$$\tau_{gd}(\omega) \equiv -\frac{d\Psi(\omega)}{d\omega}$$

where $\Psi(\omega)$ is the continuous phase function

The group delay is a function of the frequency and is defined as the negative slope of the phase.

Let us plot the continuous phase function:



The group delay is the slope of phase function at a specific frequency ω_0



The phase delay and the group delay can be different. They are the same if the continuous phase is just a straight line. This is the case when the system is an **distortionless LTI system**.

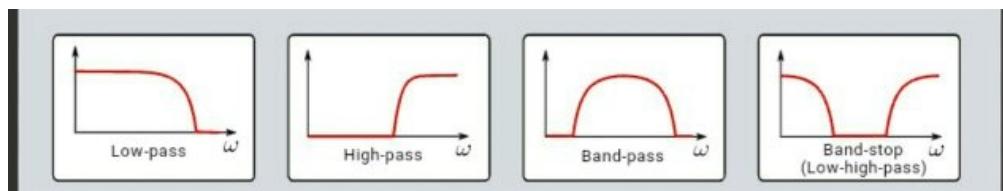
Example

Let us look at bandpass signal:

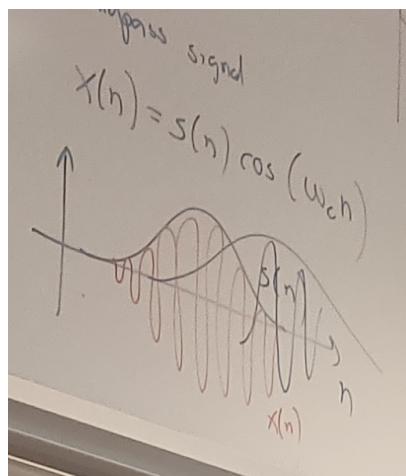
$$x[n] = s[n] \cos(\omega_c n)$$

where

- ω_c is the carrier frequency
- $s[n]$ is the envelope signal. This signal can consist of multiple frequency components. It can be a lowpass signal where maximum frequency content ω_{max} is much smaller than ω_c .
- $\cos(\omega_c n)$ is a high frequency signal known as high-frequency carrier.



We have a slowly oscillating signal $s[n]$ multiplied by a fast oscillating signal $\cos(\omega_c n)$



Let us examine what happens when we pass a bandpass signal $x[n]$ through a system where $\Psi(\omega)$ is approximately linear around the carrier frequency ω_c . We can work out an analytic expression.

$$y[n] = |H(e^{j\omega_c})| \cdot s[n - \tau_{gd}(\omega_c)] \cdot \cos(\omega_c(n - \tau_{pd}(\omega_c)))$$

There are three factors in the expression:

1. The first factor is the magnitude response or the gain of the system at the carrier frequency. We evaluate the system function at the carrier frequency ω_c
2. The lowpass signal is delayed by the group delay. This means that the envelope will be delayed some amount of samples given by $\tau_{gd}(\omega_c)$
3. The carrier signal will be delayed by the phase delay.

In summary:

- whenever we have a signal consisting of multiple frequency components like $s[n]$, an LTI system will introduce a delay in the form of group delay
- for a signal of a single frequency component, the LTI system will introduce a delay corresponding to a phase delay.

z -Transform of the Power Transfer Function

The frequency response $H(e^{j\omega})$ is equal to the system function $H(z)$ evaluated on the unit circle. We wish to find out whether there is a z -transform $R(z)$ such that

$$R(z)|_{z=e^{j\omega}} = |H(e^{j\omega})|^2 = H(e^{j\omega})H^*(e^{j\omega}). \quad (5.146)$$

The quantity $|H(e^{j\omega})|^2$ is called the power transfer function which relates input and output spectrum of a process. In some case, this is the z -transform of the auto-correlation.

So how do we find the z -transform of $H(e^{j\omega})H^*(e^{j\omega})$

First, we know that the z -transform of $H(e^{j\omega})$ is:

$$H(z) = \sum_{k=-\infty}^{\infty} h[k]z^{-k}$$

However, the z -transform of $H^*(e^{j\omega})$ is unknown. Let us focus on this part and phrase the problem in another way:

Can we find a $V(z)$ such if we evaluate $z = e^{j\omega}$ will yield $H^*(e^{j\omega})$?

$$V(z) \Big|_{z=e^{j\omega}} = H^*(e^{j\omega})$$

We know that:

$$H^*(e^{j\omega}) = \left(\sum_{k=-\infty}^{\infty} h[k]e^{-j\omega k} \right)^* = \sum_{k=-\infty}^{\infty} h^*[k]e^{j\omega k}$$

Notice that:

$$\sum_{k=-\infty}^{\infty} h^*[k]z^k \Big|_{z=e^{j\omega}} = \sum_{k=-\infty}^{\infty} h^*[k]e^{j\omega k} = H^*(e^{j\omega})$$

We are basically just going in circles so let us try something else.

$$V(z) = \sum_{k=-\infty}^{\infty} h^*[k]z^k$$

We can take the complex conjugate outside the parenthesis:

$$V(z) = \sum_{k=-\infty}^{\infty} h^*[k]z^k = \left(\sum_{k=-\infty}^{\infty} h[k](z^*)^k \right)^*$$

In order to have this look like a standard z -transform, we need to have z^{-k} . We can use the following trick. We can express the complex conjugate of z as:

$$z^* = \left(\frac{1}{z^*} \right)^{-1}$$

Putting this expression in the expression for $V(z)$ we get:

$$V(z) = \left(\sum_{k=-\infty}^{\infty} h[k] \left(\frac{1}{z^*} \right)^{-k} \right)^*$$

Finally, we get:

$$V(z) = H^* \left(\frac{1}{z^*} \right)$$

This means that the z -transform of $R(e^{j\omega}) = H(e^{j\omega})H^*(e^{j\omega})$ is:

$$R(z) = H(z)H^* \left(\frac{1}{z^*} \right)$$

when $h[n]$ is a complex number.

If $h[n]$ is real then we have:

$$R(z) = H(z)H \left(\frac{1}{z} \right)$$

In summary, to compute $H^*(1/z^*)$ is obtained by:

- conjugating the coefficients of $H(z)$, and
- replacing any z^{-1} by z

Example Problem

Suppose $H(z) = z + (1 + 2j)z^{-1} + 3z^{-2}$. What is $H^*(1/z^*)$?

Step 1: conjugate the coefficients of H :

- The coefficient of the first term is zero.
- The coefficient of the second term is $1 + 2j$. The conjugate is $1 - 2j$.
- The coefficient of the third term is 3. The conjugate is 3

Step 2: Write the terms of $H(z)$ using z^{-1} :

$$H(z) = (z^{-1})^{-1} + (1 + 2j)z^{-1} + 3(z^{-1})^2$$

Step 3: Replace any z^{-1} with z . First let :

$$H^*(1/z^*) = z^{-1} + (1 - 2j)z + 3z^2$$

Frequency Response for Rational System Function

All LTI systems of practical interest are described by a difference equation of the form (Eq. 3.76, p. 110):

$$y[n] = - \sum_{k=1}^N a_k y[n-k] + \sum_{k=0}^M b_k x[n-k]$$

The output of the system is a linear combination of input signal $x[n]$ and old output.

The system have a **rational system function**:

$$H(z) = \frac{\sum_{k=0}^M b_k z^{-k}}{1 + \sum_{k=1}^N a_k z^{-k}} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}} = \frac{B(z)}{A(z)}$$

The rational system function can be factorised in terms of its poles and zeros:

$$H(z) = \frac{b_0 \prod_{k=1}^M (1 - z_k z^{-1})}{\prod_{k=1}^N (1 - p_k z^{-1})}$$

where z_k are the zeros for $k = 1, 2, \dots, M$ and p_k are the poles for $k = 1, 2, \dots, N$. This allows us to clearly see the zeros and the poles.

Conjugate Reciprocal Pairs

Since we have $H(z)$ we can compute $R(z) = H(z)H^*(1/z^*)$:

$$R(z) = |b_0|^2 \frac{\prod_{k=1}^M (1 - z_k z^{-1})(1 - z_k^* z)}{\prod_{k=1}^N (1 - p_k z^{-1})(1 - p_k^* z)}$$

For each pole p_k of $H(z)$, there are two poles of $R(z)$ at p_k and $1/p_k^*$.

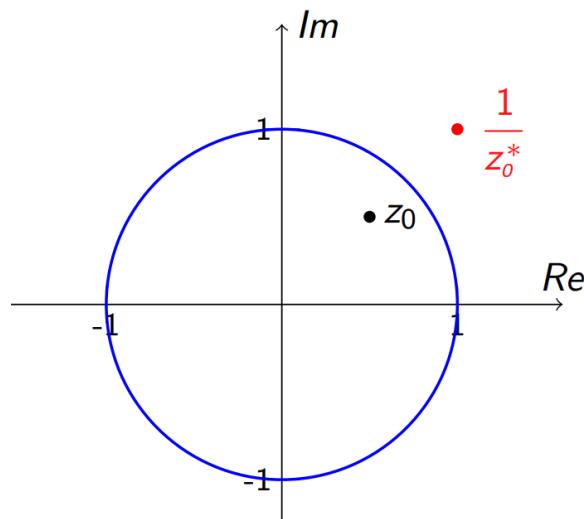
Similarly, for each zero z_k of $H(z)$, there are two zeros of $R(z)$ at z_k and $1/z_k^*$.

Therefore, the poles and zeros of $R(z)$ occur in **conjugate reciprocal pairs**.

For example, if $z_0 = re^{j\phi}$ is a pole (or zero), then $1/z_0^* = \frac{1}{r}e^{-j\phi}$ is a pole (or zero).

$$\frac{1}{z_0^*} = \frac{1}{(re^{j\phi})^*} = \frac{1}{re^{-j\phi}} = \frac{1}{r/e^{j\phi}} = \frac{1}{1} \cdot \frac{e^{j\phi}}{r} = \frac{1}{r}e^{j\phi}$$

If z_0 is inside the unit circle then its conjugate reciprocal $1/z_0^*$ is outside the unit circle.



If z_0 on the unit circle then its conjugate reciprocal $1/z_0^*$ must be in the same location on the unit circle.

Modelling Random Signals (example 5.8)

A standard problem in signal processing is given $R(z)$, what was the $H(z)$? We have measured the autocorrelation function for a given signal and we want to find the corresponding system in terms of the system function $H(z)$. This is an attempt to model what is going on in the system. signal model. It is a fancy way of modelling random signals. Once we have modelled the random signals, we can find the model parameters and describe the system in a more advanced framework than saying that the random signal is just noise. We will come back to this later in the course.

Suppose we have a second-order FIR filter with the system function:

$$H(z) = (1 - az^{-1})(1 - bz^{-1})$$

where a and b are real numbers between -1 and 1.

We can compute $R(z)$:

$$R(z) = H(z)H(1/z) = (1 - az^{-1})(1 - bz^{-1})(1 - az)(1 - bz)$$

Now, suppose we only know $R(z)$ and we want to find $H(z)$. We can come up with four equally valid choices for $H(z)$:

Given $R(z)$, by pairing different zeros, we can form four different second-order FIR systems:

$$H_1(z) = (1 - az^{-1})(1 - bz^{-1}), \quad (5.154a)$$

$$H_2(z) = (1 - az^{-1})(1 - bz), \quad (5.154b)$$

$$H_3(z) = (1 - az)(1 - bz^{-1}), \quad (5.154c)$$

$$H_4(z) = (1 - az)(1 - bz). \quad (5.154d)$$

As expected from (5.153), these systems have the same magnitude response but different phase responses. ■

We need figure out which of these system function that we like the best and why.

We will come back to this later in the course.

5.9 Allpass systems

Systems without magnitude distortion are known as **allpass systems**:

- Allpass systems have a constant magnitude response.
- Allpass systems can be completely described by their phase response.

Formally, we say that the frequency response of an allpass filter has a constant ($G > 0$) at all frequencies:

$$|H(e^{j\omega})| = G$$

But there are no constraint on the phase responses.

The simplest allpass filter is system that delays the signal by k samples:

$$H_{AP}(z) = z^{-k}$$

A more interesting, non-trivial family of allpass systems (known as **dispersive allpass systems**) is:

$$H_k(z) = \frac{z^{-1} - p_k^*}{1 - p_k z^{-1}}$$

We have have a zero at p_k^* and a pole at p_k .

Notice that there is no gain in the system but only phase delay.

In the textbook, the system is analysed and we the following results if we put our pole p_k at $r_k e^{j\phi_k}$:

For a first order allpass system with $p_k = r_k e^{j\phi_k}$ the magnitude, phase and group delay responses are given by

$$|H_k(e^{j\omega})| = 1 \quad (1)$$

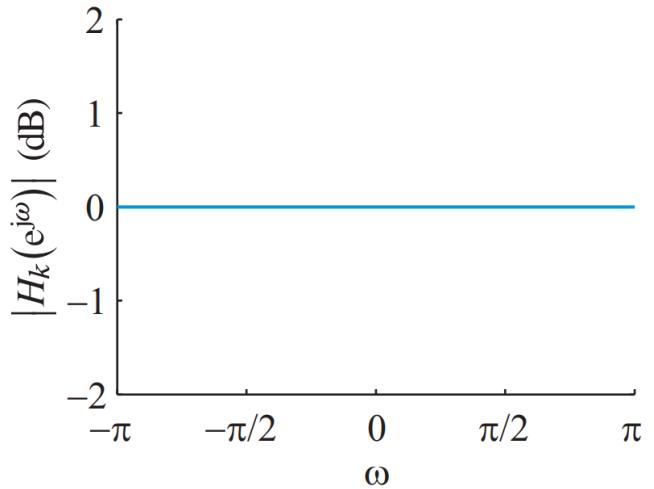
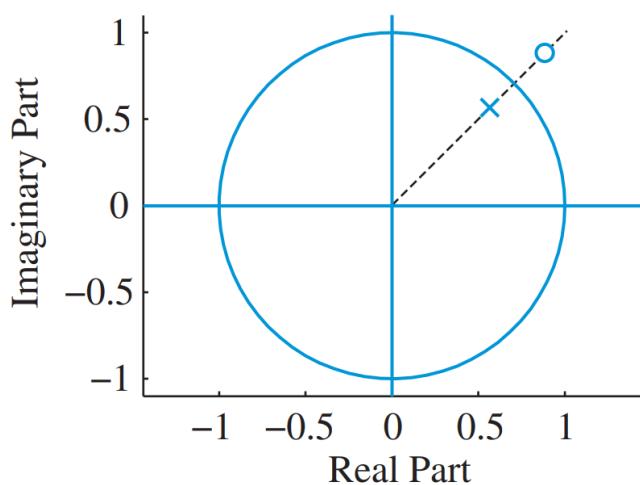
$$\angle H_k(e^{j\omega}) = -\omega - 2 \tan^{-1} \left(\frac{r_k \sin(\omega - \phi_k)}{1 - r_k \cos(\omega - \phi_k)} \right) \quad (2)$$

$$\tau_k(\omega) = \frac{1 - r_k^2}{1 + r_k^2 - 2r_k \cos(\omega - \phi_k)} \quad (3)$$

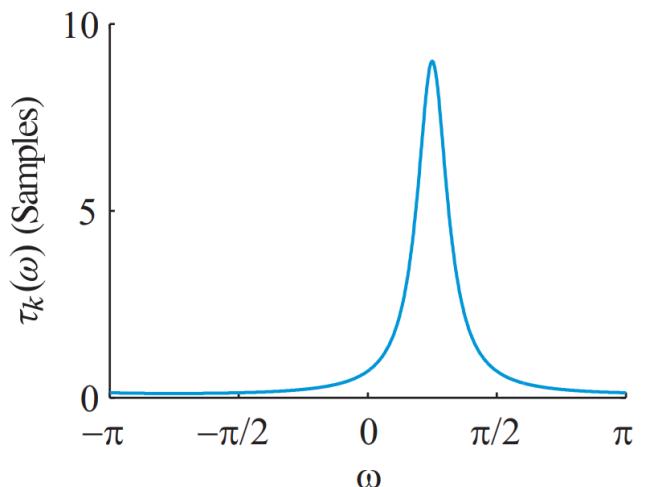
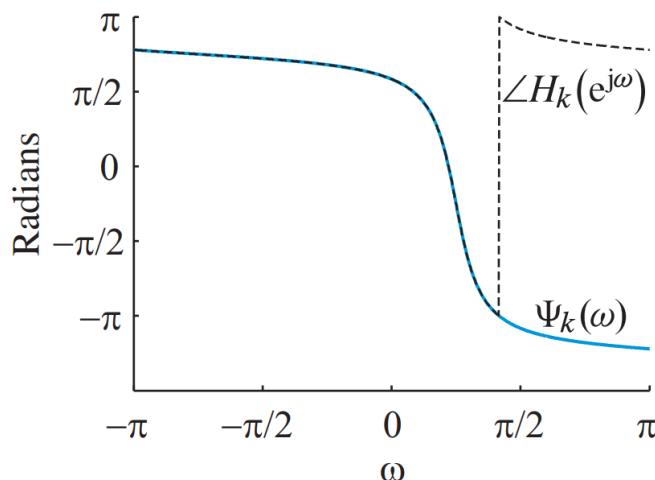
1. The magnitude response is 1
2. The phase response
3. The group delay

Looking at these equations, we see they are complex non-linear functions.

If we let $p_k = 0.8e^{j\pi/4}$ then the pole is located inside the unit circle in order for the system to be stable. Also notice that the zero is outside the unit circle, because it is set as $1/p_k^*$. The magnitude response is 1 or $\log(1) = 0$ dB.



We can compute the two phase functions; the unwrapped/continuous phase function $\Psi_k(\omega)$ and the phase function. We have a jump at $\pi/4$. This is shown on the left figure. If we take the derivative of $\Psi_k(\omega)$, when we get the plot on the right (the group delay). Notice that we get a peak at $\pi/4$.



In [6]:

```
np.pi/4
```

Out[6]:

```
0.7853981633974483
```

In [7]:

```
(3*np.pi)/4
```

Out[7]:

```
2.356194490192345
```

In summary: If we send a signal through this allpass filter, then frequencies located at $\pi/4$ will be delayed by some number of samples compared to any other frequencies. Notice there is no change in gain in this system.

Cascading Multiple First-Order Allpass Sections

Higher order allpass systems can be obtained by cascading multiple first-order sections:

$$H_{ap}(z) = e^{j\beta} \prod_{k=1}^N \frac{z^{-1} - p_k^*}{1 - p_k z^{-1}}$$

where β is a constant (usually $\beta = 0$ so $e^{j\beta} = 1$)

If we take this cascade of N first-order allpass filters and multiply it out, then we get a rational system function:

$$H_{AP}(z) = z^{-N} \frac{1 + a_1^* z + a_2^* z^2 + \cdots + a_N^* z^N}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}} = z^{-N} \frac{A^*(1/z^*)}{A(z)}$$

where $A(z)$ are polynomials. We say that $A(z)$ and $z^{-N} A^*(1/z^*)$ for a **conjugate reverse pair**.

Multiplying z^{-N} we get:

$$H_{AP}(z) = \frac{a_N^* + a_{N-1}^* z^{-1} + \cdots + z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}}$$

Invertibility and Minimum phase system

Definition: An LTI system $H(z)$ with input $x[n]$ and output $y[n]$ is said to be invertible if we can uniquely determine $x[n]$ from $y[n]$. In other words, there has to be a one-to-one mapping between input and output for each sample.

Definition: A stable-causal system has a stable-causal inverse if and only if all poles and zeros are inside the unit circle $|z| = 1$.

Definition: A causal and stable LTI system with a causal and stable inverse is known as a **minimum-phase system**.

We can write any rational system function that does not have zeros on the unit circle as:

$$H(z) = H_{min}(z)H_{AP}(z)$$

where $H_{min}(z)$ is the mini

To factor

Invertibility, minimum-phase and allpass

Table of Contents

Allpass systems.....	1
Minimum-phase systems.....	3
Invertibility.....	3
When is a filter invertible?.....	4
Filter Decomposition.....	4
Magnitude Response of Decomposition.....	5
Phase response of the Decomposition.....	5
Why is the decomposition useful?.....	6
Minimum Delay Property.....	7
Exam 2015 Problem 4: True/False.....	8
A serial connection of two all-pass filters is also an all-pass filter.....	8
Is a system invertible?.....	8
Is a filter invertible if the poles of the system function lies within the unit circle?.....	9
Quiz: can a system with zeros located exactly on the unit circle be inverted?.....	10
Quiz: Determine a minimum-phase filter based on impulse response.....	10
Quiz: Has a linear phase filter a mixed group delay?.....	12
Problem: Decompose a filter into minimum-phase and allpass.....	14
Problem: Decompose a filter into minimum-phase and allpass.....	15
Exam 2017 Problem 1: Decompose a Linear Phase Filter.....	17
1) Rewrite the system function as a product of a minimum phase filter and an all-pass filter.....	17
2) Discuss what happens to a signal if it is filtered using only instead of ?.....	18
ADSI Problem 1.2.....	18
1) Compute the impulse response of allpass filter.....	19
2) Quantisation errors.....	19
ADSI Problem 1.4: Filter decomposition.....	21
1) Decompose a FIR filter with one zero outside the unit circle.....	21
2) Show that the system and its correspond minimum-phase have the same magnitude response.....	22
3) Decompose a filter that has two zeros outside the unit circle.....	23
[✓] ADSI Problem 1.5 Filter decomposition.....	25
1) Show that a FIR filter with a difference equation is not minimum-phase.....	25
2) Find the difference equation for the corresponding minimum-phase FIR filter.....	26
ADSI Problem 1.8: Designing all-pass filters.....	28
1) Use iirgrpdelay in MATLAB to design a 4th order all-pass filter.....	28
2) What happens as you increase the order of the filter?.....	29
.....	29
Problem 2: Allpass filters.....	29
1) Show that the system function $H(z)$ is an allpass filter.....	29
2) Realise an allpass filter as an all-pole lattice filter.....	30
Functions.....	31

Allpass systems

The definition of allpass is that the magnitude of its frequency response is constant G (usually $G = 1$) for all frequencies.

$$|H(e^{j\omega})| = G. \quad (5.155)$$

We can say that:

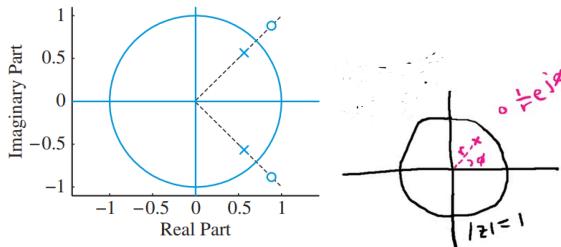
- an allpass system treats all frequencies identically with respect to gain
- an allpass system preserves the power or energy of their input signals

The system function of an allpass system has the poles and zeros occur in conjugate reciprocal pairs

$$H_{ap}(z) = e^{j\beta} \prod_{k=1}^N \frac{z^{-1} - p_k^*}{1 - p_k z^{-1}}, \quad (5.158)$$

where β is constant that is usually set to $\beta = 0$.

So if we have a pole $p_k = re^{j\phi}$ then the zero is located at $\frac{1}{p_k^*} = \frac{1}{r}e^{j\phi}$

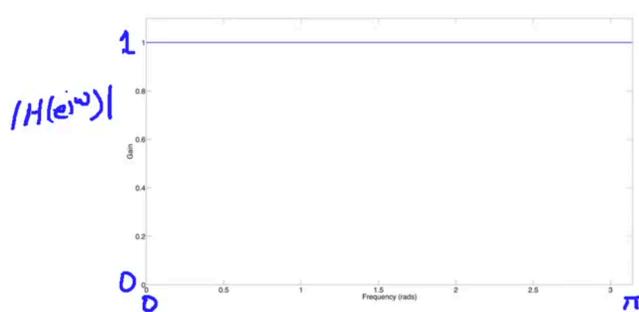
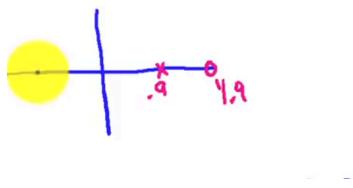


A second-order allpass filter is given by:

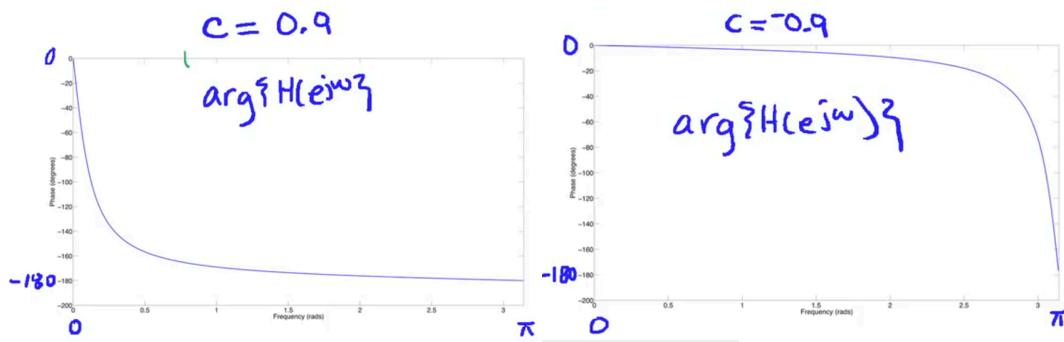
$$H_{ap}(z) = \frac{a_2^* + a_1^* z^{-1} + z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = z^{-2} \frac{1 + a_1^* z + a_2^* z^2}{1 + a_1 z^{-1} + a_2 z^{-2}}, \quad (5.164)$$

Example: All-pass System

$$H(z) = \frac{z^{-1} - c^*}{1 - c z^{-1}}$$



Although the magnitude response is the same, the phase response changes depending on the pole.



Minimum-phase systems

A stable, causal system has a stable, causal inverse
if and only if

all poles and zeros are inside $|z|=1$

called: Minimum phase system

A causal and stable LTI system with a causal and stable inverse is known as a **minimum-phase** system

If poles and zeros are outside the unit circle, we call it **maximum-phase**.

If some are outside and others inside the unit circle, it is known as **mixed-phase**.

Invertibility

5.16 Invertibility and minimum-phase systems

$x(n)$ $\boxed{h(n)}$ $y(n)$ $\boxed{h_{inv}(n)}$ $x(n)$

$$H(z) = \frac{\sum_{k=0}^M b_k z^{-k}}{1 + \sum_{k=1}^N a_k z^{-k}} = \frac{B(z)}{A(z)}$$

$$H_{inv}(z) = \frac{A(z)}{B(z)} \quad \text{zeros} \leftrightarrow \text{poles}$$

$$H(z) H_{inv}(z) = 1 \Leftrightarrow H_{inv}(z) = \frac{1}{H(z)}$$

When is a filter invertible?

A filter $H(z) = \frac{B(z)}{A(z)}$ is said to be stable and causal if all the poles of $H(z)$ are inside the unit circle.

If the inverse filter $H_{\text{inv}}(z) = \frac{A(z)}{B(z)}$ has to be stable and causal, then all the poles of $H_{\text{inv}}(z)$ must be inside the unit circle or equivalently all zeros of $H(z)$ must be inside the unit circle.

In practice, we say that a system is only invertible if its zeros and poles are inside the unit circle.

For a filter to be invertible, all its zeros and poles must be inside the unit circle. In other words, the filter must be minimum-phase.

Filter Decomposition

Any rational system function $H(z)$

$$H(z) = \underbrace{H_{\text{min}}(z)}_{\text{minimum phase}} \cdot \underbrace{H_{\text{ap}}(z)}_{\text{all pass}}$$

Any rational system function can be factorised into two system functions:

$$H(z) = H_{\text{min}}(z) \cdot H_{\text{ap}}(z)$$

By decomposing $H(z)$ we made a new polynomial that consists of two parts:

- Minimum-phase system: all zeros are inside the unit circle
- All-pass system: which will have zeros outside the unit circle. We are basically storing all the phase in the all-pass filter instead

To factorise or decompose, we take following steps:

1. Take the zeros of $H(z)$ that lie outside the unit circle and move them to the allpass system function $H_{\text{ap}}(z)$
2. Add poles to the allpass system in conjugate reciprocal locations of the zeros
3. To keep the equation balanced, we add zeros in the minimum-phase system function to even out the poles that we added to the allpass system function

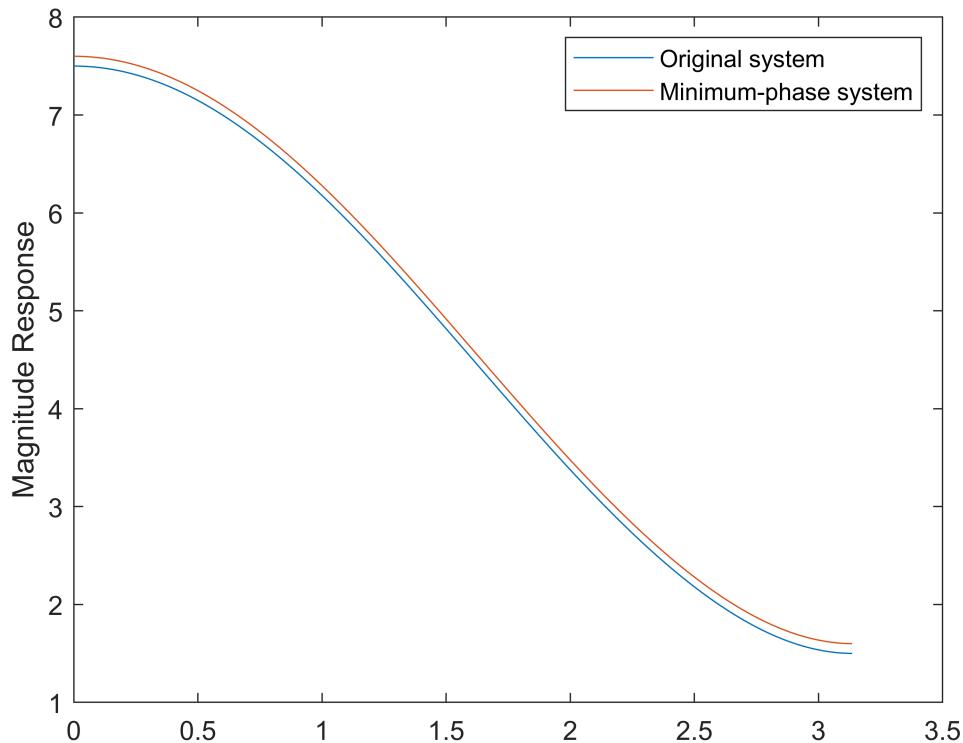
Since the poles that we added to the allpass system were all inside the unit circle, the zeros that we added to the minimum-phase system are also inside the unit circle.

Magnitude Response of Decomposition

Plotting the magnitude response of the original system and the minimum-phase system, we observe that they are on top of each other

```
[H_orig, w] = freqz([1, 4.5, 2]); % Original
[H_min, w] = freqz([4, 3, 0.5]);
[H_ap, w] = freqz([1/4, 1], [1, 1/4]);

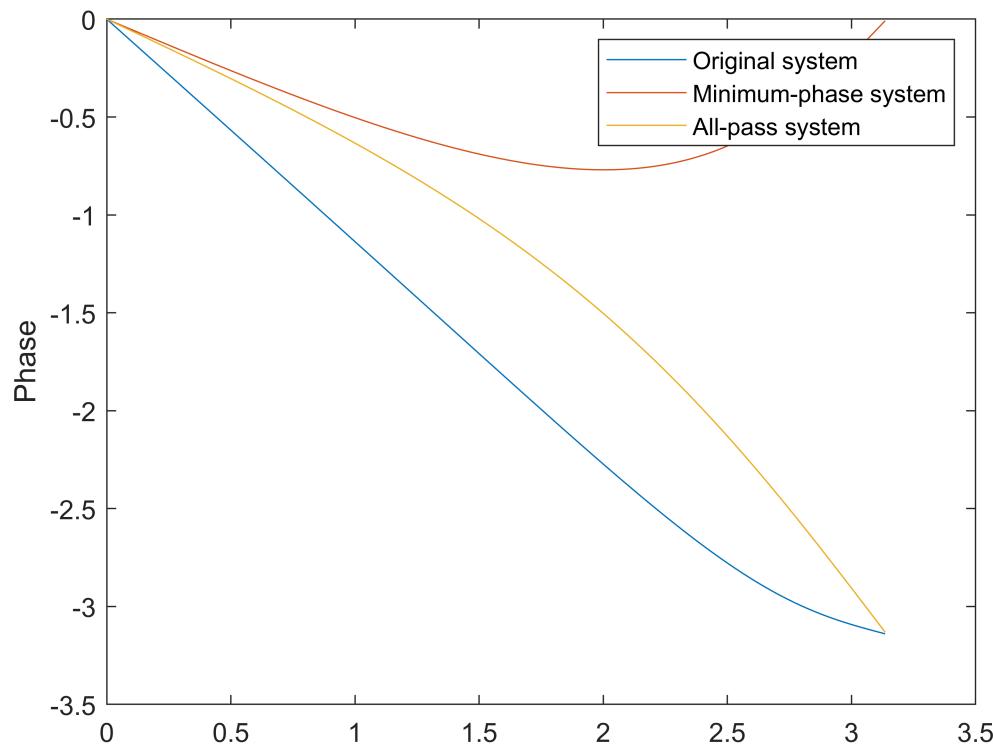
% Add 0.1 otherwise they will be on top of each other
plot(w, abs(H_orig), w, abs(H_min)+0.1)
legend('Original system', 'Minimum-phase system')
ylabel('Magnitude Response')
```



Phase response of the Decomposition

We can plot the phase of the three systems. The phase of the original system is hidden in the all-pass system. So by decomposing a filter into minimum-phase and all-pass systems, the excess phase is stored in the all-pass system. The phase lag of a system with poles and zeros inside the unit circle is less than the original system which had the identical magnitude response.

```
plot(w, angle(H_orig), w, angle(H_min), w, angle(H_ap))
ylabel('Phase')
legend('Original system', 'Minimum-phase system', 'All-pass system')
```



Why is the decomposition useful?

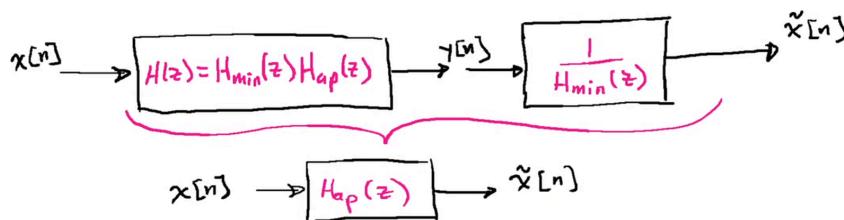
This decomposition is useful if we want to find an approximate inverse system for $H(z)$.

The minimum-phase component of any system can be inverted. This allows us to take any system, run a signal $x(n)$ through it and run the output $y(n)$ through the inverse of the minimum-phase part.

The end result is the same as passing the signal $x(n)$ through an allpass filter.

What we achieve is that:

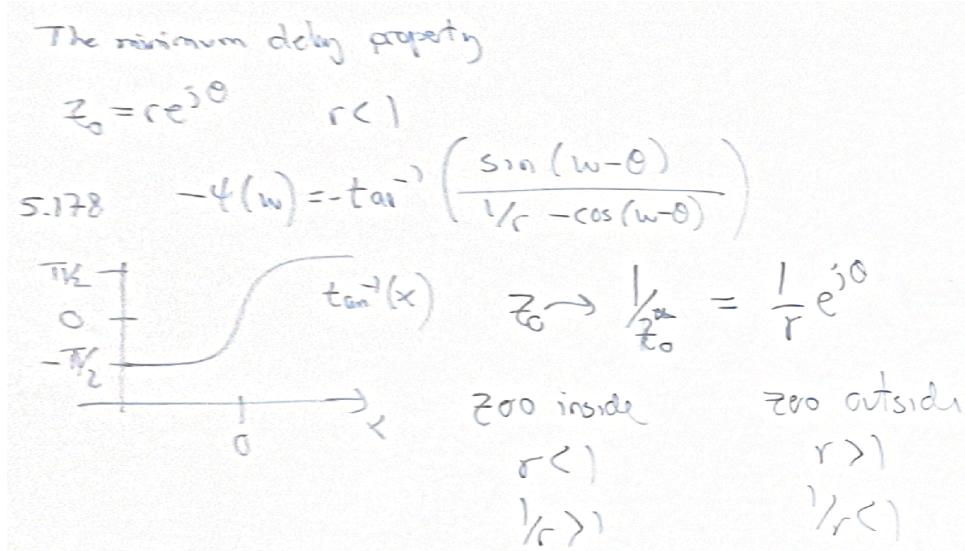
- no magnitude distortion meaning that the magnitude of each frequency is not altered by the process
- we only get phase distortion (which may not be important depending on the application)



The magnitude response of an allpass filter is constant for all frequencies, however its phase response may differ. For this reason, allpass filters are used for **delay equalisation**.

Minimum Delay Property

What happens when we take a zero and flip it. We know that the magnitude response does not change.



By setting $r < 1$ we specifically place the zero inside the unit circle.

If we flip z_0 to $\frac{1}{z_0} = \frac{1}{r} e^{j\phi}$

If we have zero inside $r < 1$ then $\frac{1}{r} > 1$

If we have zero outside $r > 1$ then $\frac{1}{r} < 1$

The only thing that changes in the phase response when we flip z_0 to inside the unit circle is $\frac{1}{r}$.

$$-\Psi(\omega) = -\tan^{-1} \frac{\sin(\omega - \theta)}{1/r - \cos(\omega - \theta)}, \quad (5.178)$$

If we go from zero inside to outside, the $\frac{1}{r}$ becomes smaller. The tangent will increase and we get a larger number. This means that we get more phase. So we add phase as the zero goes from inside the unit circle to outside the unit circle.

If we look at the group delay:

$$\tau(\omega) = \frac{r - \cos(\omega - \theta)}{(r + 1/r) - 2 \cos(\omega - \theta)}. \quad (5.179)$$

By taking a zero that is inside the unit circle and putting it outside, we increase the group delay.

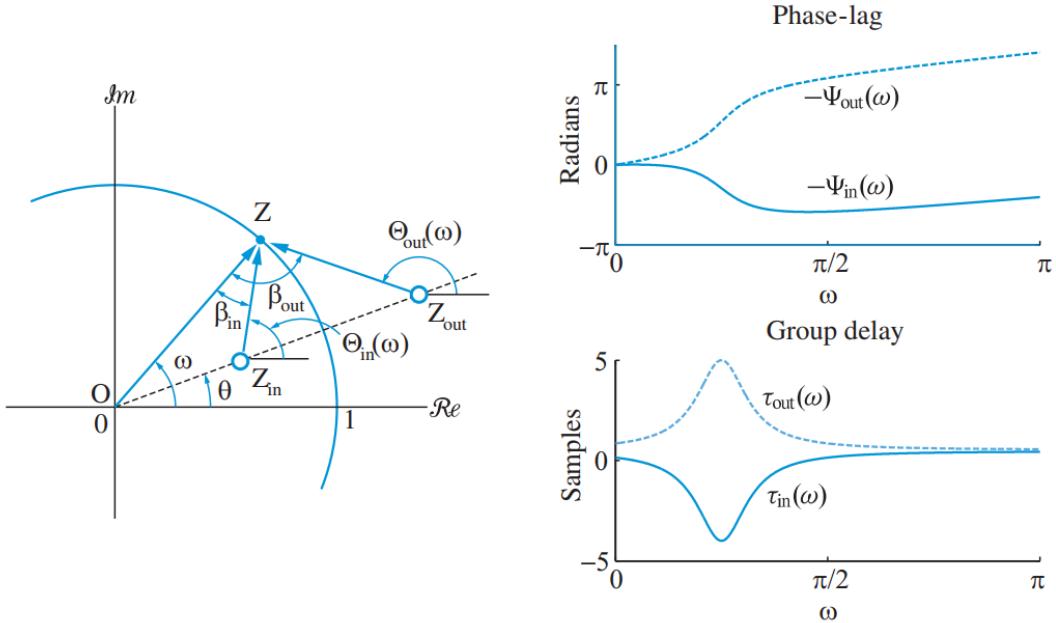


Figure 5.33 (a) Geometrical proof that a zero outside the unit circle introduces a larger phase-lag than a zero inside the unit circle. (b) Phase response and group delay for a zero at $z_{in} = 0.8e^{j\pi/4}$ and its conjugate reciprocal zero $z_{out} = (1/0.8)e^{j\pi/4}$.

Exam 2015 Problem 4: True/False

A serial connection of two all-pass filters is also an all-pass filter.

Answer: TRUE

The transfer function of a serial connection of two filters is just the individual transfer functions multiplied. For the magnitude part this becomes:

$$|H_{tot}(\omega)| = |H_{AP_1}(\omega)| \cdot |H_{AP_2}(\omega)| = 1 \cdot 1 = 1$$

Is a system invertible?

1. A system with $H(z) = \frac{1+3z^{-1}}{1+\frac{1}{4}z^{-1}}$ is invertible.

```
clear variables;
```

Answer: FALSE!

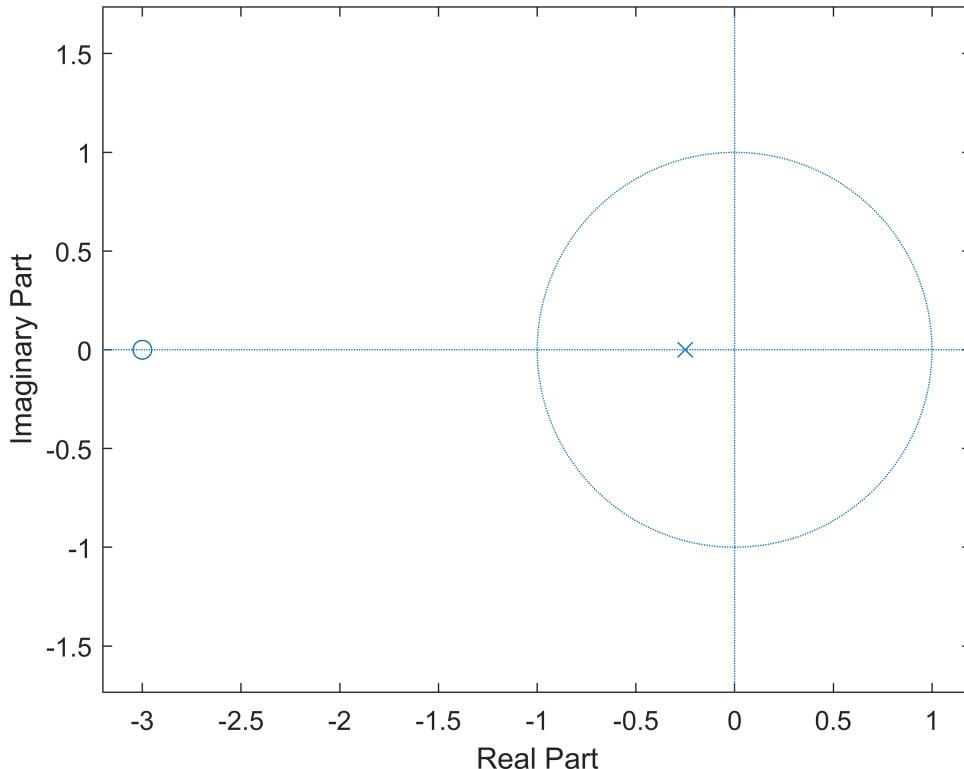
A filter $H(z) = \frac{B(z)}{A(z)}$ is said to be stable and causal if all the poles of $H(z)$ are inside the unit circle.

If the inverse filter $H_{\text{inv}}(z) = \frac{A(z)}{B(z)}$ has to be stable and causal, then all the poles of $H_{\text{inv}}(z)$ must be inside the unit circle or equivalently all zeros of $H(z)$ must be inside the unit circle.

In practice, we say that a system is only invertible if its zeros and poles are inside the unit circle (minimum phase).

Plotting the poles and zeros of the given system function, we observe that the zero is outside the unit circle.

```
b = [1, 3];  
a = [1, 1/4];  
zplane(b, a)
```



Is a filter invertible if the poles of the system function lies within the unit circle?

A filter with a rational system function $H(z) = B(z)/A(z)$ is invertible if the poles of the system function lies within the unit circle.

Answer: FALSE.

A filter $H(z) = \frac{B(z)}{A(z)}$ is said to be stable and causal if all the poles of $H(z)$ are inside the unit circle.

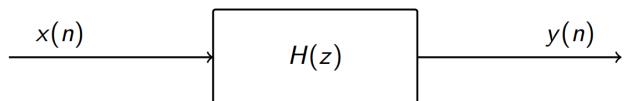
If the inverse filter $H_{\text{inv}}(z) = \frac{A(z)}{B(z)}$ has to be stable and causal, then all the poles of $H_{\text{inv}}(z)$ must be inside the unit circle or equivalently all zeros of $H(z)$ must be inside the unit circle.

So if we want the inverse filter to be stable, then the answer is false. Typically, we want an invertible filter to be minimum-phase which means that the filter *and* its inverse must be causal and stable.

For a filter to be invertible, zeros and poles must be inside the unit circle.

Quiz: can a system with zeros located exactly on the unit circle be inverted?

Consider a FIR filter with a number of zeros. Most of them are located inside the unit circle but a number of them is located exactly on the unit circle. There are no zeros outside the unit circle.



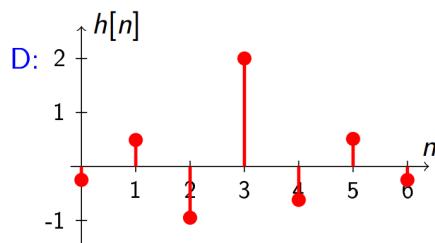
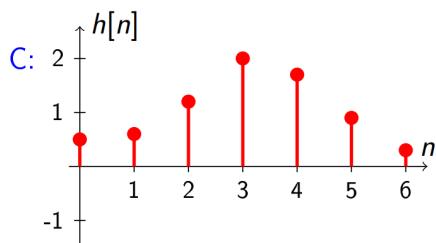
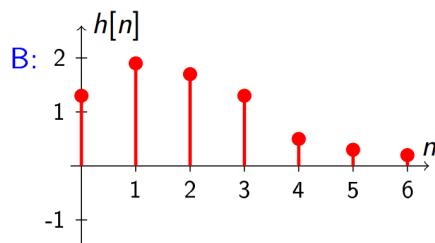
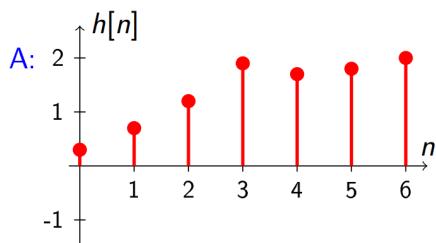
Can the action of the filter be undone by an inverse filter ?

- A: Yes, the filtering process can be undone
- B: No, the filtering process can't be undone
- C: We need to know more about $H(z)$

The answer is NO. If we have a FIR where the zero is exactly at 1 kHz. Given a signal, the filter will remove frequency component at 1 kHz. The problem is that the inverse filter cannot reconstruct the original signal because we have deleted information from the signal i.e., the missing frequency component cannot be recovered.

Quiz: Determine a minimum-phase filter based on impulse response

The impulse response from 4 different FIR filters is shown below.
 Which filter seems to be a minimum phase filter ?



The answer is B. Minimum-phase is the same thing as minimum group delay. If we put in energy any of these four filters, which one is the fastest to respond and pass the signal though. How fast does the system propagate the energy of a signal through the filter? If it is slow, then it is maximum-phase. If a filter is in-between then it is mixed-phase. If it is very phase, then it is minimum-phase.

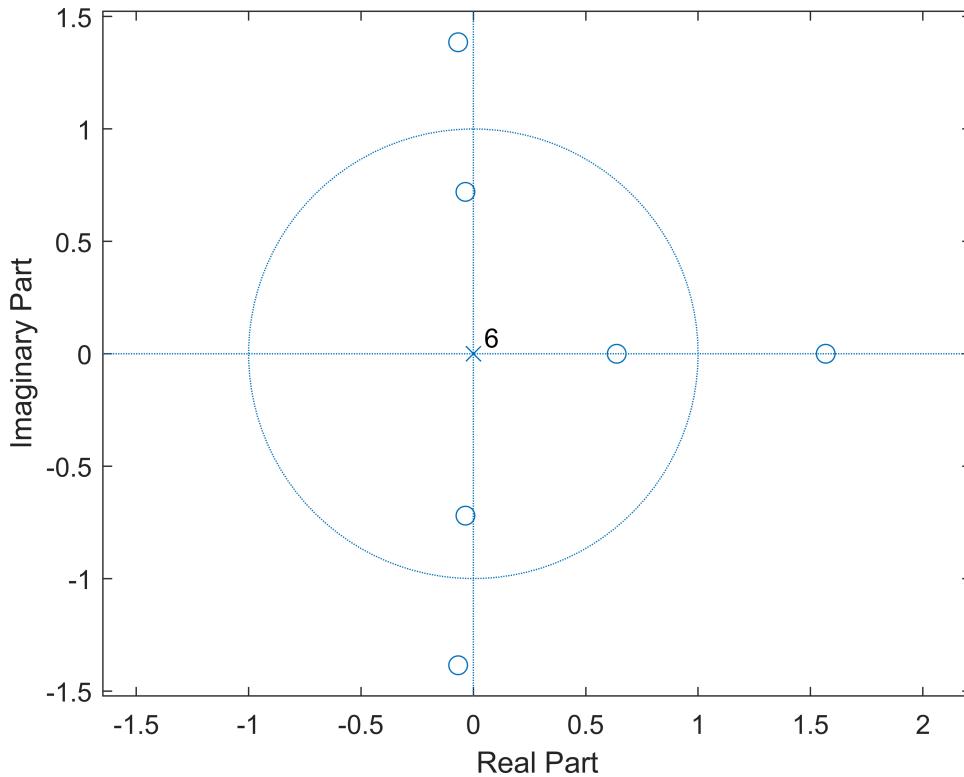
A and D has all the energy at the late end so it takes a while before all the energy passes through the system. Therefore, they have larger group delay than B, where the larger impulses are early.

Comparing B and C: B has energy at the beginning, whereas C the energy is in the middle.

- A: Maximum-phase
- B: Minimum-phase
- C: Mixed-phase
- D: Mixed-phase

Let us simulate filter D in MATLAB:

```
zplane([-1, 2, -3, 5, -3, 2, -1])
```



Quiz: Has a linear phase filter a mixed group delay?

A FIR filter has a linear phase response if the coefficients fulfil the symmetry requirement

$$b(k) = b(M - k), \quad \text{for } k = 0, 1, \dots, M$$

What is the classification of such a filter?

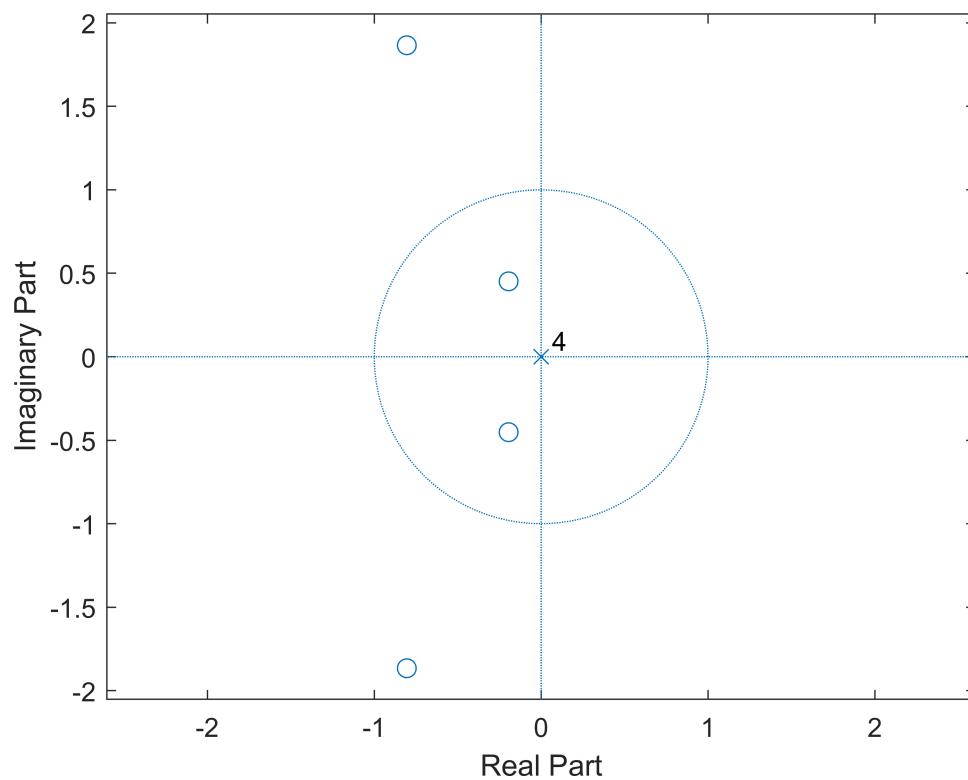
- A:** Minimum group delay
- B:** Mixed group delay
- C:** Maximum group delay
- D:** We need to know more about the $b(k)$'s

Recall that linear phase correspond to unity group delay.

The symmetry will be in the middle of the filter. The answer is B.

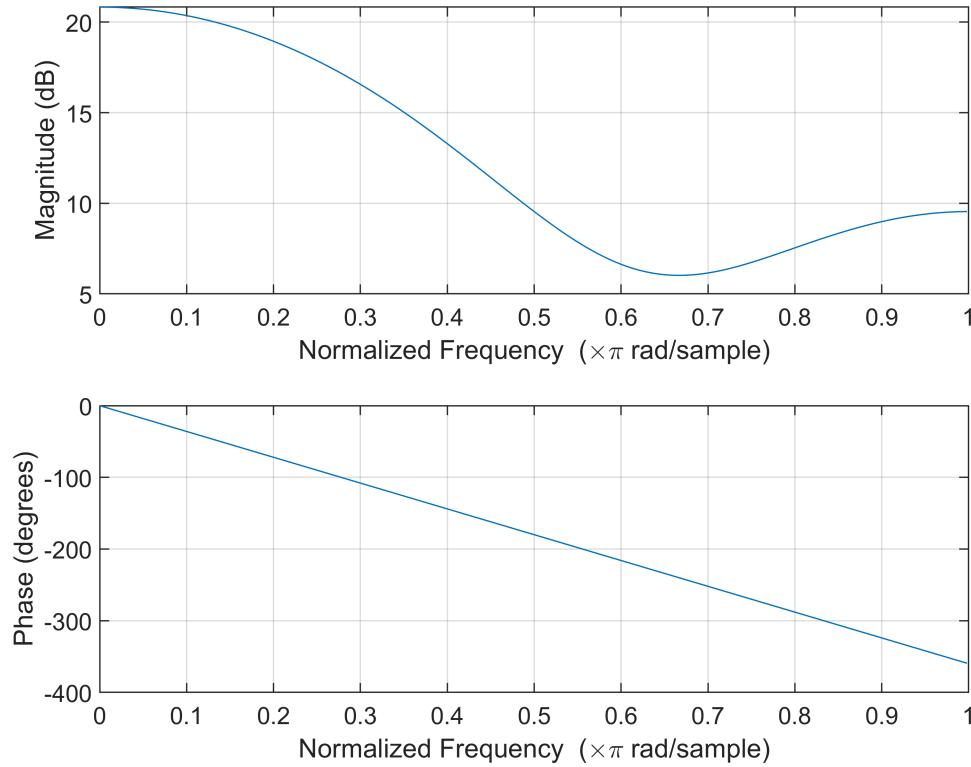
We can try it out in MATLAB. If we plot the a linear phase filter, we get two zeros inside and two zeros outside. Therefore, the filter is mixed-phase filter.

```
zplane([1, 2, 5, 2, 1])
```



The reason why the filter is called linear phase filter:

```
freqz([1, 2, 5, 2, 1])
```



Problem: Decompose a filter into minimum-phase and allpass

Decompose the system $H(z) = H_1(z)(1 - \beta z^{-1})$ where $|\beta| > 1$ and $H_1(z)$ is a minimum-phase system

Any system function can be decomposed into a product of a minimum-phase filter and an all-pass filter using the following formula:

$$H(z) = H_{\min}(z)H_{\text{ap}}(z)$$

To factorise or decompose, we take following steps:

1. Take the zeros of $H(z)$ that lie outside the unit circle and move them to the allpass system function $H_{\text{ap}}(z)$
2. Add poles to the allpass system in conjugate reciprocal locations of the zeros
3. To keep the equation balanced, we add zeros in the minimum-phase system function to even out the poles that we added to the allpass system function

Since the poles that we added to the allpass system were all inside the unit circle, the zeros that we added to the minimum-phase system are also inside the unit circle.

Step 1: We notice that $1 - \beta z^{-1}$ is not an allpass system. We want it to look like this:

$$H_{\text{ap}}(z) = e^{j\beta} \prod_{k=1}^N \frac{z^{-1} - p_k^*}{1 - p_k z^{-1}}, \quad (5.158)$$

So we start by factorising $-\beta$ out. By doing this, we put a zero into allpass system

$$H(z) = H_1(z)(-\beta) \left(-\frac{1}{\beta} + z^{-1} \right) = H_1(z)(-\beta) \left(z^{-1} - \frac{1}{\beta} \right)$$

Step 2: We need to put a pole into the allpass system that is in conjugate reciprocal locations of the zero

$$H(z) = H_1(z)(-\beta) \left(\frac{z^{-1} - \frac{1}{\beta}}{1 - \frac{1}{\beta^*} z^{-1}} \right)$$

Step 3: To keep the equality, we have to introduce a zero to the minimum-phase system function

$$H(z) = H_1(z)(-\beta) \left(1 - \frac{1}{\beta^*} z^{-1} \right) \left(\frac{z^{-1} - \frac{1}{\beta}}{1 - \frac{1}{\beta^*} z^{-1}} \right)$$

Now the decomposition is:

$$H(z) = H_{\text{min}}(z)H_{\text{ap}}(z)$$

where:

$$H_{\text{min}}(z) = H_1(z)(-\beta) \left(1 - \frac{1}{\beta^*} z^{-1} \right)$$

$$H_{\text{ap}}(z) = \frac{z^{-1} - \frac{1}{\beta}}{1 - \frac{1}{\beta^*} z^{-1}}$$

Suppose $H(z) = H_1(z)(1 - \beta z^{-1})$; $|\beta| > 1$, $H_1(z)$ min phase

$$\begin{aligned} 1) \quad H(z) &= H_1(z)(-\beta)(z^{-1} - \frac{1}{\beta}) \\ 2+3) \quad H(z) &= \underbrace{H_1(z)(-\beta)}_{H_{\text{min}}(z)} \underbrace{(1 - \frac{1}{\beta^*} z^{-1})}_{H_{\text{ap}}(z)} \end{aligned}$$

zero at $z = \frac{1}{\beta}$
 reflected in $H(z)$ inside $|z| = 1$


Problem: Decompose a filter into minimum-phase and allpass

Decompose the system $H(z) = 1 + 4.5z^{-1} + 2z^{-2}$

Any system function can be decomposed into a product of a minimum-phase filter and an all-pass filter using the following formula:

$$H(z) = H_{\min}(z)H_{\text{ap}}(z)$$

To factorise or decompose, we take following steps:

1. Take the zeros of $H(z)$ that lie outside the unit circle and move them to the allpass system function $H_{\text{ap}}(z)$
2. Add poles to the allpass system in conjugate reciprocal locations of the zeros
3. To keep the equation balanced, we add zeros in the minimum-phase system function to even out the poles that we added to the allpass system function

Since the poles that we added to the allpass system were all inside the unit circle, the zeros that we added to the minimim-phase system are also inside the unit circle.

Step 1: Take the zeros of $H(z)$ that lie outside the unit circle and move them to the allpass system function

First, we need to write it as a product of two system functions. We can do that by finding the zeros:

```
roots([1, 4.5, 2])
```

```
ans = 2x1
-4.0000
-0.5000
```

Now we can write the system as:

$$H(z) = (1 - (-0.5)z^{-1})((1 - (-4)z^{-1}))$$

$$H(z) = (1 + 0.5z^{-1})(1 + 4z^{-1})$$

$$\begin{aligned} H(z) &= 1 + 4.5z^{-1} + 2z^{-2} \\ H(z) &= (1 + 0.5z^{-1})(1 + 4z^{-1}) \\ z_0 &= \begin{cases} -0.5 \\ -4 \end{cases} \end{aligned}$$

One zero $z = -0.5$ is inside the unit circle and the other is outside $z = -4$. We want to flip the zero outside the unit circle to be inside the unit circle.

So we start by factorising 4 out. By doing this, we put a zero into allpass system.

$$H(z) = (1 + 0.5z^{-1})(4)\left(\frac{1}{4} + z^{-1}\right) = 4(1 + 0.5z^{-1})\left(z^{-1} + \frac{1}{4}\right)$$

Step 2: We need to put a pole into the allpass system that is in conjugate reciprocal locations of the zero

$$H(z) = 4(1 + 0.5z^{-1}) \left(\frac{z^{-1} + \frac{1}{4}}{1 + \frac{1}{4}z^{-1}} \right)$$

Step 3: To keep the equality, we have to introduce a zero to the minimum-phase system function

$$H(z) = 4(1 + 0.5z^{-1}) \left(1 + \frac{1}{4}z^{-1}\right) \left(\frac{z^{-1} + \frac{1}{4}}{1 + \frac{1}{4}z^{-1}} \right)$$

To tidy up, we will multiply the first three products together. Multiplying two polynomials together is the same as doing convolution:

```
4 * conv([1, 0.5], [1, 1/4])
```

```
ans = 1x3
    4.0000    3.0000    0.5000
```

So we can rewrite the system function as:

$$H(z) = (4 + 3z^{-1} + 0.5z^{-2}) \left(\frac{z^{-1} + \frac{1}{4}}{1 + \frac{1}{4}z^{-1}} \right)$$

Exam 2017 Problem 1: Decompose a Linear Phase Filter

Linear phase filters with the symmetry property are commonly applied in audio signal processing due to their frequency independent group delay.

Linear phase filters are FIR filters $H(z) = b_0 + b_1z^{-1} + \dots + b_Mz^{-M}$ with the symmetry property of the coefficients $b_k = b_{M-k}$ for $k = 0, 1 \dots M$. Such filters are commonly applied in e.g. audio signal processing due to their frequency independent group delay.

Consider the following linear phase filter

$$H(z) = 1 + 2.5z^{-1} + z^{-2}.$$

```
h = [1, 2.5, 1];
```

1) Rewrite the system function as a product of a minimum phase filter and an all-pass filter

- Rewrite $H(z)$ as a product of a minimum phase filter and an all-pass filter, i.e.

$$H(z) = H_{min}(z)H_{ap}(z).$$

Any system function can be decomposed into a product of a minimum-phase filter and an all-pass filter using the following formula:

$$H(z) = H_{\min}(z)H_{\text{ap}}(z)$$

To factorise or decompose, we take following steps:

1. Take the zeros of $H(z)$ that lie outside the unit circle and move them to the allpass system function $H_{\text{ap}}(z)$
2. Add poles to the allpass system in conjugate reciprocal locations of the zeros
3. To keep the equation balanced, we add zeros in the minimum-phase system function to even out the poles that we added to the allpass system function

Since the poles that we added to the allpass system were all inside the unit circle, the zeros that we added to the minimim-phase system are also inside the unit circle.

These steps are coded in MATLAB function (see at the end of the document):

```
[H_min, H_ap] = decompose_min_ap(h)
```

$H_{\min} =$

$$\frac{2}{z} + \frac{1}{2z^2} + 2$$

$H_{\text{ap}} =$

$$\frac{\frac{1}{z} + \frac{1}{2}}{\frac{1}{2z} + 1}$$

So we have:

$$H_{\min}(z) = 2 + 2z^{-1} + \frac{1}{2}z^{-2} \quad \text{and} \quad H_{\text{ap}}(z) = \frac{0.5 + z^{-1}}{1 + 0.5z^{-1}}$$

2) Discuss what happens to a signal if it is filtered using only $H_{\min}(z)$ instead of $H(z)$?

The two system functions $H(z)$ and $H_{\min}(z)$ have the same magnitude response, but their phase properties differ. In this particular case the $H(z)$ is a linear phase filter and the group delay through the filter is constant and independent of frequency. For the minimum phase system function the phase is not linear and the group delay becomes frequency dependent. A signal passing through $H_{\min}(z)$ instead of $H(z)$ will thus become phase distorted.

ADSI Problem 1.2

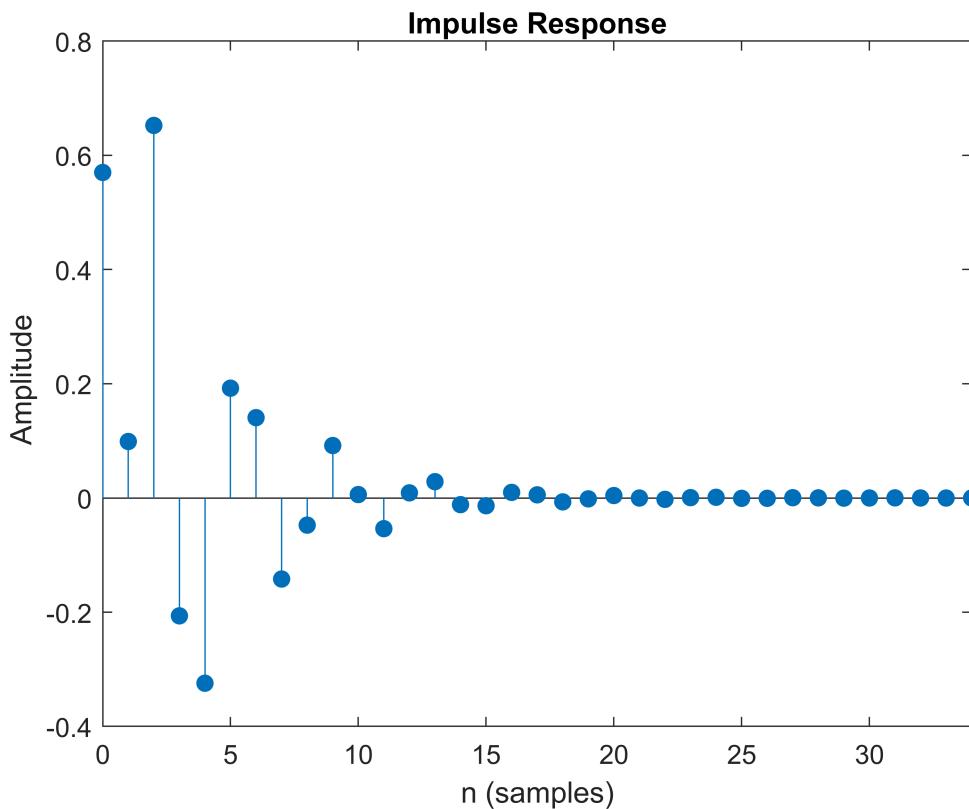
1) Compute the impulse response of allpass filter

An All-Pass filter has the following system function

$$H(z) = \frac{0.57 + 0.23z^{-1} + z^{-2}}{1 + 0.23z^{-1} + 0.57z^{-2}}$$

1. First, without performing calculations, try to predict what the impulse response of the filter will look like. Second, plot the impulse response and compare with your prediction.

```
b = [0.57, 0.23, 1];
a = [1, 0.23, 0.57];
impz(b, a);
```



2) Quantisation errors

Next, assume that the filter is to be implemented on a fixed point 16 bits signal processor, e.g. an Analog Devices Blackfin. This implies that the coefficients must be quantized to (1.15) format.

2. Is the filter still an all-pass filter when the coefficients have been quantized ?
3. If you recall how the calculation works, quantize the filter coefficients to 16 bits in (1.15) format.

```
b = [0.57, 0.23, 1];
a = [1, 0.23, 0.57];
[h, t] = impz(b, a);

bits = 16;
maxVal = (2^(bits-1))-1;
```

```
minVal = (2^(bits-1));
```

Normalise the impulse response between -1 and +1 assuming that the maximum value is positive.

```
normalisedH = h./max(h);
```

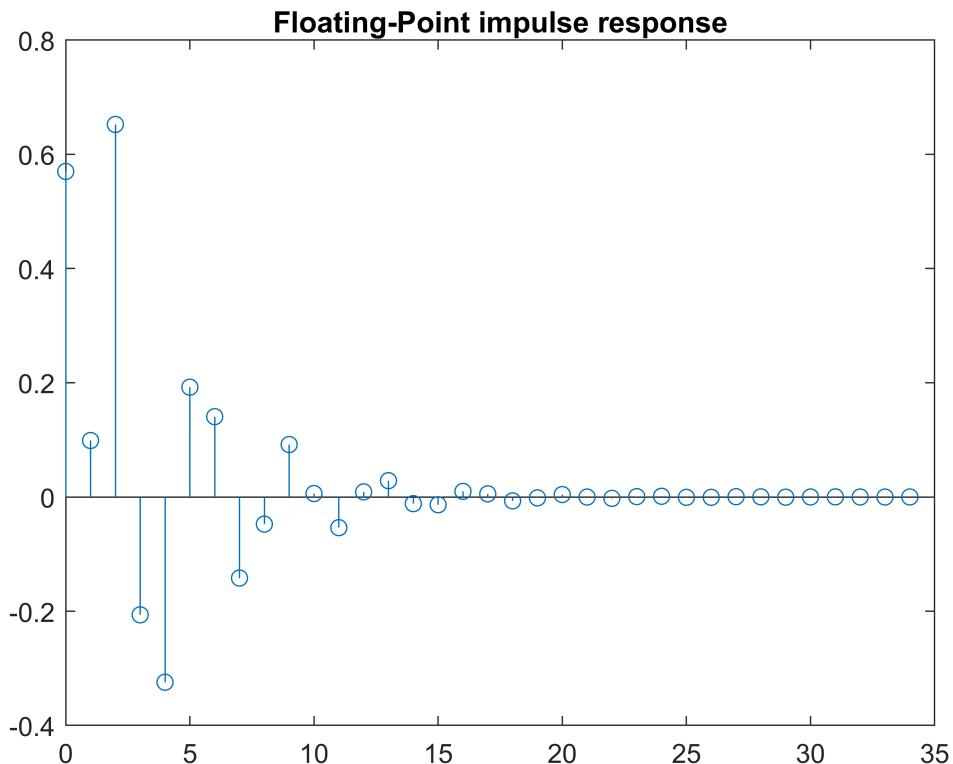
Quantise the positive values

```
quantisedH = normalisedH.* maxVal;
```

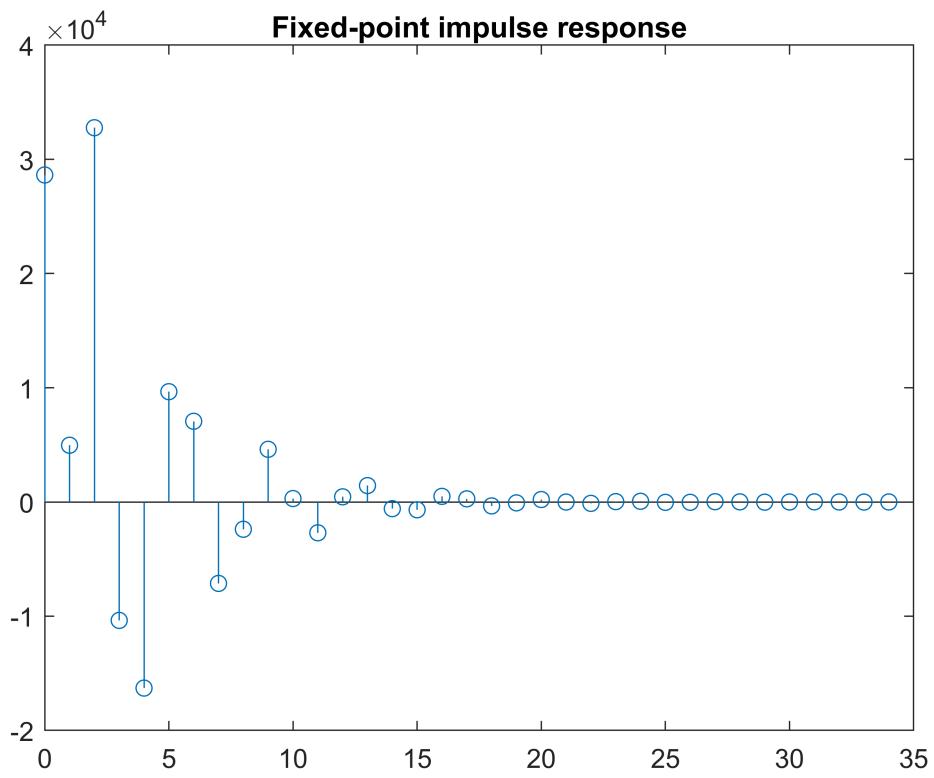
Find the index of negative values:

```
negIndices = find(normalisedH < 0);
quantisedH(negIndices) = normalisedH(negIndices).* minVal;
```

```
stem([0:34], h);
title('Floating-Point impulse response');
```



```
stem([0:34], quantisedH);
title('Fixed-point impulse response');
```



ADSI Problem 1.4: Filter decomposition

```
clear variables;
```

1) Decompose a FIR filter with one zero outside the unit circle

Let the system function for a FIR filter be given by

$$H(z) = 1 - 3z^{-1} + \frac{5}{2}z^{-2} - z^{-3}$$

1. Decompose the system function into a product of a minimum-phase filter and an all-pass filter, $H(z) = H_{min}(z)H_{ap}(z)$.

To decompose the system function, we take following steps:

1. Take the zeros of $H(z)$ that lie outside the unit circle and move them to the allpass system function $H_{ap}(z)$
2. Add poles to the allpass system in conjugate reciprocal locations of the zeros
3. To keep the equation balanced, we add zeros in the minimum-phase system function. This will even out the poles that we added to the allpass system function.

Since the poles that we added to the allpass system were all inside the unit circle, the zeros that we added to the minimim-phase system are also inside the unit circle.

This is coded in a MATLAB function:

```
h = [1, -3, 5/2, -1];
[H_min, H_ap] = decompose_min_ap(h)
```

$$\begin{aligned}H_{\min} &= \frac{3}{z} - \frac{2}{z^2} + \frac{1}{2z^3} - 2 \\H_{ap} &= -\frac{\frac{1}{z} - \frac{1}{2}}{\frac{1}{2z} - 1}\end{aligned}$$

We put everything together:

$$\begin{aligned}H(z) &= H_{\min}(z)H_{ap}(z) \\&= \left(-2 + 3z^{-1} - 2z^{-2} + \frac{1}{2}z^{-3}\right)\left(\frac{z^{-1} - 0.5}{1 - 0.5z^{-1}}\right)\end{aligned}$$

2) Show that the system and its correspond minimum-phase have the same magnitude response

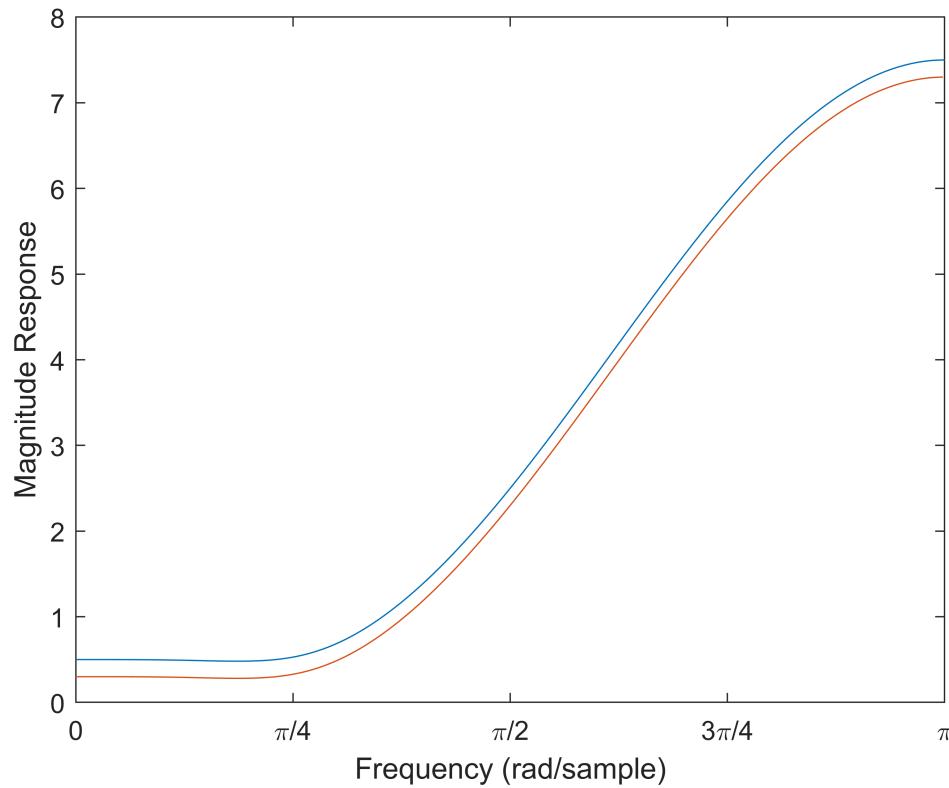
2. Demonstrate that $H(z)$ and $H_{\min}(z)$ have the same amplitude response.

To show that $H(z)$ and $H_{\min}(z)$ have the same magnitude response, we can plot them. If the two graphs are on top of each other then then they have the same magnitude response.

```
% Coefficients for H(z)
b = [1, -3, 5/2, -1];
a = 1;
[H, w] = freqz(b,a, 'whole');

% Coefficients for H_min(z)
H_min_b = [-2, 3, -2, 1/2];
H_min_a = 1;
[H_min_z, H_min_w] = freqz(H_min_b, H_min_a);
% plot(H_min_w, log10(abs(H_min_z)));

% The offset ensures that the two graphs are not on top of each other
offset = -0.2;
plot(w, abs(H), H_min_w, abs(H_min_z) + offset);
set(gca, 'XTick', 0:pi/4:pi)
set(gca, 'XTickLabel', {'0', '\pi/4', '\pi/2', '3\pi/4', '\pi'})
xlabel('Frequency (rad/sample)')
ylabel('Magnitude Response')
xlim([0, pi]);
```



3) Decompose a filter that has two zeros outside the unit circle

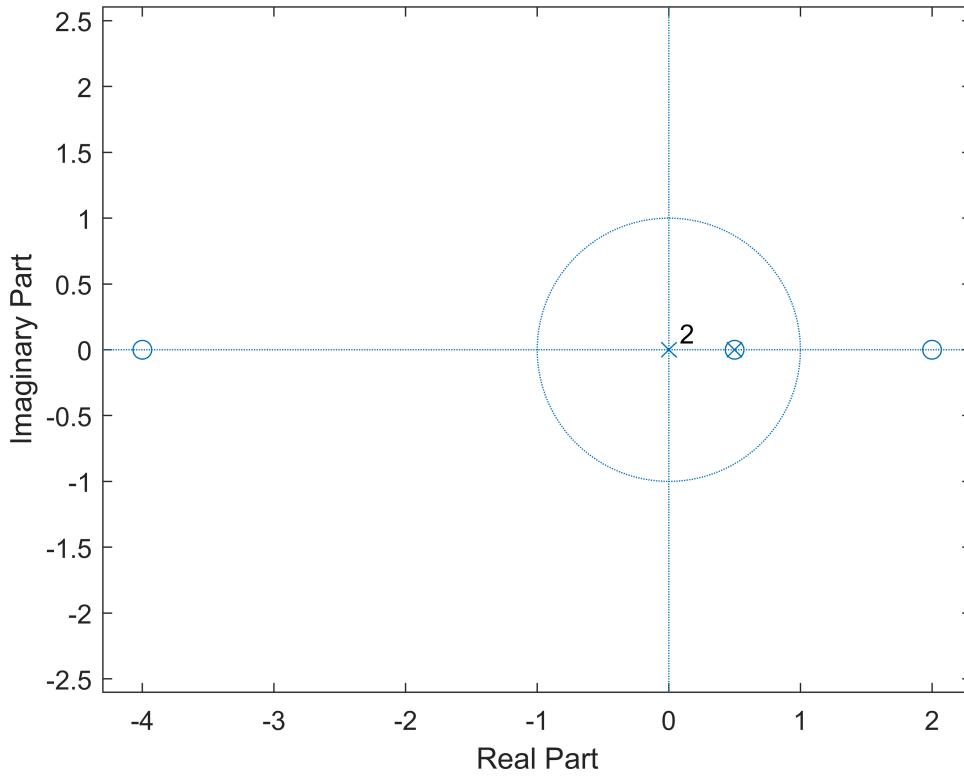
3. Repeat for the following filter

$$H(z) = \frac{1 + \frac{3}{2}z^{-1} - 9z^{-2} + 4z^{-3}}{1 - \frac{1}{2}z^{-1}}$$

```
b = [1, 3/2, -9, 4];
a = [1, -1/2];
```

Plot the zeros and the poles:

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



```
[H_min, H_ap] = decompose_min_ap(b)
```

$$\begin{aligned} H_{\text{min}} &= \frac{10}{z} - \frac{3}{z^2} - \frac{1}{2z^3} + \frac{1}{4z^4} - 8 \\ H_{\text{ap}} &= -\frac{\left(\frac{1}{z} - \frac{1}{2}\right) \left(\frac{1}{z} + \frac{1}{4}\right)}{\left(\frac{1}{2z} - 1\right) \left(\frac{1}{4z} + 1\right)} \end{aligned}$$

Expanding the numerator, we get:

```
syms z;
H_ap_num = expand(-1*(1/z - 1/2)*(1/z + 1/4))
```

$$H_{\text{ap_num}} = \frac{1}{4z} - \frac{1}{z^2} + \frac{1}{8}$$

```
H_ap_den = expand((1/(2*z) - 1)*(1/(4*z) + 1))
```

$$H_{\text{ap_den}} =$$

$$\frac{1}{4z} + \frac{1}{8z^2} - 1$$

We can multiply both the numerator and denominator with 8, to get nice numbers:

```
(H_ap_num * 8) / (H_ap_den * 8)
```

ans =

$$\frac{\frac{2}{z} - \frac{8}{z^2} + 1}{\frac{2}{z} + \frac{1}{z^2} - 8}$$

Now, we can write the transfer function for the allpass filter:

$$H_{ap}(z) = \frac{1 + 2z^{-1} - 8z^{-2}}{-8 + 2z^{-1} + z^{-2}}$$

So we have:

$$\begin{aligned} H(z) &= H_{min}(z)H_{ap}(z) \\ &= (-8 + 2z^{-1} + z^{-2}) \left(\frac{1 + 2z^{-1} - 8z^{-2}}{-8 + 2z^{-1} + z^{-2}} \right) \end{aligned}$$

[✓] ADSI Problem 1.5 Filter decomposition

1) Show that a FIR filter with a difference equation is not minimum-phase

Consider a FIR filter with the following difference equation

$$y(n) = x(n) + 2x(n-1) + 2x(n-2)$$

1. Show that the FIR filter is not minimum-phase.

To determine whether the filter is a minimum-phase or not, we need get the transfer function of this FIR filter $H(z)$ which is defined as:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1z^{-1} + b_2z^{-2} \dots + b_Mz^{-M}}{1 + a_1z^{-1} + a_2z^{-2} \dots + a_Nz^{-N}}$$

This means that we must take the z-transform of difference equation:

$$Y(z) = X(z) + 2X(z)z^{-1} + 2X(z)z^{-2}$$

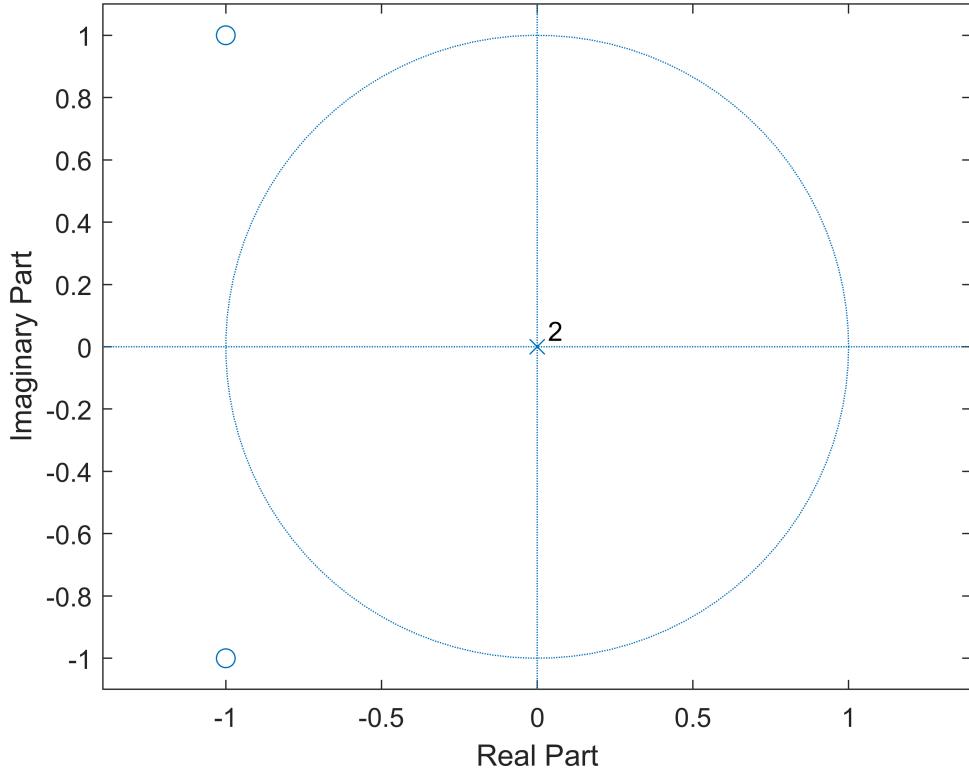
$$Y(z) = X(z)(1 + 2z^{-1} + 2z^{-2})$$

$$\frac{Y(z)}{X(z)} = 1 + 2z^{-1} + 2z^{-2}$$

$$H(z) = 1 + 2z^{-1} + 2z^{-2}$$

Once we have the transfer function, we can determine if the filter is a minimum-phase by looking at the zplane plot:

```
zplane([1, 2, 2])
```



Since all the zeros are outside the unit circle, clearly the given FIR filter is not minimum-phase. It is a **maximum-phase**.

2) Find the difference equation for the corresponding minimum-phase FIR filter

To decompose the system function, we take following steps:

1. Take the zeros of $H(z)$ that lie outside the unit circle and move them to the allpass system function $H_{ap}(z)$
2. Add poles to the allpass system in conjugate reciprocal locations of the zeros
3. To keep the equation balanced, we add zeros in the minimum-phase system function. This will even out the poles that we added to the allpass system function.

Since the poles that we added to the allpass system were all inside the unit circle, the zeros that we added to the minimum-phase system are also inside the unit circle.

This is coded in a MATLAB function.

We found that the transfer function for the difference equation is:

$$H(z) = 1 + 2z^{-1} + 2z^{-2}$$

```
h = [1, 2, 2];
[H_min, H_ap] = decompose_min_ap(h)
```

$H_{\text{min}} =$

$$\frac{2}{z} + \frac{1}{z^2} + 2$$

$H_{\text{ap}} =$

$$\frac{\left(\frac{1}{z} + \frac{1}{2} - \frac{1}{2}i\right) \left(\frac{1}{z} + \frac{1}{2} + \frac{1}{2}i\right)}{\left(1 + \frac{\frac{1}{2} - \frac{1}{2}i}{z}\right) \left(1 + \frac{\frac{1}{2} + \frac{1}{2}i}{z}\right)}$$

Let us simplify this expression:

```
H_ap_num = expand((1/z + 1/2 - 1/2i) * (1/z + 1/2 + 1/2i))
```

$H_{\text{ap_num}} =$

$$\frac{1}{z} + \frac{1}{z^2} + \frac{1}{2}$$

```
H_ap_den = expand((1+ (1/2 -1/2i)*z^-1) * (1 + (1/2+1/2i)*z^-1 ))
```

$H_{\text{ap_den}} =$

$$\frac{1}{z} + \frac{1}{2z^2} + 1$$

So the allpass system function is:

$$H_{\text{ap}}(z) = \frac{\frac{1}{2} + z^{-1} + z^{-2}}{1 + z^{-1} + \frac{1}{2}z^{-2}}$$

And the decomposition is:

$$H(z) = H_{\text{min}}(z)H_{\text{ap}}(z)$$

$$= (2 + 2z^{-1} + z^{-2}) \left(\frac{\frac{1}{2} + z^{-1} + z^{-2}}{1 + z^{-1} + \frac{1}{2}z^{-2}} \right)$$

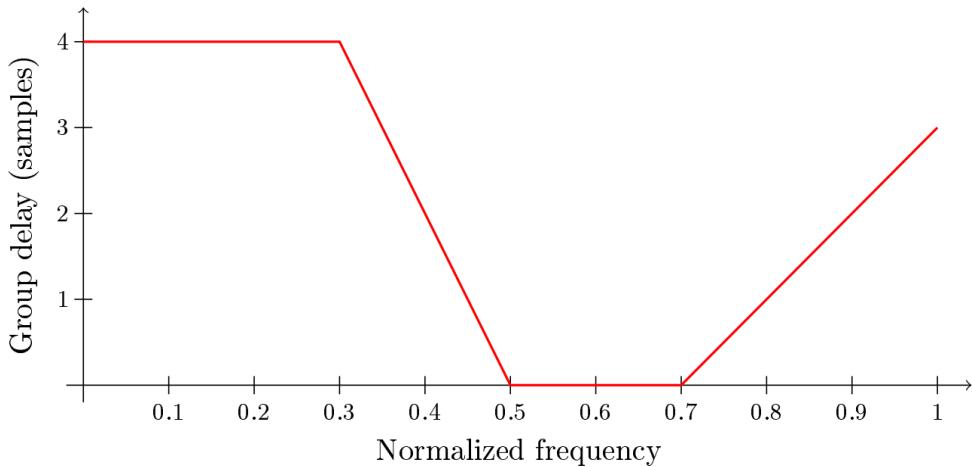
The difference equation for the minimum-phase system is:

$$y(n) = 2x(n) + 2x(n - 1) + x(n - 2)$$

ADSI Problem 1.8: Designing all-pass filters

```
clear variables;
```

Assume it is desired to create an all-pass filter with group delay versus frequency as shown in plot below. Due to the sharp edges in the group delay such a filter can hardly be designed but an approximation is feasible.



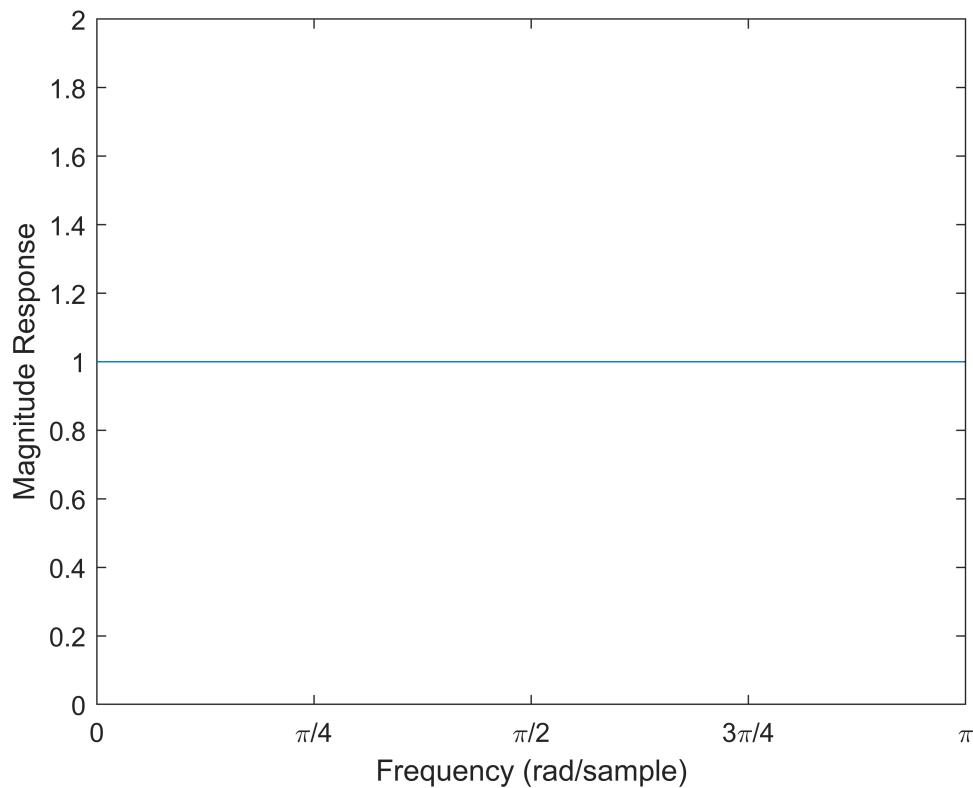
1) Use iirgrpdelay in MATLAB to design a 4th order all-pass filter.

You can set edges=[0 1]

```
n = 4; % all-pass filter order
f = [0, 0.3, 0.5, 0.7, 1];
a = [4, 4, 0, 0, 3];
edges = [0, 1];

[b_ap, a_app] = iirgrpdelay(n, f, edges, a);

[H, w] = freqz(b_ap, a_app);
plot(w, H.*conj(H));
set(gca, 'XTick', 0:pi/4:pi)
set(gca, 'XTickLabel', {'0', '\pi/4', '\pi/2', '3\pi/4', '\pi'})
xlabel('Frequency (rad/sample)')
ylabel('Magnitude Response')
xlim([0, pi]);
ylim([0, 2])
```



2) What happens as you increase the order of the filter?

Problem 2: Allpass filters

Let a system function be given by:

$$H(z) = \frac{a + bz^{-1} + cz^{-2}}{c + bz^{-1} + az^{-2}}$$

where a , b and c are non-zero numbers.

1) Show that the system function $H(z)$ is an allpass filter

An N -order allpass systems is given by

$$H_{\text{ap}}(z) = e^{j\beta} \prod_{k=1}^N \frac{z^{-1} - p_k^*}{1 - p_k z^{-1}}, \quad (5.158)$$

where β is a constant (usually $\beta = 0$).

A second-order allpass system function can be expressed as:

$$H_{\text{ap}}(z) = \frac{a_2^* + a_1^* z^{-1} + z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = z^{-2} \frac{1 + a_1^* z + a_2^* z^2}{1 + a_1 z^{-1} + a_2 z^{-2}}, \quad (5.164)$$

We can show that the given system function is an allpass filter if we can express it as in Eq. (5.164).

Since we want the denominator to contain a single 1 term, we scale both numerator and denominator by $\frac{1}{c}$:

$$H(z) = \frac{a + bz^{-1} + cz^{-2}}{c + bz^{-1} + az^{-2}}$$

$$H(z) = \frac{\frac{1}{c}(a + bz^{-1} + cz^{-2})}{\frac{1}{c}(c + bz^{-1} + az^{-2})}$$

$$H(z) = \frac{\frac{a}{c} + \frac{b}{c}z^{-1} + \frac{c}{c}z^{-2}}{\frac{c}{c} + \frac{b}{c}z^{-1} + \frac{a}{c}z^{-2}}$$

$$H(z) = \frac{\frac{a}{c} + \frac{b}{c}z^{-1} + z^{-2}}{1 + \frac{b}{c}z^{-1} + \frac{a}{c}z^{-2}}$$

2) Realise an allpass filter as an all-pole lattice filter

Allpass filters can be realized as all-pole lattice filters.

2. Find the two reflection coefficients as functions of the parameters, a, b, c and draw the corresponding lattice structure.

Based on Eq. 9.89 in the textbook we have that

$$A_2(z) = 1 + \frac{b}{c}z^{-1} + \frac{a}{c}z^{-2} \text{ and } B_2(z) = \frac{a}{c} + \frac{b}{c}z^{-1} + z^{-2}$$

From $A_2(z)$ the reflection coefficient $k_2 = \frac{a}{c}$ is directly evident. The other reflection coefficient is found using the step-down procedure 9.71

$$A_1(z) = \frac{1}{1 - k_2^2} (A_2(z) - k_2 B_2(z))$$

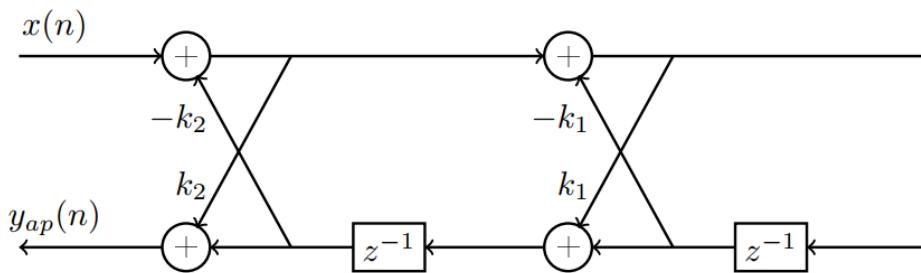
Therefore

$$\begin{aligned} A_1(z) &= \frac{1}{1 - \left(\frac{a}{c}\right)^2} \left(1 + \frac{b}{c}z^{-1} + \frac{a}{c}z^{-2} - \frac{a}{c} \left(\frac{a}{c} + \frac{b}{c}z^{-1} + z^{-2} \right) \right) \\ &= \frac{1}{1 - \left(\frac{a}{c}\right)^2} \left(1 - \left(\frac{a}{c}\right)^2 + \frac{b}{c} \left(1 - \frac{a}{c} \right) z^{-1} \right) \end{aligned}$$

From which it is found that

$$k_1 = \frac{\frac{b}{c} \left(1 - \frac{a}{c} \right)}{1 - \left(\frac{a}{c}\right)^2} = \frac{\frac{b}{c}}{1 + \frac{a}{c}}$$

The corresponding lattice structure is



Functions

```
function [H_min, H_ap]=decompose_min_ap(h)
% Decomposes a filter into minimum-phase filter and all-pass filter
% h: the filter coefficients
syms z;
rts = roots(h);

H_outside = 1; % Represents part of H where zeros are outside the unit circle
H_inside = 1; % Represents part of H where zeros are inside the unit circle
zeros_outside = [];
for i = 1:numel(rts)
    root = rts(i);
    if abs(root) > 1
        H_outside = H_outside * (1 - root*z^-1);
        zeros_outside = [zeros_outside; root];
    end
end
```

```

    else
        H_inside = H_inside * (1 - root*z^-1);
    end
end

% Sanity check
% H = expand(H_inside * H_outside)

% Compute minimum-phase filter and all-pass filter
H_min = 1;
N = numel(zeros_outside);
H_ap = 1;
for i = 1:N
    z_i = zeros_outside(i);
    a = 1/z_i;
    a_conj = conj(a);
    H_min = H_min * (-(1/a_conj) * H_inside * (1-a * z^(-1)));
    H_ap = H_ap * (z^(-1) - a_conj) / (1 - a*z^(-1));
end
H_min = expand(H_min);
end

```

ARMA(p, q) process

Table of Contents

ARMA(p, q) model.....	1
Autocorrelation of ARMA(p,q) process.....	2
Quiz: why can we choose to set $b_0=1$?.....	2
AR(p) process.....	2
Autocorrelation of AR(p) process.....	3
Computing AR(p) coefficients from data.....	3
MA(q) process.....	4
Problems.....	4
Estimate PSD from an ACRS assuming MA(1) process.....	4
Estimate PSD from an ACRS assuming MA(2) process.....	5
How to generate a realisation of an AR(3) process?.....	7
ADSI Problem 4.9: MA(q) processes.....	7
1) Full expressions of MA(0) - MA(3) processes.....	7
2) Calculate the autocorrelation for the MA processes.....	8
3) Calculate power spectral density.....	9
4) Plot the power density spectra given coefficients.....	10
ADSI Problem 4.10: MA(2) processes and phase properties.....	11
1) What is the order q of the processes.....	12
2) Compare the phase properties of the two systems.....	12
3) Compute and plot the power density spectra of the two processes.....	13
ADSI Problem 4.12: MA(q) spectral estimation from autocorrelation.....	15
1) Compute and plot the power density spectrum.....	15
2) Calculate the model parameters.....	16
Plot the power density spectrum an AR(3) process.....	18
Exam 2017 Problem 4: Prove the autocorrelation function for an MA(1) process.....	19
ADSI Problem 6.2: Autocorrelation expression for an AR(1) process (proof).....	20
Problems True/False.....	23
The autocorrelation function of an AR(p) is equal to zero for lags larger than p.....	23
Quiz: Is the comb filter an MA(1) process?.....	23
Quiz: Determine kind of model based on the impulse response.....	23
[✓] Exam 2012, Problem 2: Construct an AR(2) model using a signal from a random process.....	24
Exam 2016 Problem 1: AR(p) signal modelling.....	26
1) Compute AR(2) model coefficient given autocorrelation sequence.....	27
2) Is the random process better modelled as an AR(3) or higher-order process?.....	29
Exam 2018 Problem 2: Signal Modelling using MA(2) or AR(2).....	30
[✓] 1) Calculate and plot the autocorrelation function of the data.....	30
2) Can data be described by an MA(2) model?.....	31
3) Calculate the power spectral density of the random process assuming an AR(2) model.....	32
Problem 1: Signal modelling assuming AR(2) process.....	34
1) Compute AR(2) model coefficient given autocorrelation sequence.....	34
2) Is the random process better modelled as an AR(3) or higher-order process?.....	37
Functions.....	38

ARMA(p, q) model

An ARMA(p,q) process is defined by the difference equation:

$$y[n] = - \sum_{k=1}^p a_k y[n-k] + \sum_{k=0}^q b_k x[n-k], \quad (13.132)$$

feedback
autoregressive
feed forward
moving average

where $x[n] \sim \text{WN}(0, \sigma_x^2)$

Lesson: we always assume that the input to an ARMA(p,q) process is white noise.

Autocorrelation of ARMA(p,q) process

The autocorrelation of an ARMA(p,q) process is given by:

$$r_{yy}[\ell] = - \sum_{k=1}^p a_k r_{yy}[\ell-k] + \sigma_x^2 \sum_{k=0}^q b_k h[k-\ell], \quad \text{all } \ell. \quad (13.137)$$

However, we have an issue with this expression. The impulse response $h[n]$ is actually some function of the coefficients $\{a_k, b_k\}$. The issue is that this function is a non-linear. So solving the analytical expression becomes troublesome.

However, if we get rid of the second term by setting $q = 0$, the relation becomes linear and we have an AR(p) process:

$$r_{yy}[\ell] = - \sum_{k=1}^p a_k r_{yy}[\ell-k] + \sigma_x^2 b_0 h[-\ell]. \quad \text{all } \ell \quad (13.138)$$

We only consider positive lags $\ell > 0$ because $h(\ell) = 0$ for $\ell < 0$ i.e., the impulse response does not exist at negative times because it is a causal system.

$$r_{yy}[\ell] = - \sum_{k=1}^p a_k r_{yy}[\ell-k], \quad \ell > 0 \quad (13.140)$$

Quiz: why can we choose to set $b_0=1$?

Quiz: why can we choose to set $b_0 = 1$?

AR(p) process

AR(p) model is given by:

$$y(n) = - \sum_{k=1}^p [a_k y(n-k)] + b_0 x(n)$$

where the input is white noise with zero mean $x(n) \sim \text{WN}(0, \sigma_x^2)$.

Autocorrelation of AR(p) process

The autocorrelation of an ARMA(p,q) process is given by:

$$r_{yy}[\ell] = - \sum_{k=1}^p a_k r_{yy}[\ell - k] + \sigma_x^2 \sum_{k=0}^q b_k h[k - \ell], \text{ all } \ell. \quad (13.137)$$

However, we have an issue with this expression. The impulse response $h[n]$ is actually some function of the coefficients $\{a_k, b_k\}$. The issue is that this function is a non-linear. So solving the analytical expression becomes troublesome.

However, if we get rid of the second term by setting $q = 0$, the relation becomes linear and we have an AR(p) process:

$$r_{yy}[\ell] = - \sum_{k=1}^p a_k r_{yy}[\ell - k] + \sigma_x^2 b_0 h[-\ell]. \text{ all } \ell \quad (13.138)$$

We only consider positive lags $\ell > 0$ because $h(\ell) = 0$ for $\ell < 0$ i.e., the impulse response does not exist at negative times because it is a causal system.

$$r_{yy}[\ell] = - \sum_{k=1}^p a_k r_{yy}[\ell - k], \ell > 0 \quad (13.140)$$

This is useful because we can use it to find the coefficients $\{a_k\}$ of an AR(q) model. Given an output signal $y(n)$, we can compute the autocorrelation $r_{yy}(\ell)$ numerically in MATLAB. This means that the expression in Eq. 13.141 becomes a set of p linear equations.

Computing AR(p) coefficients from data

For example, if we want to model second-order AR model, $p = 2$ we get two equations with two unknowns:

$$r_{yy}(1) = -a_1 r_{yy}(0) - a_2 r_{yy}(-1)$$

$$r_{yy}(2) = -a_1 r_{yy}(1) - a_2 r_{yy}(0)$$

We can write it into matrix form:

$$\begin{bmatrix} r_{yy}(0) & r_{yy}(1) \\ r_{yy}(1) & r_{yy}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = - \begin{bmatrix} r_{yy}(1) \\ r_{yy}(2) \end{bmatrix}$$

In the general case, it becomes:

$$\begin{bmatrix} r_{yy}[0] & r_{yy}[1] & \dots & r_{yy}[p-1] \\ r_{yy}[1] & r_{yy}[0] & \dots & r_{yy}[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_{yy}[p-1] & r_{yy}[p-2] & \dots & r_{yy}[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} r_{yy}[1] \\ r_{yy}[2] \\ \vdots \\ r_{yy}[p] \end{bmatrix},$$

More compactly written as *Yule–Walker equations*:

$$\mathbf{R}_y \mathbf{a} = -\mathbf{r}_y. \quad (13.143)$$

The input noise variance can be compute as follows:

$$\sigma_x^2 = \sigma_y^2 + \mathbf{a}^T \mathbf{r}_y = \sigma_y^2 - \mathbf{r}_y^T \mathbf{R}_y^{-1} \mathbf{r}_y \leq \sigma_y^2. \quad (13.144)$$

We can write a general MATLAB function to find the coefficients $\{a_k\}$ for any AR(p) model arfit.

MA(q) process

MA(q) process is given by:

$$y(n) = \sum_{k=0}^q b_k x(n-k)$$

This is the ARMA(p, q) model where the feedback part is excluded i.e., all values of a_k is set to zero.

Problems

Estimate PSD from an ACRS assuming MA(1) process

The autocorrelation function for a MA(1) process has been found to be

$ l $	$r_{yy}(l)$
0	2
1	1
$ l \geq 2$	0

```
clear variables;
```

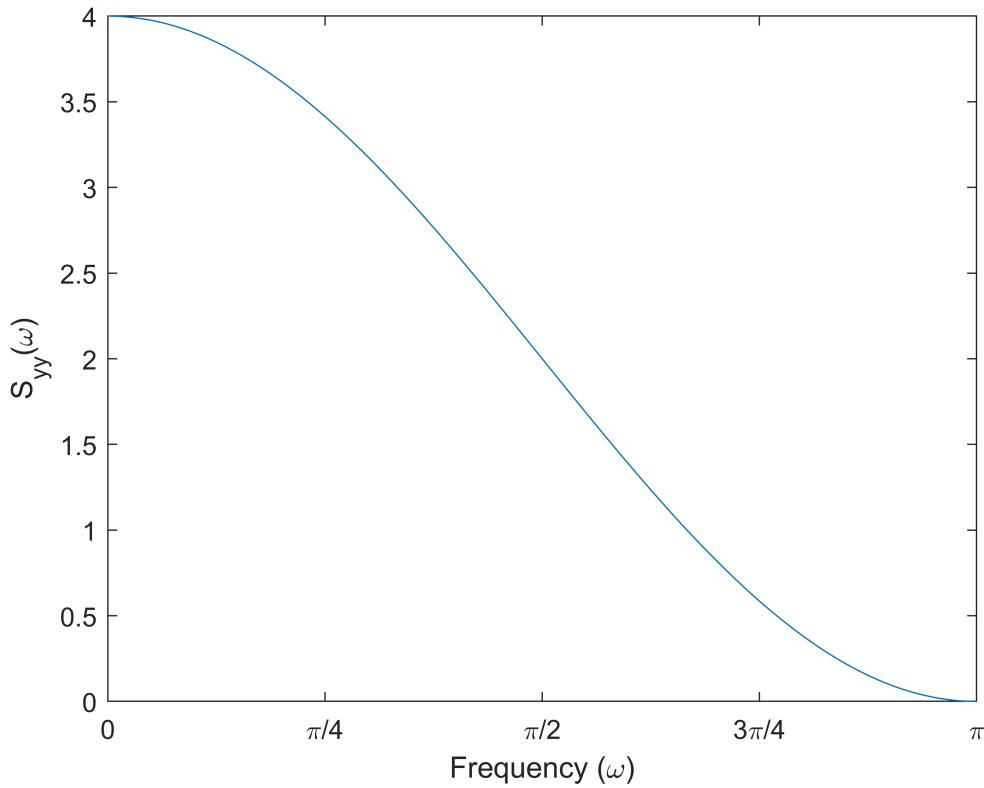
The PSD for an MA(q) process can be computed using Eq. (13.119):

$$S_{xx}(\omega) = r_{xx}[0] + 2 \sum_{\ell=1}^{\infty} r_{xx}[\ell] \cos \omega \ell, \quad (13.119)$$

Plugging in our values, we get:

$$S_{yy}(\omega) = 2 + 2(1 \cos(1\omega)) = 2 + 2\cos(\omega)$$

```
w = 0:0.001:2*pi;
r_xx = [2, 1, 0];
S = r_xx(1);
for l = 2:numel(r_xx)
    S = S + 2 * r_xx(l)*cos(w*(l-1));
end
plot(w, S);
xlabel('Frequency (\omega)');
ylabel('S_{yy}(\omega)');
set(gca,'XTick',0:pi/4:pi)
set(gca,'XTickLabel',{'0', '\pi/4', '\pi/2', '3\pi/4', '\pi'})
xlim([0, pi])
```



Estimate PSD from an ACRS assuming MA(2) process

For a given random process $\{x(n)\}$ the autocorrelation has been estimated and is given by:

$ m $	$r_x(m)$
0	4
1	2
2	-1

Estimate the power density spectrum under the assumption that $\{x(n)\}$ can be described as a $MA(2)$ process.

```
clear variables;
```

The PSD for an MA(q) process can be computed using Eq. (13.119):

$$S_{xx}(\omega) = r_{xx}[0] + 2 \sum_{\ell=1}^{\infty} r_{xx}[\ell] \cos \omega \ell, \quad (13.119)$$

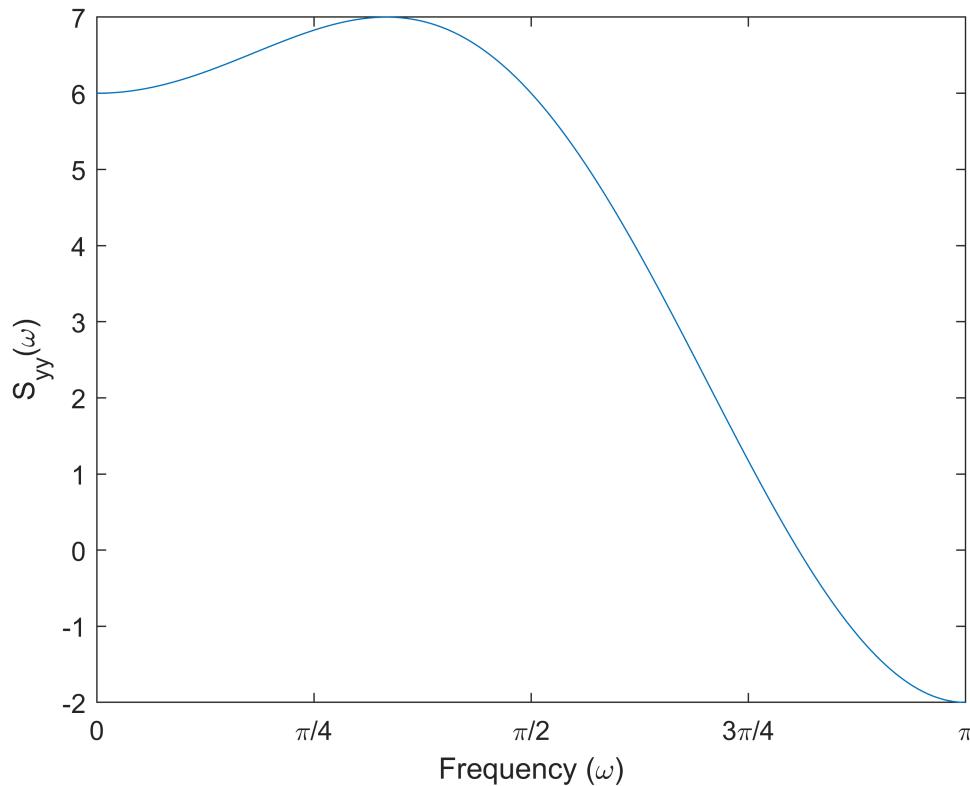
This gives following expression:

$$\begin{aligned} S_{xx}(\omega) &= 4 + 2(2\cos(\omega) + (-1)\cos(2\omega)) \\ &= 4 + 4\cos(\omega) - 2\cos(2\omega) \end{aligned}$$

```
w=0:0.001:pi;
r_xx = [4, 2, -1];

S = r_xx(1);
for l = 2:numel(r_xx)
    S = S + 2 * r_xx(l)*cos(w*(l-1));
end

plot(w, S);
xlabel('Frequency (\omega)');
ylabel('S_{yy}(\omega)');
set(gca,'XTick',0:pi/4:pi)
set(gca,'XTickLabel',{'0', '\pi/4', '\pi/2', '3\pi/4', '\pi'})
xlim([0, pi])
```



How to generate a realisation of an AR(3) process?

The signal is disturbed by additive noise from an AR(3) process given by

$$v(n) = -0.5v(n-1) - 0.5v(n-2) - 0.25v(n-3) + w(n)$$

where $w(n) \sim WGN(0, 4)$.

```
% Generate white noise w[n] with zero mean and variance 4
w_var = 4;
w = sqrt(w_var) * randn(10000, 1);

% Generate noise from an AR(3) process
b = 1;
a = [1, 0.5, 0.5, 0.25];
v = filter(b, a, w);
```

ADSI Problem 4.9: MA(q) processes

In this problem we will investigate the MA(q) process as defined by Eq. (13.132) by excluding the feedback part

$$y[n] = \sum_{k=0}^q b_k x[n-k]$$

where the input is a zero-mean Gaussian white noise process with unit variance.

1) Full expressions of MA(0) - MA(3) processes

Write out the full expressions for MA(0), MA(1), MA(2) and MA(3) processes.

The general ARMA(p, q) is given by the difference equation:

$$y[n] = - \sum_{k=1}^p a_k y[n-k] + \sum_{k=0}^q b_k x[n-k], \quad (13.132)$$

When the feedback part is excluded, all values of a_k is set to zero, we are left with:

$$y[n] = \sum_{k=0}^q b_k x[n-k]$$

We can write out the full expressions for the difference processes as follows:

$$\text{MA}(0) \rightarrow y[n] = b_0 x[n]$$

$$\text{MA}(1) \rightarrow y[n] = b_0 x[n] + b_1 x[n-1]$$

$$\text{MA}(2) \rightarrow y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2]$$

$$\text{MA}(3) \rightarrow y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + b_3 x[n-3]$$

2) Calculate the autocorrelation for the MA processes

Calculate the autocorrelation $r_{yy}[l]$ for the MA(0) to MA(3) processes.

The autocorrelation function of a random process is defined as:

$$r_{yy}(\ell) = E[y(n)y(n-\ell)]$$

To compute the autocorrelation for the MA(0), just plug its difference equation into this equation:

$$r_{yy}(\ell) = E[b_0 x(n) \cdot b_0 x(n-\ell)]$$

$$r_{yy}(\ell) = b_0^2 E[x(n) \cdot x(n-\ell)]$$

$$r_{yy}(\ell) = b_0^2 r_{xx}(\ell)$$

The autocorrelation of a white noise signal $w(n)$ is: $r_{ww}(\ell) = \sigma_w^2 \delta(\ell)$ where σ_w^2 is the signal's variance (see week 6 notes)

We are given that the input signal $x(n)$ is Gaussian white noise with unit variance. This means $r_{xx}(\ell) = \delta(\ell)$

Therefore, the autocorrelation for MA(0) process is:

$$r_{yy}(\ell) = b_0^2 \delta(\ell)$$

The autocorrelation for the MA(1) process is:

$$r_{yy}(\ell) = E[(b_0 x[n] + b_1 x[n-1]) \cdot (b_0 x[n-\ell] + b_1 x[n-\ell-1])]$$

Multiply the two factors:

$$r_{yy}(\ell) = E[b_0^2 x[n]x[n-\ell] + b_0 b_1 x[n]x[n-\ell-1] + b_0 b_1 x[n-1]x[n-\ell] + b_1^2 x[n-1]x[n-\ell-1]]$$

Rearrange the coefficients:

$$r_{yy}(\ell) = E[b_0^2 x[n]x[n-\ell] + b_0 b_1 (x[n]x[n-\ell-1] + x[n-1]x[n-\ell]) + b_1^2 x[n-1]x[n-\ell-1]]$$

Split the expectation values and move constants outside $E[\cdot]$

$$r_{yy}(\ell) = b_0^2 E[x[n]x[n-\ell]] + b_0 b_1 E[(x[n]x[n-\ell-1] + x[n-1]x[n-\ell])] + b_1^2 E[x[n-1]x[n-\ell-1]]$$

Split the expectation value in the second term:

$$r_{yy}(\ell) = b_0^2 E[x[n]x[n-\ell]] + b_0 b_1 (E[x[n]x[n-\ell-1]] + E[x[n-1]x[n-\ell]]) + b_1^2 E[x[n-1]x[n-\ell-1]]$$

Replace expectation values with autocorrelation functions:

$$r_{yy}(\ell) = b_0^2 r_{xx}(\ell) + b_0 b_1 (r_{xx}(\ell - 1) + r_{xx}(\ell + 1)) + b_1^2 r_{xx}(\ell)$$

Rearrange the terms

$$r_{yy}(\ell) = (b_0^2 + b_1^2) r_{xx}(\ell) + b_0 b_1 (r_{xx}(\ell - 1) + r_{xx}(\ell + 1))$$

As before $r_{xx}(\ell) = \delta(\ell)$ because the input signal is a white noise with unit variance. Substitute:

$$r_{yy}(\ell) = (b_0^2 + b_1^2) \delta(\ell) + b_0 b_1 (\delta(\ell - 1) + \delta(\ell + 1))$$

The autocorrelation for an MA(2) process is:

$$r_{yy}(\ell) = (b_0^2 + b_1^2 + b_2^2) \delta(\ell) + (b_0 b_1 + b_1 b_2) \delta(\ell + 1) + (b_0 b_1 + b_1 b_2) \delta(\ell - 1) + b_0 b_2 \delta(\ell + 2) + b_0 b_2 \delta(\ell - 2)$$

The autocorrelation for an MA(3) process is:

$$\begin{aligned} r_{yy}(l) &= (b_0^2 + b_1^2 + b_2^2 + b_3^2) \delta(l) + (b_0 b_1 + b_1 b_2 + b_2 b_3) \delta(l + 1) + (b_0 b_1 + b_1 b_2 + b_2 b_3) \delta(l - 1) \\ &\quad + (b_0 b_2 + b_1 b_3) \delta(l + 2) + (b_0 b_2 + b_1 b_3) \delta(l - 2) + b_0 b_3 \delta(l + 3) + b_0 b_3 \delta(l - 3) \end{aligned}$$

3) Calculate power spectral density

Calculate the general expressions for power density spectra of MA(0), MA(1) and MA(2) processes using Eq. (13.112).

The power spectral density (PSD) is defined as:

$$S_{xx}(\omega) \triangleq \sum_{\ell=-\infty}^{\infty} r_{xx}[\ell] e^{-j\ell\omega}. \quad (13.112)$$

In 2) we found that the autocorrelation for MA(0) is $r_{yy}(\ell) = b_0^2 \delta(\ell)$, so plug it in this PSD formula:

$$S_{yy}^{MA(0)}(\omega) = \sum_{\ell=-\infty}^{\infty} r_{yy}(\ell) e^{-j\ell\omega} = \sum_{\ell=-\infty}^{\infty} b_0^2 \delta(\ell) e^{-j\ell\omega}$$

We notice that $r_{yy}(\ell)$ has non-zero value only when $\ell = 0$. Therefore, the terms for when $\ell \neq 0$ are zero. We can, therefore, remove the infinite sum. Since $\delta(0) = 1$ and $e^{-j0\omega} = 1$, we have:

$$S_{yy}^{MA(0)}(\omega) = b_0^2 \delta(0) e^{-j0\omega} = b_0^2 \cdot 1 \cdot 1 = b_0^2$$

We do similar computation to find the power density spectrum of MA(1) process.

In 2) we found that the autocorrelation for MA(1) is:

$$r_{yy}(\ell) = (b_0^2 + b_1^2)\delta(\ell) + b_0b_1 (\delta(\ell - 1) + \delta(\ell + 1))$$

There are three non-zero values of the autocorrelation function so the infinite sum has only three terms

$$S_{yy}^{MA(1)}(\omega) = \sum_{\ell=-\infty}^{\infty} ((b_0^2 + b_1^2)\delta(\ell) + b_0b_1 (\delta(\ell - 1) + \delta(\ell + 1)))e^{-j\ell\omega}$$

There are three non-zero values of the autocorrelation function so the infinite sum has only three terms:

- When $\ell = 0$: $(b_0^2 + b_1^2)\delta(0)e^{-j(0)\omega} = (b_0^2 + b_1^2) \cdot 1 \cdot 1 = (b_0^2 + b_1^2)$
- When $\ell = -1$: $b_0b_1 \delta(0)e^{-j(-1)\omega} = b_0b_1 (1)e^{j\omega} = b_0b_1 e^{j\omega}$
- When $\ell = 1$: $b_0b_1 \delta(0)e^{-j(1)\omega} = b_0b_1 (1)e^{-j\omega} = b_0b_1 e^{-j\omega}$

Therefore, we have:

$$S_{yy}^{MA(1)}(\omega) = (b_0^2 + b_1^2) + b_0b_1 e^{j\omega} + b_0b_1 e^{-j\omega}$$

$$S_{yy}^{MA(1)}(\omega) = (b_0^2 + b_1^2) + b_0b_1(e^{j\omega} + e^{-j\omega})$$

Since $\cos(\theta) = \frac{1}{2}(e^{j\theta} + e^{-j\theta})$ and $2 \cdot \cos(\theta) = e^{j\theta} + e^{-j\theta}$, we write:

$$S_{yy}^{MA(1)}(\omega) = (b_0^2 + b_1^2) + 2b_0b_1\cos(\omega)$$

In similar fashion, the PSD of MA(2) is:

$$S_{yy}^{MA(2)}(\omega) = (b_0^2 + b_1^2 + b_2^2) + 2(b_0b_1 + b_1b_2)\cos(\omega) + 2b_0b_1\cos(2\omega)$$

4) Plot the power density spectra given coefficients

Plot the power density spectra for the two processes defined by

q	b_0	b_1	b_2
1	3	2	
2	3	2	1

We are given the coefficients for $S_{yy}^{MA(1)}(\omega)$ and $S_{yy}^{MA(2)}(\omega)$. We use results from 3) to plot the power density spectra for the two processes.

$$S_{yy}^{MA(1)}(\omega) = (b_0^2 + b_1^2) + 2b_0b_1\cos(\omega)$$

$$S_{yy}^{MA(2)}(\omega) = (b_0^2 + b_1^2 + b_2^2) + 2(b_0b_1 + b_1b_2)\cos(\omega) + 2b_0b_1\cos(2\omega)$$

```
w=0:0.001:2*pi;
```

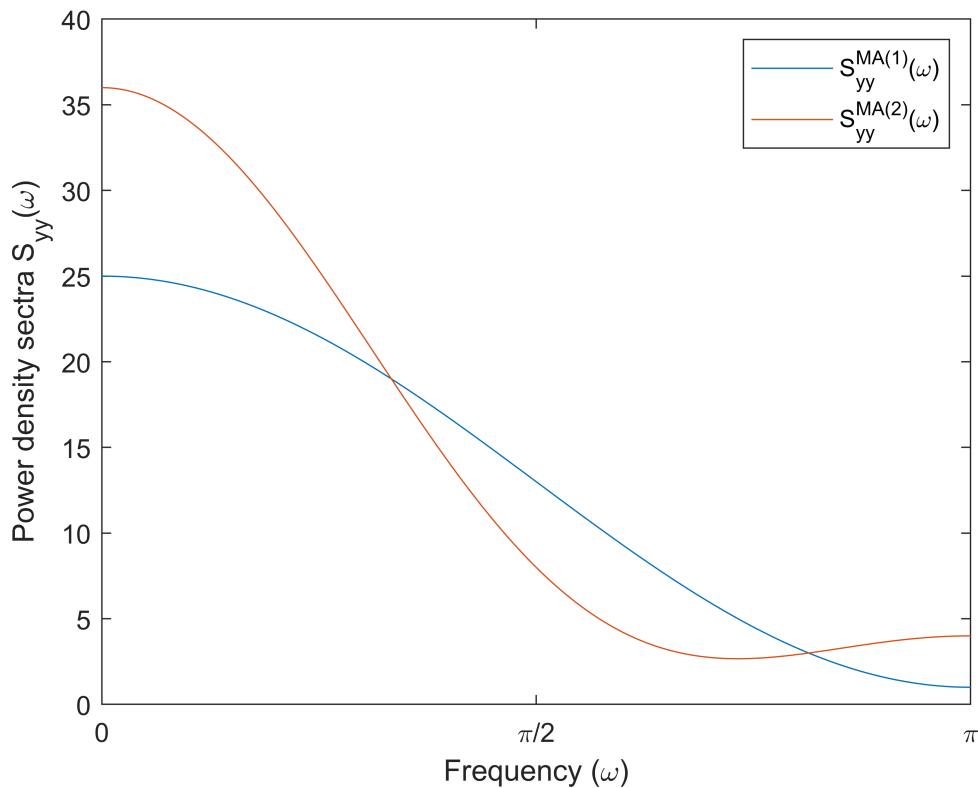
```

% Define the coefficients
b0 = 3; b1 = 2; b2 = 1;

% The power density spectrum for the MA(1) process
S1 = (b0^2 + b1^2) + 2*b0*b1*cos(w);
% The power density spectrum for the MA(2) process
S2 = (b0^2 + b1^2 + b2^2) + 2*(b0*b1 + b1*b2)*cos(w) + 2*b0*b2*cos(2*w);

plot(w, S1, w, S2);
legend('S_{yy}^{MA(1)}(\omega)', 'S_{yy}^{MA(2)}(\omega)');
xlabel('Frequency (\omega)');
ylabel('Power density spectra S_{yy}(\omega)');
set(gca, 'XTick', 0:pi/2:2*pi)
set(gca, 'XTickLabel', {'0', '\pi/2', '\pi', '3\pi/2', '2\pi'})
xlim([0,pi])

```



ADSI Problem 4.10: MA(2) processes and phase properties

Consider the following two MA(q) systems and assume that they are excited by zero-mean white Gaussian noise with unit variance.

$$y_1[n] = 2x[n] - x[n-2] \quad \text{and} \quad y_2[n] = x[n] - 2x[n-2]$$

1) What is the order q of the processes

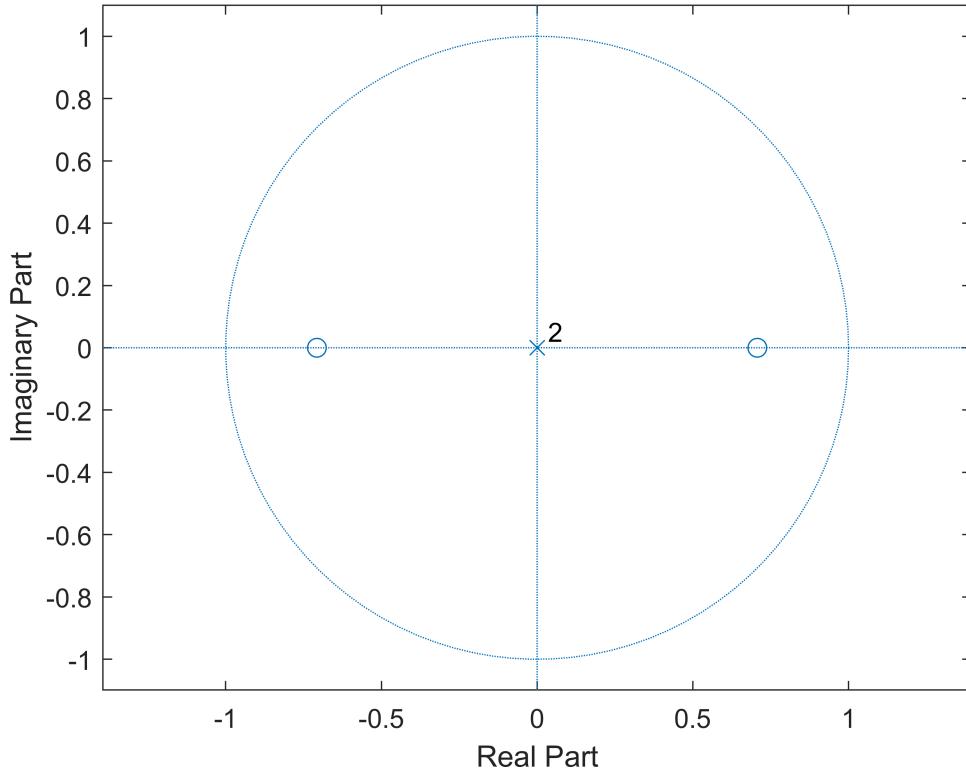
The order q of an $\text{MA}(q)$ process is given by the longest delay, i.e., $x[n - q]$. Therefore, both systems are of order 2.

2) Compare the phase properties of the two systems

One important phase property of a system is whether it is invertible. For a $\text{MA}(q)$ system, this implies that all zeros should be inside the unit circle. If all zeros are located within the unit circle, then we have a minimum-phase system.

We can use zero-pole plot in MATLAB:

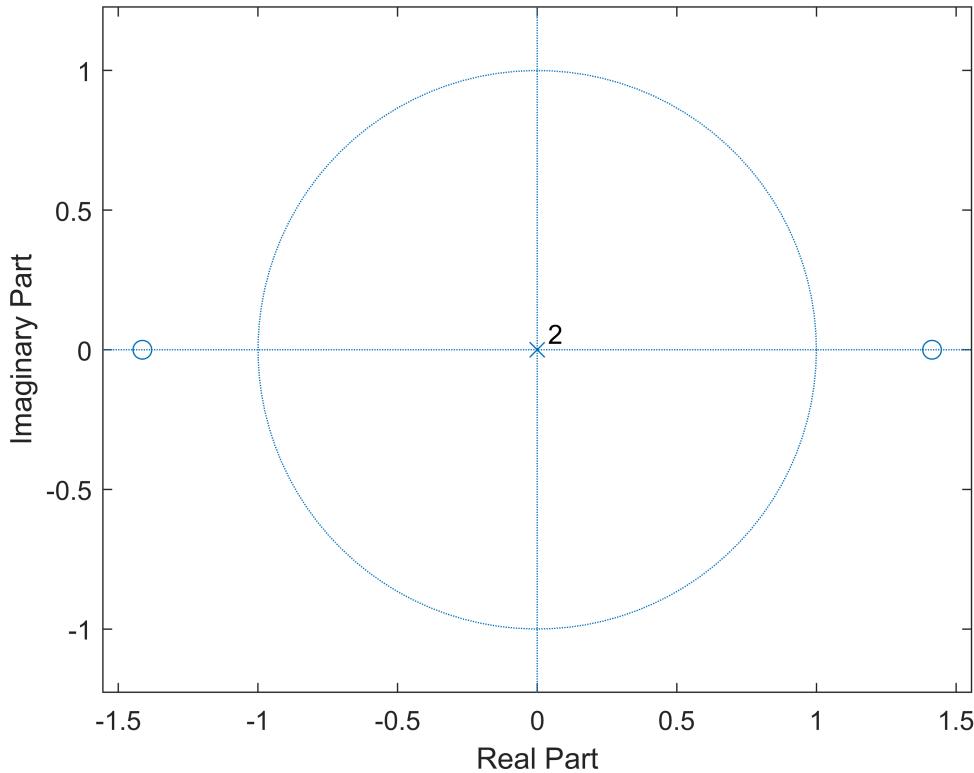
```
h1 = [2, 0, -1]; % Impulse response for y1[n]
zplane(h1);
```



From the zero-pole plot, we observe that all the zeros are within the unit circle. Therefore, we can conclude that the system $y_1[n]$ is a minimum-phase.

Let us make the zero-plot for the system $y_2[n] = x[n] - 2x[n - 2]$

```
h2 = [1, 0, -2]; % Impulse response for y2[n]
zplane(h2);
```



Since both zeros are outside the unit circle, we can conclude the system $y_2[n]$ is a maximum-phase system.

3) Compute and plot the power density spectra of the two processes

We are given two MA(q) systems:

$$y_1(n) = 2x(n) - x(n-2)$$

$$y_2(n) = x(n) - 2x(n-2)$$

where the input is $x(n) \sim \text{WGN}(0, 1)$ i.e., white Gaussian noise with unit variance.

We are asked to compute the power density spectra of both systems.

Before we can compute the power density spectrum, we need to find the autocorrelation.

Since we know impulse response, we can use Eq. (13.104) to compute the autocorrelation:

$$r_{yy}[\ell] = r_{hh}[\ell] * r_{xx}[\ell], \quad (13.104)$$

where

$$r_{hh}[m] = h[-m] * h[m], \quad (13.103)$$

The autocorrelation of white noise is $\sigma_x^2 \delta(\ell)$ so the ACRS of the input signal:

$$r_{xx}(\ell) = \sigma_x^2 \delta(\ell) = 1 \cdot \delta(\ell)$$

Since $r_{xx}(\ell) = \delta(\ell)$ then Eq. (13.104) becomes:

$$r_{yy}(\ell) = r_{hh}(\ell)$$

This means that if we compute $r_{hh}(\ell)$ then we have the autocorrelation.

We can use MATLAB to compute $r_{h1}(\ell)$ and $r_{h2}(\ell)$:

```
r_h1 = conv(fliplr(h1), h1)
```

```
r_h1 = 1x5
-2      0      5      0      -2
```

```
r_h2 = conv(fliplr(h2), h2)
```

```
r_h2 = 1x5
-2      0      5      0      -2
```

Now, we have the autocorrelation of both systems.

$ \ell $	r_{y1}	r_{y2}
0	5	5
1	0	0
2	-2	-2
≥ 3	0	0

We observe that they are the same. This means that the power spectrum density will be the same.

Whenever we need to compute the PSD based on an autocorrelation sequence, we use Eq. (13.119):

$$S_{xx}(\omega) = r_{xx}[0] + 2 \sum_{\ell=1}^{\infty} r_{xx}[\ell] \cos \omega \ell, \quad (13.119)$$

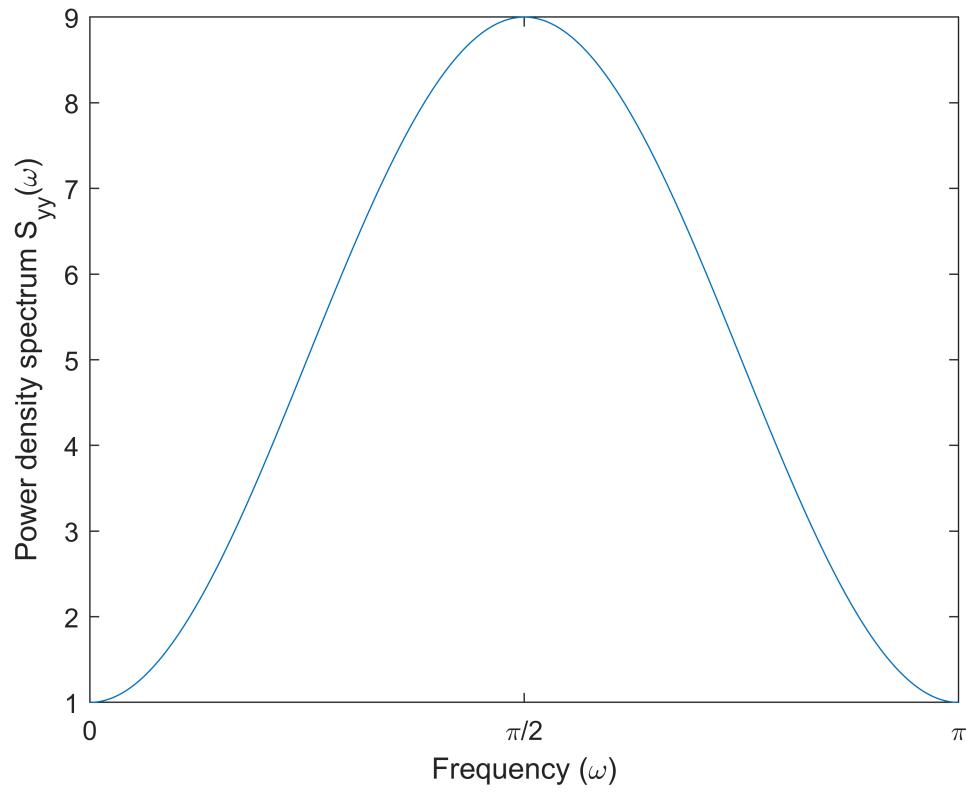
Plugging in our values, we get:

$$S_{yy}(\omega) = 5 + 2(0 + (-2)\cos(2\omega))$$

$$S_{yy}(\omega) = 5 - 4\cos(2\omega)$$

We can plot it in MATLAB:

```
w=0:0.001:2*pi;
S_yy = 5 - 4*cos(2*w);
plot(w, S_yy);
xlabel('Frequency (\omega)');
ylabel('Power density spectrum S_{yy}(\omega)');
set(gca,'XTick',0:pi/2:2*pi)
set(gca,'XTickLabel',{'0','\pi/2','\pi','3\pi/2','2\pi'})
xlim([0, pi])
```



ADSI Problem 4.12: MA(q) spectral estimation from autocorrelation

The autocorrelation function for a MA(1) process has been found to be

$ l $	$r_{yy}(l)$
0	2
1	1
$ l \geq 2$	0

```
clear variables;
```

1) Compute and plot the power density spectrum

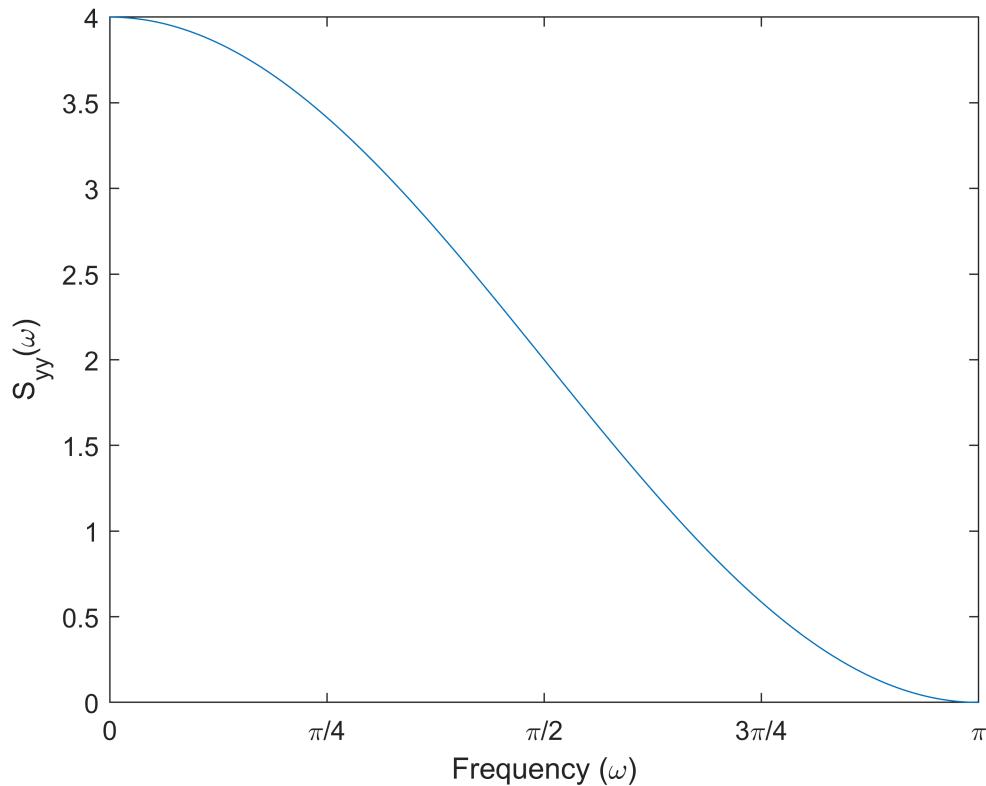
Whenever we need to compute the PSD based on an autocorrelation sequence, we use Eq. (13.119):

$$S_{xx}(\omega) = r_{xx}[0] + 2 \sum_{\ell=1}^{\infty} r_{xx}[\ell] \cos \omega \ell, \quad (13.119)$$

Plugging in our values, we get:

$$S_{yy}(\omega) = 2 + 2(1\cos(1\omega)) = 2 + 2\cos(\omega)$$

```
w=0:0.001:2*pi;
S_yy = 2 + 2*cos(w);
plot(w, S_yy);
xlabel('Frequency (\omega)');
ylabel('S_{yy}(\omega)');
set(gca,'XTick',0:pi/4:pi)
set(gca,'XTickLabel',{'0','\pi/4','\pi/2','3\pi/4','\pi'})
xlim([0, pi])
```



2) Calculate the model parameters

We are asked to find b_0 and b_1 for an MA(1) process given its autocorrelation sequence.

In 1) we found that:

$$S_{yy}(\omega) = 2 + 2\cos(\omega)$$

In ADSI Problem 4.9, we found that:

$$S_{yy}^{\text{MA}(1)}(\omega) = (b_0^2 + b_1^2) + 2b_0b_1\cos(\omega)$$

This implies that:

$$b_0^2 + b_1^2 = 2 \quad \text{and} \quad b_0b_1 = 1$$

Rewriting the second equation:

$$b_0 = \frac{1}{b_1}$$

We can substitute in the first equation:

$$\frac{1}{b_1^2} + b_1^2 = 2$$

From the equation above, we see that $b_1 = \pm 1$. We turn our attention the second equation:

$$b_0 = 1 \text{ if } b_1 = 1$$

$$b_0 = -1 \text{ if } b_1 = -1$$

So the model parameters are $b_0 = 1, b_1 = 1$ or $b_0 = -1, b_1 = -1$.

```
% Verify in MATLAB
syms b0 b1
solution = solve([b0^2 + b1^2 == 2, b0*b1 == 1], [b0, b1]);
solution.b0
```

ans =

$$\begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

```
solution.b1
```

ans =

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

Plot the power density spectrum an AR(3) process

```
clear variables;
```

Here is an AR(3) process plus some white Gaussian noise:

$$v(n) = -0.5v(n-1) - 0.5v(n-2) - 0.25v(n-3) + w(n)$$

where $w(n) \sim WGN(0, 4)$.

We can determine the Power Density Spectrum of an ARMA(p,q) process is given by

$$S_{yy}(\omega) = \sigma_x^2 |H(e^{j\omega})|^2 = \sigma_x^2 \left| \frac{\sum_{k=0}^q b_k e^{-j\omega k}}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2. \quad (13.133)$$

The power spectrum of an AR(p) process is given by:

$$S_{yy}(\omega) = \sigma_x^2 \left| \frac{1}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2$$

For this problem, we have:

$$S_{vv}(\omega) = 4 \left| \frac{1}{1 + 0.5e^{-j\omega} + 0.5e^{-j2\omega} + 0.25e^{-j3\omega}} \right|^2$$

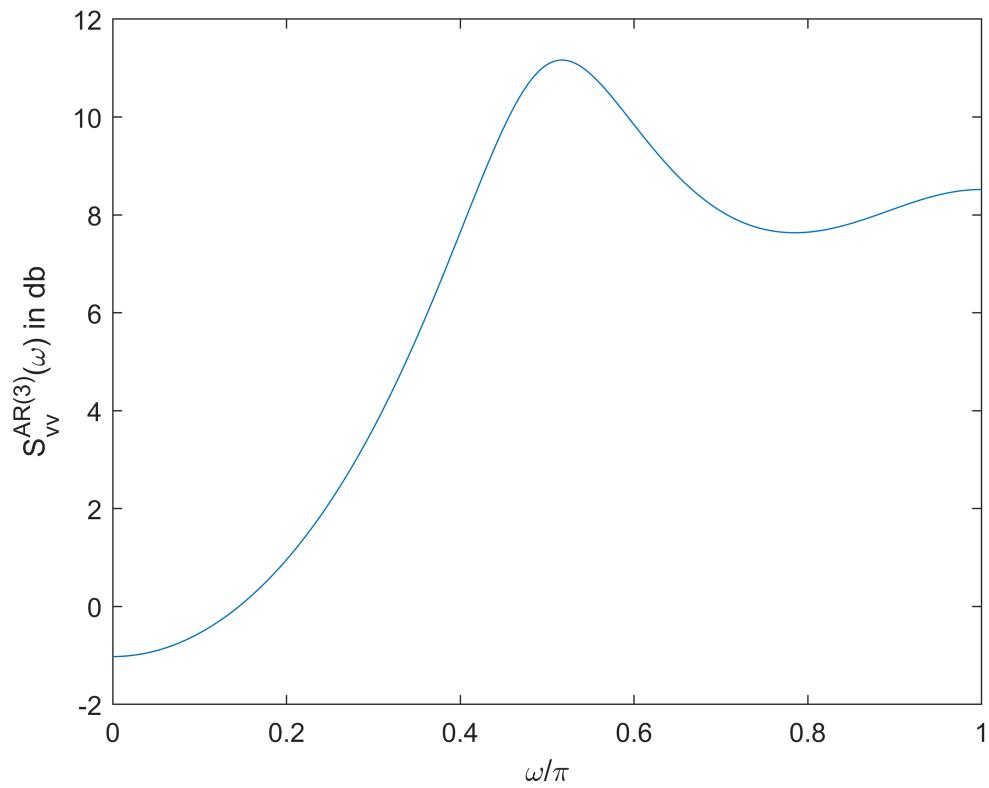
The algorithm is as follows:

1. Use the coefficents $\{a_1, a_2, \dots, a_p\}$ for the AR(p) model,
2. Compute the transfer function for the AR(p) by computing the sum and finding its reciprocal
3. Compute the conjugate of the transfer function: $|H(e^{j\omega})|^2$
4. Multiply it with the variance σ_x^2

The algorithm is implemented in the functions ar2psd() function:

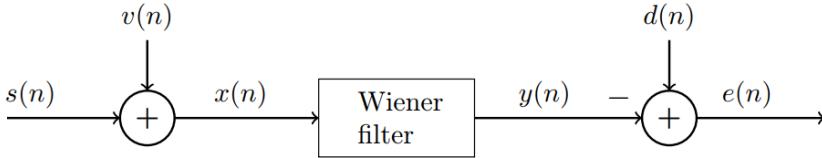
```
N = 256;
a = [0.5, 0.5, 0.25]; % The coefficients of the AR(3) model
w_var = 4; % The variance of white noise
[S_vv, w] = ar2psd(a, w_var, N); % Compute the PSD of AR(3) model

plot(w/pi, pow2db(S_vv))
xlabel('omega/pi')
ylabel('S_{vv}^{AR(3)}(omega) in db')
```



Exam 2017 Problem 4: Prove the autocorrelation function for an MA(1) process

Consider the Wiener filtering problem shown below. A signal $s(n)$ is corrupted by additive, uncorrelated noise $v(n)$ giving the signal $x(n) = s(n) + v(n)$. It is desired to use a 2-tap Wiener filter to recover the signal i.e. $d(n) = s(n)$.



The noise $v(n)$ is generated in an MA(1) process as

$$v(n) = w(n) + w(n - 1)$$

where $w(n)$ is a zero-mean gaussian white noise sequence with $\sigma_w^2 = 1$. The autocorrelation function of the signal is

$$r_s(l) = 4 \cdot 0.25^{|l|}.$$

```
clear variables;
```

1. Show that the autocorrelation function of the noise $v(n)$ is given by

$$r_v(l) = \delta(l - 1) + 2\delta(l) + \delta(l + 1).$$

In ADSI Problem 4.9, we found that the autocorrelation for an MA(1) process where the input signal is a white noise with unit variance can be described as:

$$r_v(\ell) = (b_0^2 + b_1^2)\delta(\ell) + b_0 b_1 (\delta(\ell - 1) + \delta(\ell + 1))$$

In this problem $b_0 = b_1 = 1$:

$$r_v(\ell) = \delta(\ell - 1) + 2\delta(\ell) + \delta(\ell + 1)$$

```
r_vv = [2, 1, 0]
```

```
r_vv = 1x3
      2     1     0
```

ADSI Problem 6.2: Autocorrelation expression for an AR(1) process (proof)

Consider an AR(1) process given by $y(n) = -ay(n-1) + x(n)$

with $-1 < a < 1$ and $x(n) \sim WN(0, \sigma_x^2)$.

1. Show that the autocorrelation of the AR(1) process is given by

$$r_{yy}(l) = \frac{\sigma_x^2}{1-a^2}(-a)^{|l|}$$

Hint: Use equation (13.138) and (13.140).

The autocorrelation of an AR(p) process is given by Eq. 13.138:

$$r_{yy}[\ell] = -\sum_{k=1}^p a_k r_{yy}[\ell-k] + \sigma_x^2 b_0 h[-\ell]. \quad \text{all } \ell \quad (13.138)$$

where $h[n]$ is the impulse response of an all-pole system.

The impulse response of an all-pole system satisfies the difference equation

$$h[n] = -\sum_{k=1}^p a_k h[n-k] + b_0 \delta[n], \quad n \geq 0 \quad (13.139)$$

where $h[n] = 0$ for $n < 0$

Equation 13.138 is a general expression for all ℓ . However, since $h[\ell] = 0$ for negative values of ℓ then we know that $h[-\ell] = 0$. Therefore, Eq. 13.138 can be reduced to:

$$r_{yy}[\ell] = -\sum_{k=1}^p a_k r_{yy}[\ell-k], \quad \ell > 0 \quad (13.140)$$

For an AR(1) process, Eq. 13.138 simplifies to:

$$r_{yy}[\ell] = -a_1 r_{yy}[\ell-1] + \sigma_x^2 b_0 h[-\ell] \quad \text{for all } \ell$$

For an AR(1) process, Eq. 13.140 simplifies to:

$$r_{yy}[\ell] = -a_1 r_{yy}[\ell-1] \quad \text{for } \ell > 0$$

Setting $\ell = 0$ in the first equation, we get:

$$r_{yy}[0] = -a_1 r_{yy}[-1] + \sigma_x^2 b_0 h[0]$$

The book says that $h[0] = b_0 = 1$ so we are left with:

$$r_{yy}[0] = -a_1 r_{yy}[-1] + \sigma_x^2$$

We can use the symmetry property of autocorrelation function i.e., $r_{yy}[-\ell] = r_{yy}[\ell]$:

$$r_{yy}[0] = -a_1 r_{yy}[1] + \sigma_x^2$$

To find an expression for $r_{yy}[1]$, we set $\ell = 1$ in the second equation:

$$r_{yy}[1] = -a_1 r_{yy}[0]$$

We insert the second equation into the first equation:

$$r_{yy}[0] = -a_1(-a_1 r_{yy}[0]) + \sigma_x^2$$

$$r_{yy}[0] = a_1^2 r_{yy}[0] + \sigma_x^2$$

$$\sigma_x^2 = r_{yy}[0] - a_1^2 r_{yy}[0]$$

$$\sigma_x^2 = r_{yy}[0](1 - a_1^2)$$

$$r_{yy}[0] = \frac{\sigma_x^2}{1 - a_1^2}$$

Now, we need to find an expression for $\ell > 0$. We can do this by using the second equation.

First, we set $\ell = 1$ in the second equation:

$$r_{yy}[1] = -a_1 r_{yy}[0] \Leftrightarrow -a_1 \frac{\sigma_x^2}{1 - a_1^2}$$

$$r_{yy}[2] = -a_1 r_{yy}[1] \Leftrightarrow -a_1 \left(-a_1 \frac{\sigma_x^2}{1 - a_1^2} \right) \Leftrightarrow (-a_1)^2 \frac{\sigma_x^2}{1 - a_1^2}$$

$$r_{yy}[3] = -a_1 r_{yy}[2] \Leftrightarrow -a_1 \left((-a_1)^2 \frac{\sigma_x^2}{1 - a_1^2} \right) \Leftrightarrow (-a_1)^3 \frac{\sigma_x^2}{1 - a_1^2}$$

This means that in general, we have:

$$r_{yy}[\ell] = (-a_1)^\ell \frac{\sigma_x^2}{1 - a_1^2}$$

If we apply the symmetric property of the autocorrelation, we get:

$$r_{yy}[\ell] = (-a_1)^{|\ell|} \frac{\sigma_x^2}{1 - a_1^2}$$

Problems True/False

The autocorrelation function of an AR(p) is equal to zero for lags larger than p .

Answer: FALSE!

An AR(p) process is effectively white noise sent through an IIR filter. This implies that a correlation will exist between infinitely distant samples. While the autocorrelation function can be zero for any lag it is generally different from zero for almost all lags.

Note though, that for all AR(2) processes encountered in real life, the autocorrelation values gets exponentially close to zero for large lags.

Quiz: Is the comb filter an MA(1) process?

One special type of filter, *the comb filter*, is given by

$$y(n) = x(n) - x(n - D)$$

where D is a positive integer. Is this an MA(1) process?

A: Yes

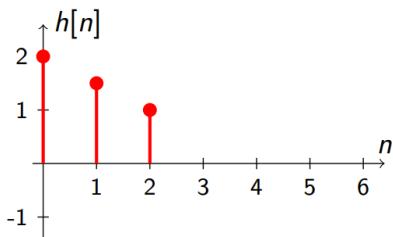
B: No

The answer is no!

Quiz: Determine kind of model based on the impulse response

The impulse response for a specific ARMA(p,q) model is shown below.

What kind of model is shown?



- A: ARMA(1,0)=AR(1)
- B: ARMA(2,0)=AR(2)
- C: ARMA(0,1)=MA(1)
- D: ARMA(1,1)
- E: There are two correct answers among A-D
- F: A-D are all wrong

The correct answer is F.

The autoregressive part of the ARMA(p,q) model has infinite impulse response. What we see in the figure is finite impulse response with three elements. Therefore, it must be an MA(q) model.

Since the figure shows three filter coefficients, the system must be an MA(2) process.

If the figure indicated infinite filter coefficients, we could not have determined the order of the model. In this case, it would be either an autoregressive model or a high-order MA(q) model.

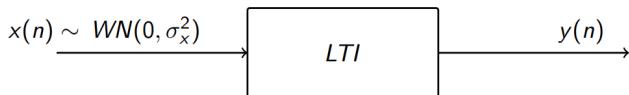
[✓] Exam 2012, Problem 2: Construct an AR(2) model using a signal from a random process

A sequence of data from a random process is given by

$$\{1, -1, 0, 2, 3, 2, -4, 1\}$$

Use the data to construct an *AR*(2) model of the random process and account for any assumptions you make.

We assume that the given sequence is the output of a process whose input was white Gaussian noise with zero mean and unit variance:



Given the input is white noise with zero mean $x(n) \sim WN(0, \sigma_x^2)$, the autocorrelation of an AR(q) model is given by Eq. 13.141:

$$r_{yy}(\ell) = -\sum_{k=1}^p a_k r_{yy}(\ell - k), \quad \ell > 0$$

This is useful because we can use it to find the coefficients $\{a_k\}$ of an AR(q) model. Given an output signal $y(n)$, we can compute the autocorrelation $r_{yy}(\ell)$ numerically in MATLAB. This means that the expression in Eq. 13.141 becomes a set of p linear equations.

For a second-order $p = 2$ AR model, we get two equations with two unknowns:

$$r_{yy}(1) = -a_1 r_{yy}(0) - a_2 r_{yy}(-1)$$

$$r_{yy}(2) = -a_1 r_{yy}(1) - a_2 r_{yy}(0)$$

We can write it into matrix form:

$$\begin{bmatrix} r_{yy}(0) & r_{yy}(1) \\ r_{yy}(1) & r_{yy}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = - \begin{bmatrix} r_{yy}(1) \\ r_{yy}(2) \end{bmatrix}$$

Solving it will give us the AR(2) model coefficients $\{a_1, a_2\}$.

This is implemented in the arfit() function:

```
y = [1, -1, 0, 2, 3, 2, -4, 1];
[a, ~] = arfit(y, 2)
```

```
a = 2x1
0.0340
0.2232
```

The coefficients for the AR(2) model is; $a_1 = 0.0340$ and $a_2 = 0.2232$.

$$y(n) = -[0.0340y(n-1) + 0.2232y(n-2)] + b_0x(n)$$

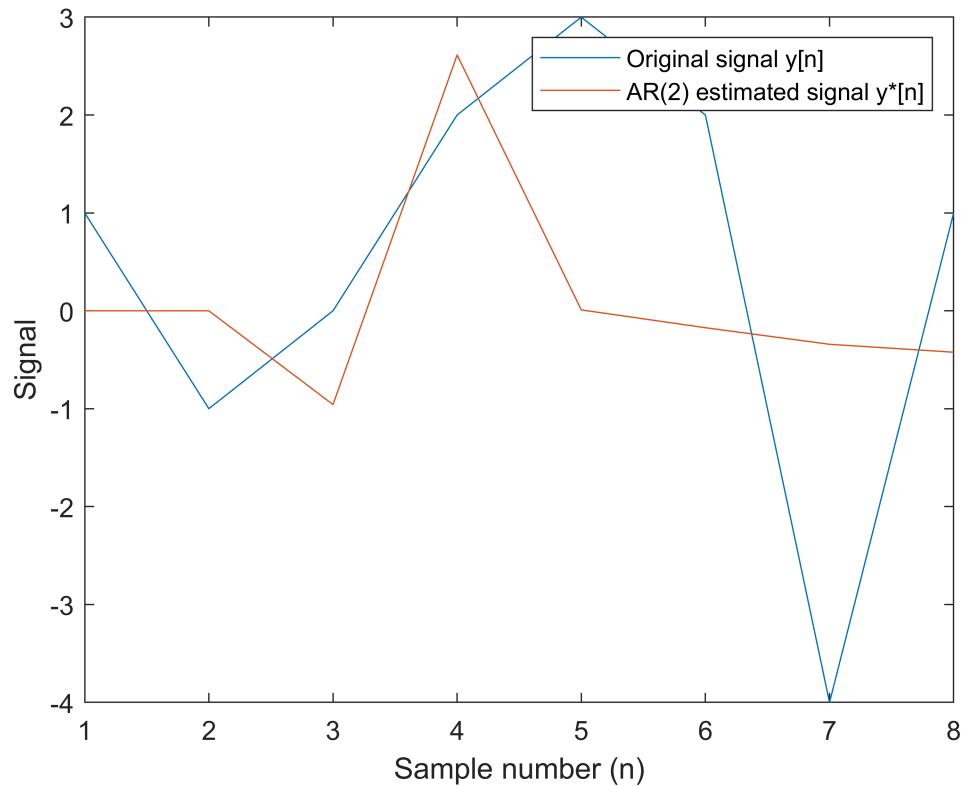
We can plot the signal and the estimated AR(2) signal:

```
N = 8;

% Generate the input
x = randn(N, 1)'; % Random noise with zero mean and unit variance.
b0 = 1;

% Estimate the output signal using the AR(2) model
y_hat = zeros(N, 1);
n = 3:N; % Trick to make it work in MATLAB
y_hat(n) = -(a(1)*y(n-1) + a(2)*y(n-2)) + b0*x(n);

% Plot the two
n = 1:N;
plot(n, y(n), n, y_hat(n))
legend('Original signal y[n]', 'AR(2) estimated signal y*[n]');
xlabel('Sample number (n)')
ylabel('Signal')
```



Exam 2016 Problem 1: AR(p) signal modelling

The first few values of the autocorrelation function of a random process has been determined as

$ l $	$r_x(l)$
0	11.26
1	9.70
2	6.00
3	1.49
4	-2.53

Initially, it can be assumed that the random process can be modeled as an AR(2) process.

```
clear variables;
```

1) Compute AR(2) model coefficient given autocorrelation sequence

- Calculate the model parameters and plot the power spectral density.

An AR(p) model is given by:

$$y(n) = -\sum_{k=1}^p [a_k y(n-k)] + b_0 x(n)$$

The autocorrelation of AR(q) model was derived in Eq. 13.141 as:

$$r_{yy}(\ell) = -\sum_{k=1}^p a_k r_{yy}(\ell-k), \quad \ell > 0$$

This is useful because we can use it to find the coefficients $\{a_k\}$ of an AR(q) model using the autocorrelation $r_{yy}(\ell)$ computed numerically in MATLAB. This means that the expression in Eq. 13.141 becomes a set of p linear equations.

An AR(2) model is given by:

$$y(n) = -(a_1 y(n-1) + a_2 y(n-2)) + b_0 x(n)$$

We can estimate the model parameters of a second-order AR model $p = 2$ by creating two equations with two unknowns:

$$r_{yy}(1) = -a_1 r_{yy}(0) - a_2 r_{yy}(-1)$$

$$r_{yy}(2) = -a_1 r_{yy}(1) - a_2 r_{yy}(0)$$

We can write it into matrix form using the Toeplitz matrix:

$$\begin{bmatrix} r_{yy}(0) & r_{yy}(-1) \\ r_{yy}(1) & r_{yy}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} r_{yy}(1) \\ r_{yy}(2) \end{bmatrix}$$

In the general case, it becomes:

$$\begin{bmatrix} r_{yy}[0] & r_{yy}[1] & \dots & r_{yy}[p-1] \\ r_{yy}[1] & r_{yy}[0] & \dots & r_{yy}[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_{yy}[p-1] & r_{yy}[p-2] & \dots & r_{yy}[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = -\begin{bmatrix} r_{yy}[1] \\ r_{yy}[2] \\ \vdots \\ r_{yy}[p] \end{bmatrix},$$

More compactly written as *Yule–Walker equations*:

$$\mathbf{R}_y \mathbf{a} = -\mathbf{r}_y. \quad (13.143)$$

The input noise variance can be compute as follows:

$$\sigma_x^2 = \sigma_y^2 + \mathbf{a}^T \mathbf{r}_y = \sigma_y^2 - \mathbf{r}_y^T \mathbf{R}_y^{-1} \mathbf{r}_y \leq \sigma_y^2. \quad (13.144)$$

This is coded in MATLAB function (see end of document):

```
p = 2;
r_xx = [11.26, 9.70, 6.00, 1.49, -2.53]';
[a, v] = ar_from_acrs(r_xx, p)

a = 2x1
-1.5604
0.8114
v = 0.9922
```

The power spectrum of an ARMA(p, q) process is given by:

$$S_{yy}(\omega) = \sigma_x^2 |H(e^{j\omega})|^2 = \sigma_x^2 \left| \frac{\sum_{k=0}^q b_k e^{-j\omega k}}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2. \quad (13.133)$$

The power spectrum of an AR(p) process is given by:

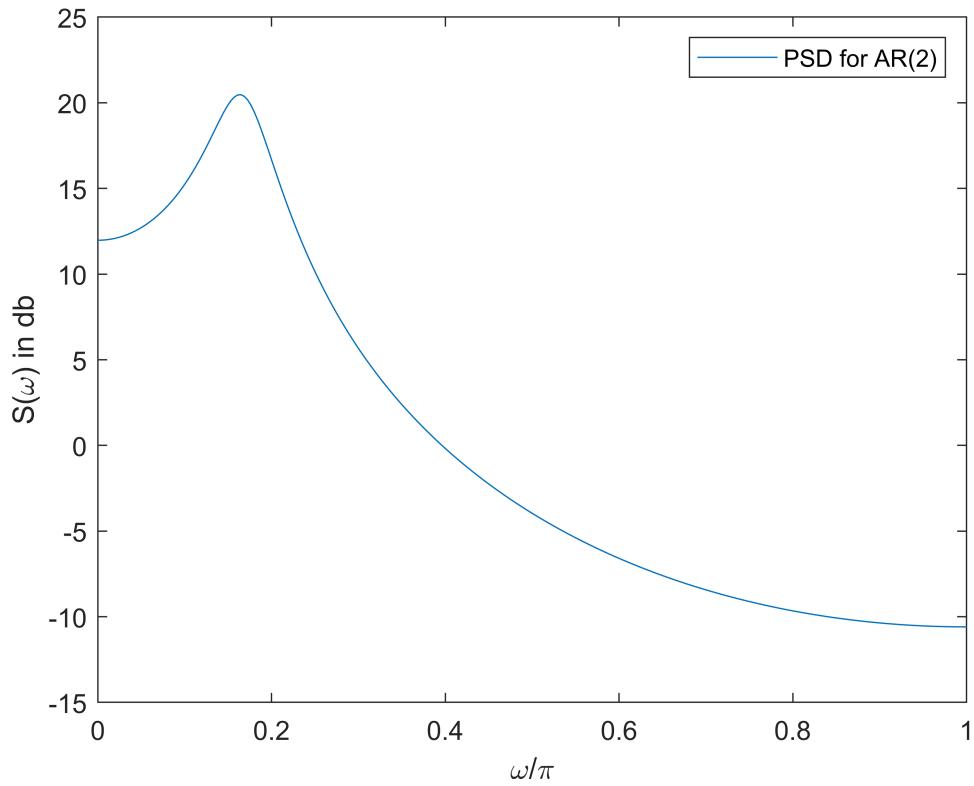
$$S_{yy}(\omega) = \sigma_x^2 \left| \frac{1}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2$$

To solve this problem, the method is as follows:

1. Find the coefficients $\{a_1, a_2, \dots, a_p\}$ for the AR(p) model
2. Compute the transfer function for the AR(p) by computing the sum and finding its reciprocal
3. Compute the conjugate of the transfer function: $|H(e^{j\omega})|^2$
4. Multiply it with the variance σ_x^2

The method above is implemented in the function ar2psd (see at the end of document):

```
[S, w] = ar2psd(a, v, 256);
plot(w/pi, real(pow2db(S)))
legend('PSD for AR(2)')
xlabel('\omega/\pi')
ylabel('S(\omega) in db')
```



2) Is the random process better modelled as an AR(3) or higher-order process?

2. Decide if the random process is better modeled as an AR(3) or higher order process.

The variance σ_x^2 calculated above corresponds to the minimum Mean Squared Error (MSE) for the AR(2) model. So for a higher order model to be a better model the MSE must be lower than for the AR(2) model.

We can solve the equations for different AR(q) models:

```
MSE_ar2 = v
```

```
MSE_ar2 = 0.9922
```

```
[~, MSE_ar3] = ar_from_acrs(r_xx, 3)
```

```
MSE_ar3 = 0.9922
```

```
[~, MSE_ar4] = ar_from_acrs(r_xx, 4)
```

```
MSE_ar4 = 0.9919
```

The MSE is lowered very slightly when the model order is increased.

Because of the slight change we can conclude that the process is not better modeled as an AR(3) or AR(4) process.

Exam 2018 Problem 2: Signal Modelling using MA(2) or AR(2)

```
clear variables;
```

Consider the following set of data measured from an unknown random process.

```
{ -2.39 2.28 0.65 -4.68 2.88 0.13 -2.48 1.78 2.27 -4.49 1.13 0.90 -0.92 }
```

[✓] 1) Calculate and plot the autocorrelation function of the data

1. Calculate and plot the autocorrelation function of the data for lags $0 \leq l \leq 4$.

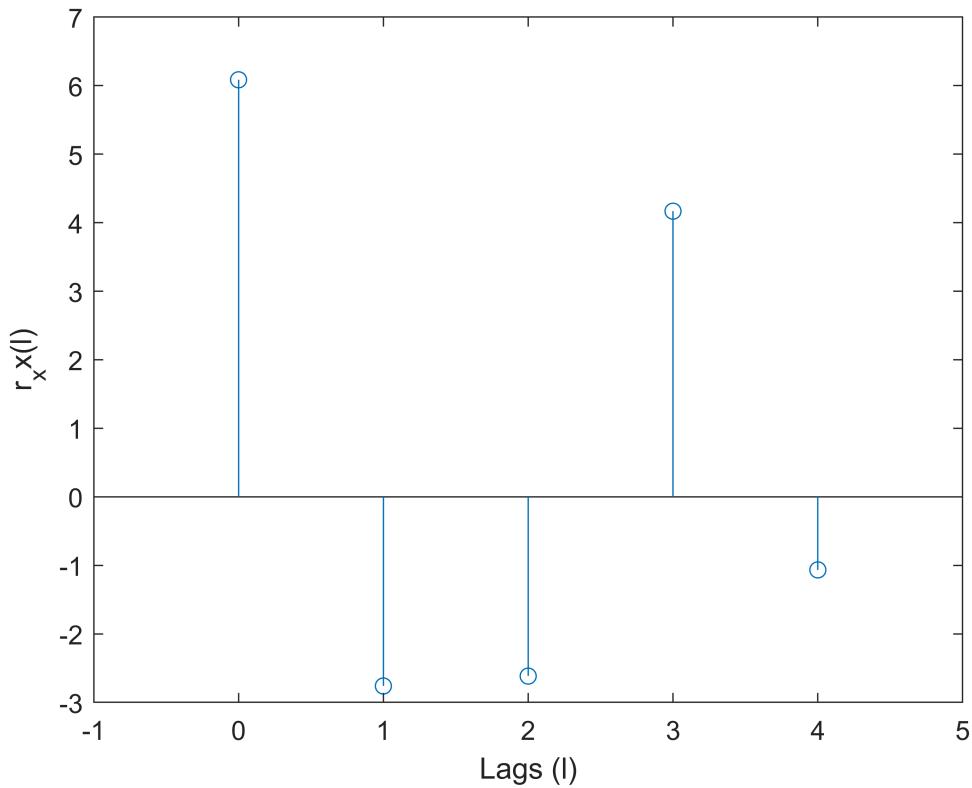
```
x = [-2.39, 2.28, 0.65, -4.68, 2.88, 0.13, -2.48, 1.78, 2.27, -4.49, 1.13, 0.9, -0.92];
```

```
% Estimate autocorrelation using data  
[r_xx, ell] = xcorr(x, 'biased');
```

```
% Print out the lags 0 to 4  
mid = floor(numel(ell)/2)+1;  
r_xx(mid:mid+4)
```

```
ans = 1x5  
6.0855 -2.7605 -2.6160 4.1680 -1.0665
```

```
% Plot the results  
stem(0:4, r_xx(mid:mid+4))  
xlim([-1, 5])  
xlabel('Lags (l)')  
ylabel('r_xx(l)')
```



2) Can data be described by an MA(2) model?

2. Discuss whether the data describe an MA(2) model.

MA(2) model is given by:

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2]$$

where $x[n] \sim WN(0, \sigma_x^2)$

The autocorrelation for an MA(2) process is:

$$r_{yy}(\ell) = (b_0^2 + b_1^2 + b_2^2)\delta(\ell) + (b_0b_1 + b_1b_2)\delta(\ell+1) + (b_0b_1 + b_1b_2)\delta(\ell-1) + b_0b_2\delta(\ell+2) + b_0b_2\delta(\ell-2)$$

Clearly, the autocorrelation for an MA(2) for $\ell > 2$ is zero.

The estimated autocorrelation for the data is:

```
r_xx(mid:mid+4)
```

```
ans = 1x5
 6.0855   -2.7605   -2.6160    4.1680   -1.0665
```

Since the estimated autocorrelation has non-zero values for $\ell > 2$, it cannot be described by an MA(2) model.

3) Calculate the power spectral density of the random process assuming an AR(2) model

3. Calculate the power spectral density of the random process assuming an AR(2) model.

Before we can compute the PSD, we need to find the AR(2) model coefficients.

An AR(p) model is given by:

$$y(n) = -\sum_{k=1}^p [a_k y(n-k)] + b_0 x(n)$$

The autocorrelation of AR(q) model was derived in Eq. 13.141 as:

$$r_{yy}(\ell) = -\sum_{k=1}^p a_k r_{yy}(\ell-k), \quad \ell > 0$$

This is useful because we can use it to find the coefficients $\{a_k\}$ of an AR(q) model using the autocorrelation $r_{yy}(\ell)$ computed numerically in MATLAB. This means that the expression in Eq. 13.141 becomes a set of p linear equations.

An AR(2) model is given by:

$$y(n) = -(a_1 y(n-1) + a_2 y(n-2)) + b_0 x(n)$$

We can estimate the model parameters of a second-order AR model $p = 2$ by creating two equations with two unknowns:

$$r_{yy}(1) = -a_1 r_{yy}(0) - a_2 r_{yy}(-1)$$

$$r_{yy}(2) = -a_1 r_{yy}(1) - a_2 r_{yy}(0)$$

We can write it into matrix form using the Toeplitz matrix:

$$\begin{bmatrix} r_{yy}(0) & r_{yy}(-1) \\ r_{yy}(1) & r_{yy}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} r_{yy}(1) \\ r_{yy}(2) \end{bmatrix}$$

In the general case, it becomes:

$$\begin{bmatrix} r_{yy}[0] & r_{yy}[1] & \dots & r_{yy}[p-1] \\ r_{yy}[1] & r_{yy}[0] & \dots & r_{yy}[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_{yy}[p-1] & r_{yy}[p-2] & \dots & r_{yy}[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = -\begin{bmatrix} r_{yy}[1] \\ r_{yy}[2] \\ \vdots \\ r_{yy}[p] \end{bmatrix},$$

More compactly written as:

$$\mathbf{R}_y \mathbf{a} = -\mathbf{r}_y$$

This is a system of linear equations which can be solved in MATLAB using `arfit` function (see bottom of the document).

The power spectrum of an ARMA(p, q) process is given by:

$$S_{yy}(\omega) = \sigma_x^2 |H(e^{j\omega})|^2 = \sigma_x^2 \left| \frac{\sum_{k=0}^q b_k e^{-j\omega k}}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2. \quad (13.133)$$

The power spectrum of an AR(p) process is given by:

$$S_{yy}(\omega) = \sigma_x^2 \left| \frac{1}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2$$

To solve this problem, the method is as follows:

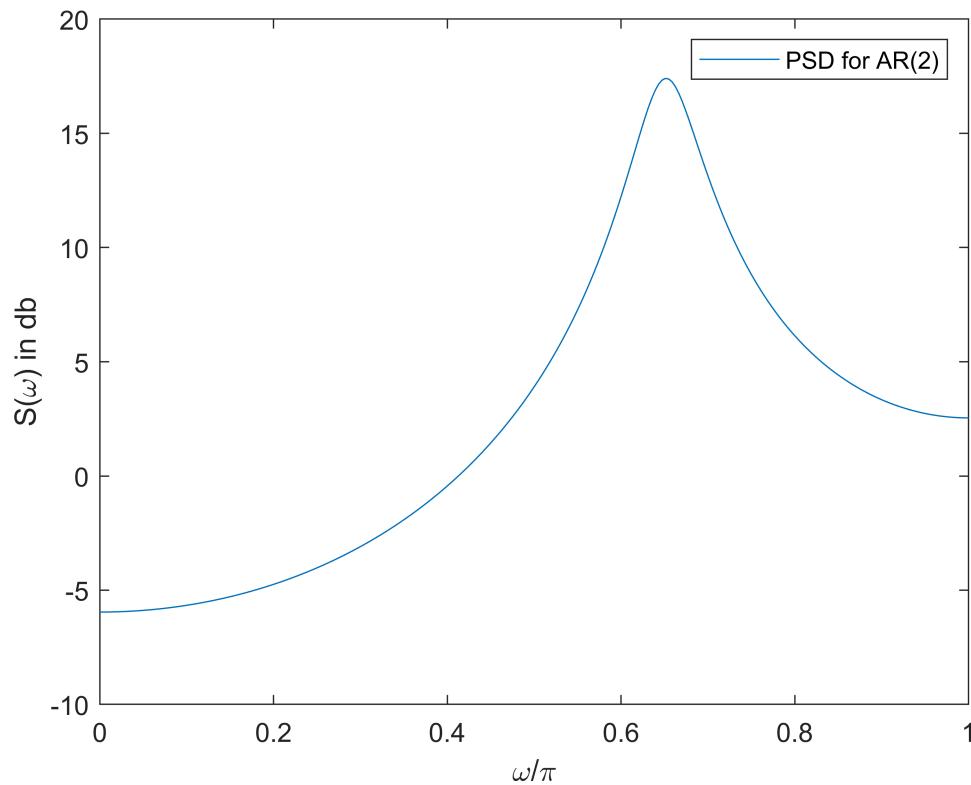
1. Find the coefficients $\{a_1, a_2, \dots, a_p\}$ for the AR(p) model
2. Compute the transfer function for the AR(p) by computing the sum and finding its reciprocal
3. Compute the conjugate of the transfer function: $|H(e^{j\omega})|^2$
4. Multiply it with the variance σ_x^2

The method above is implemented in the functions `arfit()` and `ar2psd()`:

```
[a, v] = arfit(x, 2)
```

```
a = 2x1
0.8167
0.8003
v = 1.7374
```

```
[S, w] = ar2psd(a, v, 256);
plot(w/pi, real(pow2db(S)))
legend('PSD for AR(2)')
xlabel('\omega/\pi')
ylabel('S(\omega) in db')
```



Problem 1: Signal modelling assuming AR(2) process

The first few values of the autocorrelation function of a random process has been determined as

$ l $	$r_x(l)$
0	11.26
1	9.70
2	6.00
3	1.49
4	-2.53

Initially, it can be assumed that the random process can be modeled as an AR(2) process.

```
clear variables;
```

1) Compute AR(2) model coefficient given autocorrelation sequence

1. Calculate the model parameters and plot the power spectral density.

An AR(p) model is given by:

$$y(n) = -\sum_{k=1}^p [a_k y(n-k)] + b_0 x(n)$$

The autocorrelation of AR(q) model was derived in Eq. 13.141 as:

$$r_{yy}(\ell) = -\sum_{k=1}^p a_k r_{yy}(\ell-k), \quad \ell > 0$$

This is useful because we can use it to find the coefficients $\{a_k\}$ of an AR(q) model using the autocorrelation $r_{yy}(\ell)$ computed numerically in MATLAB. This means that the expression in Eq. 13.141 becomes a set of p linear equations.

An AR(2) model is given by:

$$y(n) = -(a_1 y(n-1) + a_2 y(n-2)) + b_0 x(n)$$

We can estimate the model parameters of a second-order AR model $p = 2$ by creating two equations with two unknowns:

$$r_{yy}(1) = -a_1 r_{yy}(0) - a_2 r_{yy}(-1)$$

$$r_{yy}(2) = -a_1 r_{yy}(1) - a_2 r_{yy}(0)$$

We can write it into matrix form using the Toeplitz matrix:

$$\begin{bmatrix} r_{yy}(0) & r_{yy}(-1) \\ r_{yy}(1) & r_{yy}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} r_{yy}(1) \\ r_{yy}(2) \end{bmatrix}$$

In the general case, it becomes:

$$\begin{bmatrix} r_{yy}[0] & r_{yy}[1] & \dots & r_{yy}[p-1] \\ r_{yy}[1] & r_{yy}[0] & \dots & r_{yy}[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_{yy}[p-1] & r_{yy}[p-2] & \dots & r_{yy}[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = -\begin{bmatrix} r_{yy}[1] \\ r_{yy}[2] \\ \vdots \\ r_{yy}[p] \end{bmatrix},$$

More compactly written as *Yule–Walker equations*:

$$\mathbf{R}_y \mathbf{a} = -\mathbf{r}_y. \quad (13.143)$$

The input noise variance can be compute as follows:

$$\sigma_x^2 = \sigma_y^2 + \mathbf{a}^T \mathbf{r}_y = \sigma_y^2 - \mathbf{r}_y^T \mathbf{R}_y^{-1} \mathbf{r}_y \leq \sigma_y^2. \quad (13.144)$$

This is coded in MATLAB function (see end of document):

```
p = 2;
```

```
r_xx = [11.26, 9.70, 6.00, 1.49, -2.53]';
[a, v] = ar_from_acrs(r_xx, p)
```

```
a = 2x1
-1.5604
 0.8114
v = 0.9922
```

The power spectrum of an ARMA(p, q) process is given by:

$$S_{yy}(\omega) = \sigma_x^2 |H(e^{j\omega})|^2 = \sigma_x^2 \left| \frac{\sum_{k=0}^q b_k e^{-j\omega k}}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2. \quad (13.133)$$

The power spectrum of an AR(p) process is given by:

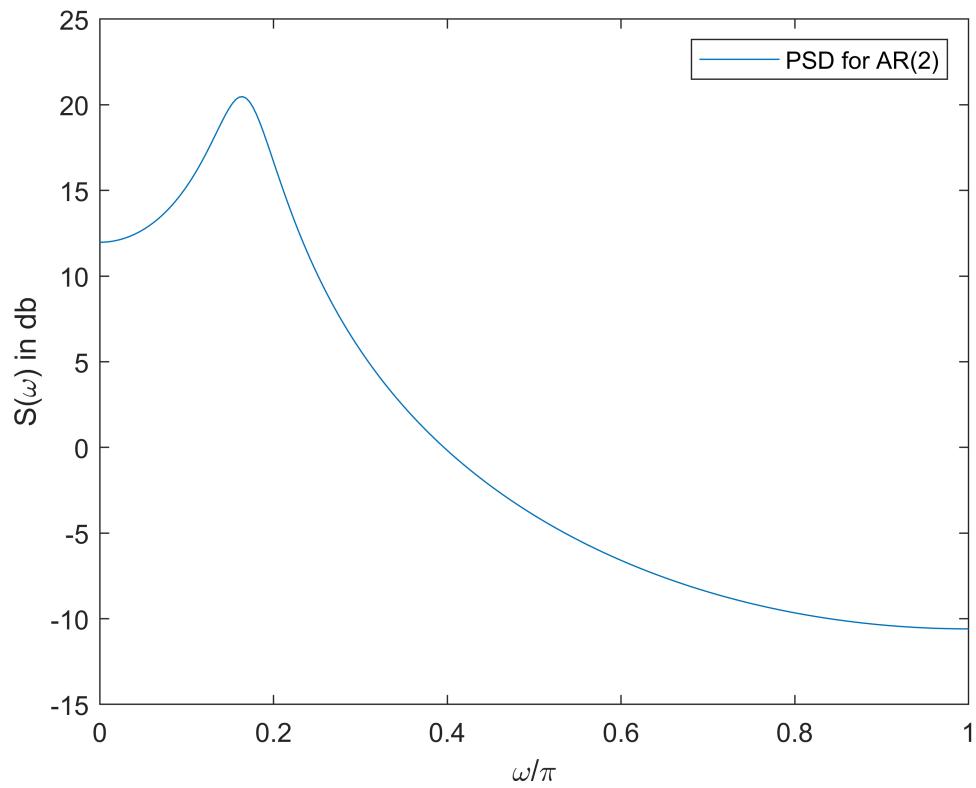
$$S_{yy}(\omega) = \sigma_x^2 \left| \frac{1}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2$$

To solve this problem, the method is as follows:

1. Find the coefficients $\{a_1, a_2, \dots, a_p\}$ for the AR(p) model
2. Compute the transfer function for the AR(p) by computing the sum and finding its reciprocal
3. Compute the conjugate of the transfer function: $|H(e^{j\omega})|^2$
4. Multiply it with the variance σ_x^2

The method above is implemented in the function ar2psd (see at the end of document):

```
[S, w] = ar2psd(a, v, 256);
plot(w/pi, real(pow2db(S)))
legend('PSD for AR(2)')
xlabel('\omega/\pi')
ylabel('S(\omega) in db')
```



2) Is the random process better modelled as an AR(3) or higher-order process?

2. Decide if the random process is better modeled as an AR(3) or higher order process.

The variance σ_x^2 calculated above corresponds to the minimum Mean Squared Error (MSE) for the AR(2) model. So for a higher order model to be a better model the MSE must be lower than for the AR(2) model.

We can solve the equations for different AR(q) models:

```
MSE_ar2 = v
```

```
MSE_ar2 = 0.9922
```

```
[~, MSE_ar3] = ar_from_acrs(r_xx, 3)
```

```
MSE_ar3 = 0.9922
```

```
[~, MSE_ar4] = ar_from_acrs(r_xx, 4)
```

```
MSE_ar4 = 0.9919
```

The MSE is lowered very slightly when the model order is increased.

Because of the slight change we can conclude that the process is not better modeled as an AR(3) or AR(4) process.

Functions

```
function [S, w] = ar2psd(a, v, N)
% AR2PSD Compute the Power Spectral Density from AR(p) coefficients
% [S, w] = ar2psd(a, v, N)
% a: AR(p) coefficients
% v: the variance
% N: number of points in the range [1, pi]
% S: the estimated power spectrum
% w: frequencies
w = linspace(0, 1, N) * pi;

% Compute the transfer function
% Used Eq. (13.133) in the book
H = ones(N, 1);
for k=1:numel(a)
    H = H + a(k)*exp(-1j * w' * k);
end
H = 1./H;

% Finally compute the PSD
S = v * H.*conj(H);
end

function [a,v] = arfit(x,p)
% fit AR(p) model from data
% x: data
% p: model order
% a: a coefficients
% v: variance
if isrow(x)
    x = x'; % Convert to a column vector
end

% Compute the autocorrelation
[r_xx, lags] = xcorr(x, p, 'biased');

% Select elements r_xx[0] to r_xx[p-1]
R_elems = r_xx(p+1:2*p);

% Create the Toeplitz matrix
R = toeplitz(R_elems);

% Select elements r_xx[1] to r_xx[p]
r = r_xx(p+2:2*p+1);

% Solve systems of linear equations using mldivide function
a = mldivide(R, -r);
```

```

% Compute the variance
v = r_xx(p+1) + a'*r;
end

function [a, v] = ar_from_acrs(r_xx, p)
R_xx = toeplitz(r_xx(1:p));

% Select elements r_xx[1] to r_xx[p]
r = r_xx(2:p+1);

% Solve Yule-Walker equation (13.143)
a = mldivide(R_xx, -r);

% Compute the variance according to Eq. (13.144)
v = r_xx(1) + a'*r;
end

```

Autocorrelation Functions

Table of Contents

Autocorrelation.....	1
Cross-correlation.....	1
Details.....	2
Properties of autocorrelation sequence.....	2
Fundamental property of autocorrelation $r(0) \geq r(l)$	2
Autocorrelation and cross-correlation of complex signals.....	2
Cross-correlation of two uncorrelated noise processes.....	2
Relations between $r(\ell)$ and $r(-\ell)$	3
Autocorrelation of complex signal.....	3
Autocorrelation of a real cosine signal.....	4
Autocorrelation of a real sine signal.....	5
Autocorrelation of white noise.....	6
System identification with white noise.....	7
High frequency noise vs low frequency noise.....	7
Problems.....	8
If a signal is scaled by 2, does its autocorrelation also scale by 2?.....	8
How to interpret ACRS estimate of the noise process.....	9
[✓] ADSI Problem 4.2: Autocorrelation functions from plot.....	10
1) Sketch and explain what the autocorrelation looks like.....	11
2) Use computer simulations to verify the above result.....	11
[✓] ADSI Problem 4.4: Autocorrelation of complex signals.....	13
1) Compute the analytic expression for the autocorrelation of the complex sinusoid.....	13
2) How are $r_x(l)$ and $r_x(-l)$ related for a complex signal?.....	14
3) Compute the autocorrelation of a real cosine signal.....	14
[✓] ADSI Problem 4.5: Autocorrelation of distorted signals.....	15
1) Derive autocorrelation function of the output signal.....	16
2) Is possible to remove the distortion from the output signal?.....	19
ADSI Problem 4.6: System identification using ARMA(2,2) as input.....	20
1) Simulate the autocorrelation and cross-correlation in MATLAB.....	20
2) Compute the transfer function and impulse response from the autocorrelation.....	21

Autocorrelation

The correlation of a signal with itself is its *autocorrelation*:

$$w(t) = u(t) \otimes u(t) = \int_{-\infty}^{\infty} u^*(\tau - t)u(\tau)d\tau$$

Cross-correlation

The *cross-correlation* between two signals $u(t)$ and $v(t)$ is

$$w(t) = u(t) \otimes v(t) \triangleq \int_{-\infty}^{\infty} u^*(\tau)v(\tau + t)d\tau$$

Details

Properties of autocorrelation sequence

Properties of autocorrelation functions:

1. $r_{xx}(0) = \overline{X^2}$
2. $r_{xx}(l) = r_{xx}(-l)$
3. $r_{xx}(0) \geq |r_{xx}(l)|$
4. If $X(k) = \overline{X} + N(k)$ then $r_{xx}(l) = \overline{X^2} + r_{NN}(l)$
5. If $X(k) = A \cos(\omega k + \theta) + N(k)$ then
 $r_{xx}(l) = \frac{A^2}{2} \cos(\omega l) + r_{NN}(l)$
6. $\lim_{|\tau| \rightarrow \infty} r_{xx}(l) = 0$ for ergodic, zero-mean processes with no periodic components
7. $\mathcal{F}[r_{xx}(l)] \geq 0 \quad \forall \omega$

Fundamental property of autocorrelation $r(0) \geq r(l)$

A fundamental property of the autocorrelation function is that $r_x(0) \geq r_x(\ell)$ for all ℓ .

Autocorrelation and cross-correlation of complex signals

The autocorrelation function of a complex signal is given by:

$$r_{xx}(\ell) = E[x(n)x^*(n - \ell)]$$

The cross-correlation function of a complex signal is given by:

$$r_{yx}(\ell) = E[y(n)x^*(n - \ell)]$$

Cross-correlation of two uncorrelated noise processes

Let $v(n)$ and $w(n)$ be **two uncorrelated white noise** processes with variance $\sigma_v^2 = 0.49$ and $\sigma_w^2 = 1$. The cross-correlation between these processes is:

$$r_{vw}(\ell) = 0$$

Relations between $r(\text{ell})$ and $r(-\text{ell})$

Let us compute the autocorrelation for $-\ell$:

$$r_{xx}(-\ell) = E[x(n)x^*(n + \ell)]$$

Suppose $m = n + \ell$. Then we can write $n = m - \ell$. Let us substitute all n with m in the above expression:

$$r_{xx}(-\ell) = E[x(m - \ell)x^*(m)]$$

Computing the complex conjugate of the expectation we get:

$$r_{xx}(-\ell) = E[x^*(m - \ell)x(m)]^*$$

$$r_{xx}(-\ell) = E[x(m)x^*(m - \ell)]^*$$

Since $r_{xx}^*(-\ell) = E[x(m)x^*(m - \ell)]^*$, we know that:

$$r_{xx}(-\ell) = r_{xx}^*(\ell)$$

Autocorrelation of complex signal

Problem:

What is the autocorrelation function of the complex sinusoid $x(n) = A e^{j(\omega n + \phi)}$ where A and ω are real constants and $\phi \sim U(0, 2\pi)$

Solution:

The autocorrelation function for complex signals can be computed as:

$$r_{xx}(\ell) = E[z(n)z^*(n - \ell)]$$

Plugging the given complex sinusoid into the formula, we get:

$$r_{xx}(\ell) = E[A e^{j(\omega n + \phi)} A e^{-j(\omega(n - \ell) + \phi)}]$$

Since A is a constant, we can move it outside the expected value:

$$r_{xx}(\ell) = A^2 E[e^{j(\omega n + \phi)} e^{-j(\omega(n - \ell) + \phi)}]$$

$$r_{xx}(\ell) = A^2 E[e^{j\omega n + j\phi} e^{-j\omega n + j\omega \ell - j\phi}]$$

$$r_{xx}(\ell) = A^2 E[e^{j\omega n + j\phi + (-j\omega n + j\omega \ell - j\phi)}]$$

$$r_{xx}(\ell) = A^2 E[e^{j\omega n + j\phi - j\omega n + j\omega \ell - j\phi}]$$

$$r_{xx}(\ell) = A^2 E[e^{j\omega \ell}]$$

We know that $E[e^{j\omega \ell}] = e^{j\omega \ell}$ because the expected value of a constant is just the constant itself. Notice that ϕ is no longer in the expression $e^{j\omega \ell}$. Therefore, the autocorrelation of the complex sinusoid is:

$$r_{xx}(\ell) = A^2 e^{j\omega \ell}$$

Thus, the autocorrelation of the complex sinusoid $z(n) = A e^{j(\omega n + \phi)}$ where A and ω are real constants and $\phi \sim U(0, 2\pi)$ is given by:

$$r_{zz}(\ell) = A^2 e^{j\omega \ell}$$

Autocorrelation of a real cosine signal

Problem:

What is the autocorrelation function of a real signal $x(n) = A \cos(\omega n + \phi)$ where A and ω are real constants and $\phi \sim U(0, 2\pi)$?

Solution:

In ADSI Problem 4.4.1, we know that the autocorrelation of the complex sinusoid $z(n) = A e^{j(\omega n + \phi)}$ where A and ω are real constants and $\phi \sim U(0, 2\pi)$ is given by:

$$r_{zz}(\ell) = A^2 e^{j\omega \ell}$$

Since the result from 1) uses Euler, we need to convert the signal to complex exponential.

We use the relation $\cos(\theta) = \frac{1}{2}(e^{j\theta} + e^{-j\theta})$

$$x(n) = A \cos(\omega n + \phi)$$

$$x(n) = \frac{A}{2} (e^{j(\omega n + \phi)} + e^{-j(\omega n + \phi)})$$

$$x(n) = \frac{A}{2} e^{j(\omega n + \phi)} + \frac{A}{2} e^{-j(\omega n + \phi)}$$

In 1) we found that the autocorrelation of a complex sinusoid $x(n) = A e^{j(\omega n + \phi)}$ is $r_{xx}(\ell) = A^2 e^{j\omega\ell}$

Therefore, the autocorrelation of the real signal is:

$$r_{xx}(\ell) = \left(\frac{A}{2}\right)^2 e^{j\omega\ell} + \left(\frac{A}{2}\right)^2 e^{-j\omega\ell}$$

$$r_{xx}(\ell) = \frac{A^2}{4} e^{j\omega\ell} + \frac{A^2}{4} e^{-j\omega\ell}$$

$$r_{xx}(\ell) = \frac{A^2}{4} (e^{j\omega\ell} + e^{-j\omega\ell})$$

$$r_{xx}(\ell) = \frac{A^2}{2} \frac{1}{2} (e^{j\omega\ell} + e^{-j\omega\ell})$$

Using the relation $\cos(\theta) = \frac{1}{2}(e^{j\theta} + e^{-j\theta})$ we can rewrite the autocorrelation to:

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

Thus, the autocorrelation function of a real signal $z(n) = A \cos(\omega n + \phi)$ where A and ω are real constants and $\phi \sim U(0, 2\pi)$ is:

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

Autocorrelation of a real sine signal

Problem:

What is the autocorrelation function of a real signal $z(n) = A \sin(\omega n + \phi)$ where A and ω are real constants and $\phi \sim U(0, 2\pi)$?

Solution:

In ADSI Problem 4.4, we found that the autocorrelation of a complex sinusoid given

by $y(n) = A e^{j(\omega n + \phi)}$ where A and ω are real constants and ϕ is a random variable with $\phi \sim U(0, 2\pi)$ is:

$$r_{yy}(\ell) = A^2 e^{j\omega\ell}$$

To use this result, we need to convert the given signal in this problem to complex exponential signal.

A complex exponential signal is always formed by the sum of two real signals:

$$A e^{j\omega n} = A \cos(\omega n) + j A \sin(\omega n)$$

Therefore, we know that:

$$\sin(\omega) = \frac{1}{2j} e^{j\omega} - \frac{1}{2j} e^{-j\omega}$$

Using this relation, we can rewrite a real signal $A \sin(\omega n + \phi)$ as:

$$z(n) = A \sin(\omega n + \phi)$$

$$z(n) = \frac{A}{2j} e^{j(\omega n + \phi)} - \frac{A}{2j} e^{-j(\omega n + \phi)}$$

To compute the autocorrelation function, we square the magnitude, remove the phase and replace n with ℓ :

$$r_{zz}(\ell) = \left(\frac{A}{2j} \right)^2 e^{j\omega\ell} - \left(\frac{A}{2j} \right)^2 e^{-j\omega\ell}$$

We know that $(2j)^2 = 2^2 \cdot j^2 = -4$ because $j = \sqrt{-1}$ so $j^2 = -1$

$$r_{zz}(\ell) = \frac{A^2}{-4} e^{j\omega\ell} - \frac{A^2}{-4} e^{-j\omega\ell}$$

$$r_{zz}(\ell) = -\frac{A^2}{4} e^{j\omega\ell} + \frac{A^2}{4} e^{-j\omega\ell}$$

We want to make the autocorrelation function in terms of $\cos(\cdot)$, we rewrite the expression as follows:

$$r_{zz}(\ell) = \left(-\frac{A^2}{2} \right) \cdot \frac{1}{2} (e^{j\omega\ell} + e^{-j\omega\ell})$$

Since $\cos(\theta) = \frac{1}{2}(e^{j\theta} + e^{-j\theta})$, we can rewrite the expression as:

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

Thus, the autocorrelation function of a real signal $z(n) = A \sin(\omega n + \phi)$ is

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

Autocorrelation of white noise

White noise is important for random signal modelling.

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:

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

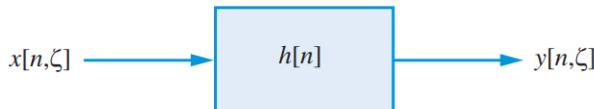
$$\sigma_w^2 = E[w^2(n)] - 0 \quad (\text{by definition } E[w(n)] = 0)$$

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

System identification with white noise

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.

We have an unknown system with an impulse response $h[n]$.



As input to this system, we give it a zero-mean white noise signal (a realisation of white noise random process):

$$x(n) \sim \text{WN}(0, \sigma_x^2)$$

The cross-correlation between this white noise signal and output of the system is given by Eq. 13.100:

$$r_{yx}[\ell] = \sum_{k=-\infty}^{\infty} h[k] r_{xx}[\ell - k] = h[\ell] * r_{xx}[\ell]. \quad (13.100)$$

Since $r_{xx}(\ell) = \sigma_x^2 \delta(\ell)$ the convolution becomes:

$$r_{yx}(\ell) = h(\ell) * \sigma_x^2 \delta(\ell)$$

$$r_{yx}(\ell) = \sigma_x^2 h(\ell)$$

This means that the cross-correlation is just the impulse response.

High frequency noise vs low frequency noise

We can use the autocorrelation of a noise to determine whether it is low-frequency vs high-frequency.

Plotting the ACRS, we see that the high-frequency noise oscillates whereas the low-frequency noise does not.

```

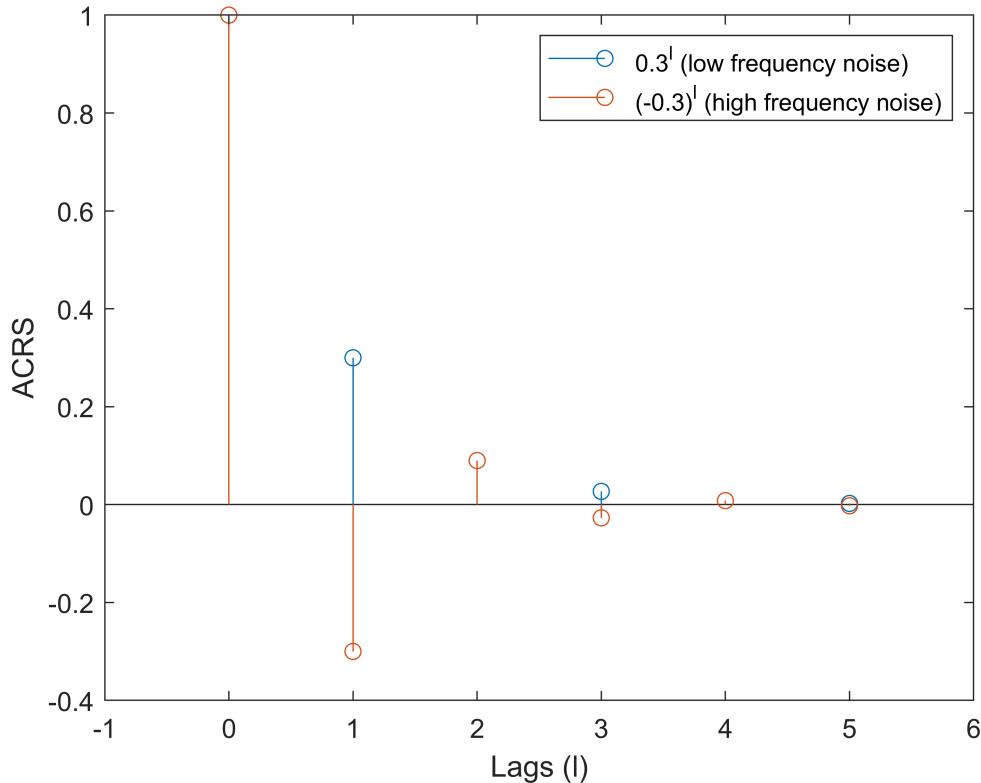
clear variables;
ell = 0:5;

```

```

stem(ell, 0.3.^ell)
hold on;
stem(ell, (-0.3).^ell)
hold off;
legend('0.3^l (low frequency noise)', '(-0.3)^l (high frequency noise)')
xlim([min(ell)-1, max(ell)+1])
hold off;
xlabel('Lags (l)')
ylabel('ACRS')

```



Problems

If a signal is scaled by 2, does its autocorrelation also scale by 2?

If a signal $x(n)$ is scaled to become $y(n) = 2x(n)$, the autocorrelation is scaled similarly and $r_y(l) = 2r_x(l)$.

Answer: FALSE.

The autocorrelation of a signal $x(n)$ is defined as:

$$r_x(\ell) = E[x(n)x(n - \ell)]$$

We can compute the autocorrelation of signal $y(n) = 2x(n)$ as follows:

$$r_y(\ell) = E[y(n)y(n - \ell)]$$

$$r_y(\ell) = E[2x(n)2x(n - \ell)]$$

$$r_y(\ell) = 4 \cdot E[x(n)x(n - \ell)]$$

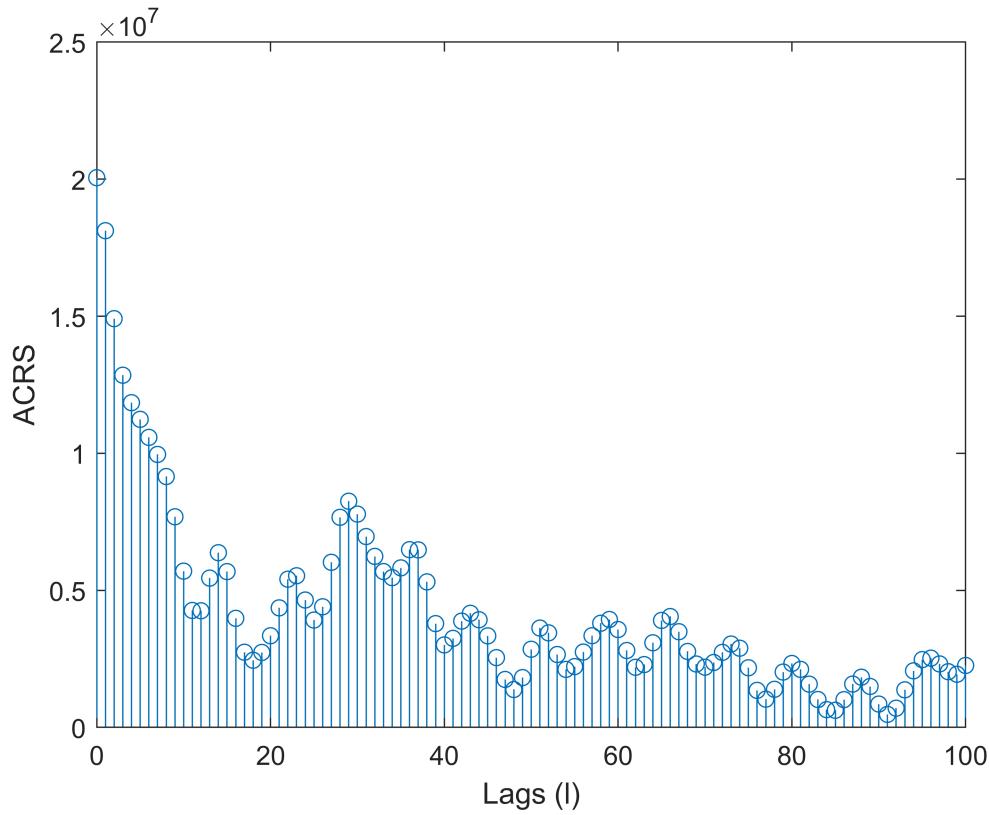
$$r_y(\ell) = 4r_x(\ell)$$

How to interpret ACRS estimate of the noise process.

This problem uses the signal file f16.mat that contains **noise** recorded at the co-pilot's seat of an F-16 airplane using a 16 bit A/D converter with $F_S = 19.98$ kHz.

```
load('f16.mat')
y = f16;
L = 100;

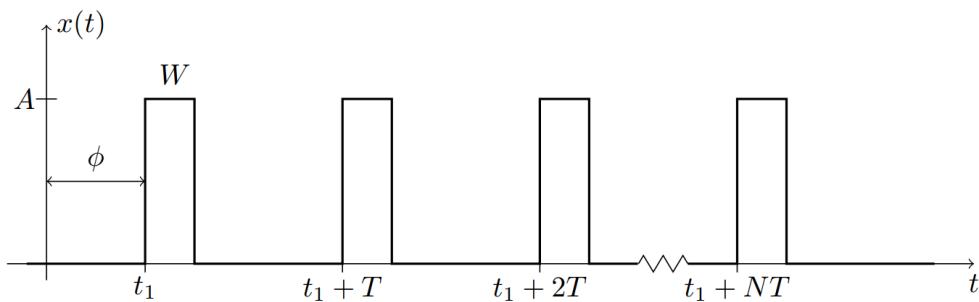
[lags, r_yy] = xcorr(y, L, 'biased');
mid = ceil(numel(r_yy)/2);
stem(r_yy(mid:end), lags(mid:end))
xlabel('Lags (1)')
ylabel('ACRS')
```



- From the autocorrelation sequence, we see that there is some structure in the signal. The sequence is not decaying smoothly from lag zero to 100.
- The ACRS does not change sign so it seems to be some kind is low-frequency signal
- In order for the ACRS to go from lags zero to 100, it must be some higher-order process AR(p) model. It is definitely not a MA(q) process model.

[✓] ADSI Problem 4.2: Autocorrelation functions from plot

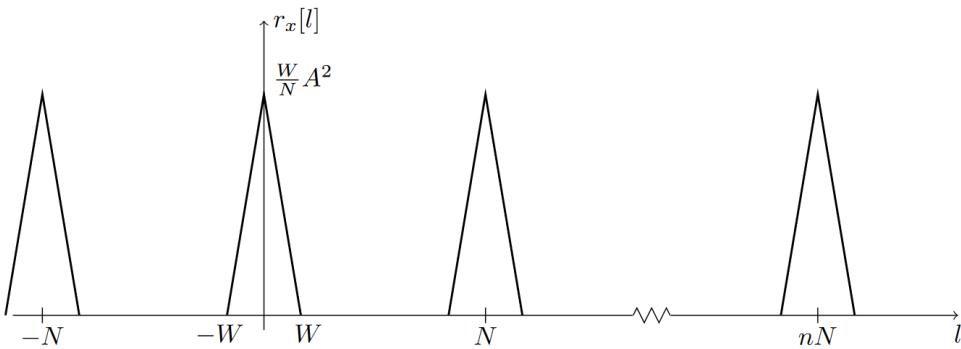
The outcome of a random process is a pulse train with period T and pulse width W as shown in the graph. Let the phase ϕ be uniformly distributed between 0 and T so that the random process is stationary.



1) Sketch and explain what the autocorrelation looks like

Determine, using only simple arguments and drawings, what the autocorrelation function $R_{XX}(\tau)$ will look like.

First of all, we note the random process is periodic. This periodicity is retained in the autocorrelation. The autocorrelation for a stationary function is defined by $r_l[l] = E(x[n]x[n-l])$. For $l = 0$ we therefore get $r_l[0] = E(x[n]x[n])$. When we multiply the signal with itself we see that throughout one period we get A^2 in a fraction $\frac{W}{N}$ of the period and 0 in the remaining $1 - \frac{W}{N}$ of the period. Therefore $r_l[0] = \frac{W}{N}A^2$. If $l = nN$ where $n \in \mathbb{Z}$ we get the same result as for $l = 0$. When we increase (or decrease) l away from 0, the overlap between $x[n]$ and $x[n-l]$ linearly decreases until the overlap is zero at $|l| = W$. The overlap between $x[n]$ and $x[n-l]$ remains at zero until $l = N - W$ (or $l = -N + W$). This repeats each time the lag is increased by N . The autocorrelation function therefore looks like an infinite series of triangular spikes.



2) Use computer simulations to verify the above result

```

A = 1;
W = 3; % Pulse width
T = 6; % Period width
N = 10;

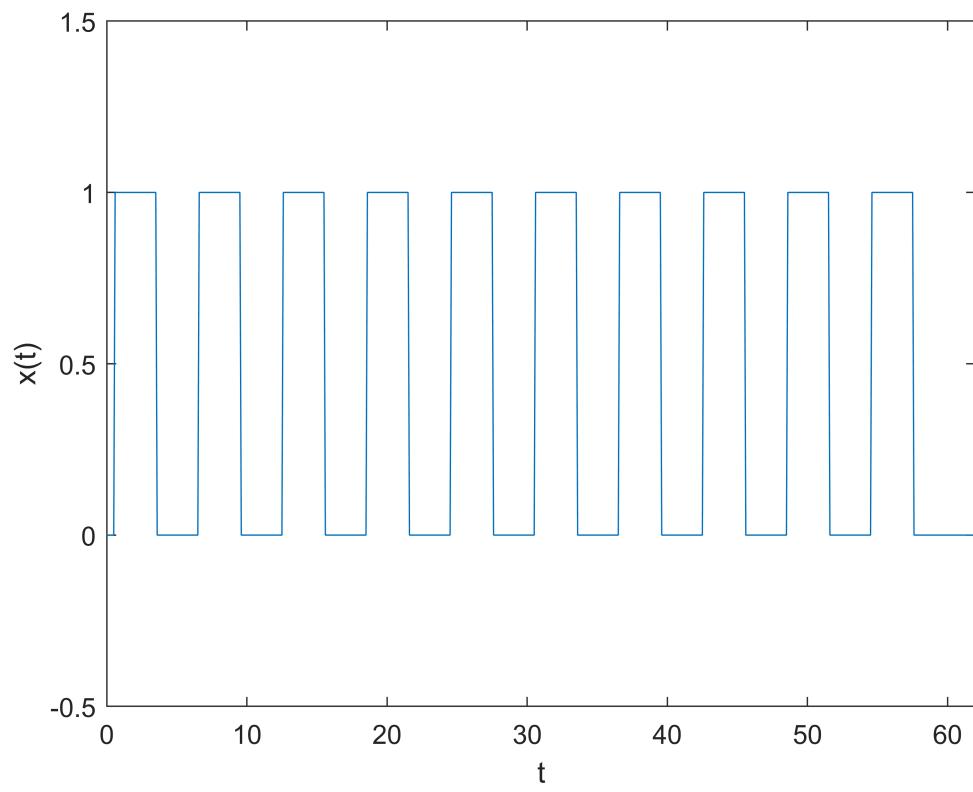
% Random number from a uniform distribution in the open interval (a, b)
% a = 0; b = T; random_number = (b-a)*rand + a
phi = T*rand(1);

maxT = phi+W/2 + T*N;

t=0:.1:maxT; % Time vector
d= phi+W/2:T:phi+W/2+T*N-1; % Delay vector
y = pulstran(t,d,'rectpuls', W);
plot(t,y)

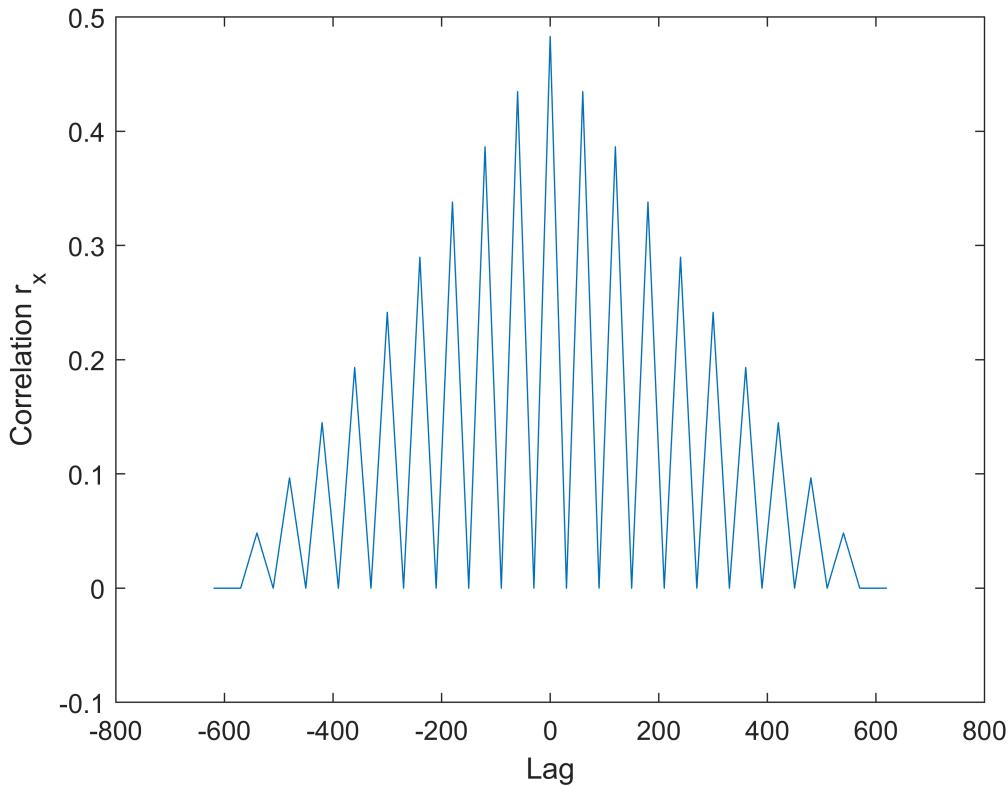
ylabel('x(t)');
xlabel('t');
ylim([-0.5, A+0.5]);
xlim([0, max(t)]);

```



Autocorrelation is symmetrical around zero.

```
[r_xx, lags] = xcorr(y, 'biased');
plot(lags, r_xx);
ylabel('Correlation r_xx');
xlabel('Lag')
```



[✓] ADSI Problem 4.4: Autocorrelation of complex signals

The autocorrelation and crosscorrelation function for complex signals are given by

$$r_x(l) = E[x(n)x^*(n-l)], \quad r_{yx}(l) = E[y(n)x^*(n-l)]$$

Consider a complex sinusoid given by $x(n) = Ae^{j(\omega n + \phi)}$ where A and ω are real constants and ϕ is a random variable with $\phi \sim U(0, 2\pi)$.

1) Compute the analytic expression for the autocorrelation of the complex sinusoid

1. Compute the analytic expression for the autocorrelation of the complex sinusoid.

The autocorrelation function for complex signals can be computed as:

$$r_{xx}(\ell) = E[x(n)x^*(n-\ell)]$$

Plugging the given complex sinusoid into the formula, we get:

$$r_{xx}(\ell) = E[A e^{j(\omega n + \phi)} A e^{-j(\omega(n-\ell) + \phi)}]$$

Since A is a constant, we can move it outside the expected value:

$$r_{xx}(\ell) = A^2 E[e^{j(\omega n + \phi)} e^{-j(\omega(n-\ell) + \phi)}]$$

$$r_{xx}(\ell) = A^2 E[e^{j\omega n + j\phi} e^{-j\omega n + j\omega\ell - j\phi}]$$

$$r_{xx}(\ell) = A^2 E[e^{j\omega n + j\phi + (-j\omega n + j\omega\ell - j\phi)}]$$

$$r_{xx}(\ell) = A^2 E[e^{j\omega n + j\phi - j\omega n + j\omega\ell - j\phi}]$$

$$r_{xx}(\ell) = A^2 E[e^{j\omega\ell}]$$

We know that $E[e^{j\omega\ell}] = e^{j\omega\ell}$ because the expected value of a constant is just the constant itself. Notice that ϕ is no longer in the expression $e^{j\omega\ell}$. Therefore, the autocorrelation of the complex sinusoid is:

$$r_{xx}(\ell) = A^2 e^{j\omega\ell}$$

2) How are $r_x(l)$ and $r_x(-l)$ related for a complex signal?

2. How are $r_x(l)$ and $r_x(-l)$ related when $x(n)$ is a complex-valued signal?

In 1) we found an analytical expression for $r_{xx}(\ell) = A^2 e^{j\omega\ell}$

Let us compute the autocorrelation for $-\ell$:

$$r_{xx}(-\ell) = E[x(n)x^*(n+\ell)]$$

Suppose $m = n + \ell$. Then we can write $n = m - \ell$. Let us substitute all n with m in the above expression:

$$r_{xx}(-\ell) = E[x(m-\ell)x^*(m)]$$

Computing the complex conjugate of the expectation we get:

$$r_{xx}(-\ell) = E[x^*(m-\ell)x(m)]^*$$

$$r_{xx}(-\ell) = E[x(m)x^*(m-\ell)]^*$$

Since $r_{xx}^*(-\ell) = E[x(m)x^*(m-\ell)]^*$, we know that:

$$r_{xx}(-\ell) = r_{xx}^*(\ell)$$

3) Compute the autocorrelation of a real cosine signal

3. Use the above result to compute the autocorrelation of the real signal $x(n) = A \cos(\omega n + \phi)$.

We can use the result from 1)

$$r_{xx}(\ell) = A^2 e^{j\omega\ell}$$

Since the result from 1) uses Euler, we need to convert the signal to complex exponential.

We use the relation $\cos(\theta) = \frac{1}{2}(e^{j\theta} + e^{-j\theta})$

$$x(n) = A \cos(\omega n + \phi)$$

$$x(n) = \frac{A}{2} (e^{j(\omega n + \phi)} + e^{-j(\omega n + \phi)})$$

$$x(n) = \frac{A}{2} e^{j(\omega n + \phi)} + \frac{A}{2} e^{-j(\omega n + \phi)}$$

In 1) we found that the autocorrelation of a complex sinusoid $x(n) = A e^{j(\omega n + \phi)}$ is $r_{xx}(\ell) = A^2 e^{j\omega\ell}$

Therefore, the autocorrelation of the real signal is:

$$r_{xx}(\ell) = \left(\frac{A}{2}\right)^2 e^{j\omega\ell} + \left(\frac{A}{2}\right)^2 e^{-j\omega\ell}$$

$$r_{xx}(\ell) = \frac{A^2}{4} e^{j\omega\ell} + \frac{A^2}{4} e^{-j\omega\ell}$$

$$r_{xx}(\ell) = \frac{A^2}{4} (e^{j\omega\ell} + e^{-j\omega\ell})$$

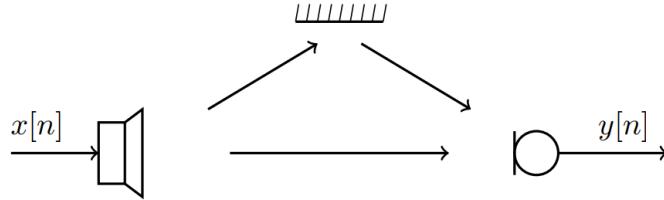
$$r_{xx}(\ell) = \frac{A^2}{2} \frac{1}{2} (e^{j\omega\ell} + e^{-j\omega\ell})$$

Using the relation $\cos(\theta) = \frac{1}{2}(e^{j\theta} + e^{-j\theta})$ we can rewrite the autocorrelation to:

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

[✓] ADSI Problem 4.5: Autocorrelation of distorted signals

An audio signal is sent across a room from a loudspeaker to a microphone as shown in the sketch below and is distorted by a reflection from a surface. The audio signal played in the loudspeaker $x[n]$ is almost white noise, i.e. the autocorrelation function contains only a few non-zero values, all at lags close to zero.



1) Derive autocorrelation function of the output signal

- Derive and sketch the autocorrelation function of the $y[n]$ signal recorded at the microphone. Account for any assumptions and simplifications used in the derivation.

From the figure, we observed that the signal at microphone $y(n)$ is a combination of two signals.

The bottom arrow describes the signal from the loudspeaker that reaches the microphone with no distortion and no delays.

The two top arrows describes the signal that is distorted by a reflection. This means that the original signal $x(n)$ will be attenuated and delayed.

Therefore, we can describe the signal at the microphone as

$$y(n) = x(n) + \alpha x(n - D)$$

where

- $\alpha < 1$ is the attenuation caused by the reflection
- D is the delay of the original signal

The autocorrelation of $y(n)$ can be derived:

$$r_{yy}(\ell) = E[y(n)y(n - \ell)]$$

$$r_{yy}(\ell) = E[(x(n) + \alpha x(n - D))(x(n - \ell) + \alpha x(n - D - \ell))]$$

Multiple each terms

$$r_{yy}(\ell) = E[x(n)x(n - \ell) + \alpha x(n)x(n - D - \ell) + \alpha x(n - D)x(n - \ell) + \alpha^2 x(n - D)x(n - D - \ell)]$$

Separate expectation terms

$$r_{yy}(\ell) = E[x(n)x(n - \ell)] + E[\alpha x(n)x(n - D - \ell)] + E[\alpha x(n - D)x(n - \ell)] + E[\alpha^2 x(n - D)x(n - D - \ell)]$$

Move constants out of the expectation:

$$r_{yy}(\ell) = E[x(n)x(n - \ell)] + \alpha E[x(n)x(n - D - \ell)] + \alpha E[x(n - D)x(n - \ell)] + \alpha^2 E[x(n - D)x(n - D - \ell)]$$

We will use the relation $r_{xx}(\ell) = E[x(n)x(n - \ell)]$ to rewrite the expression above.

The expectation in the first term and the last term are trivial to rewrite:

$$r_{yy}(\ell) = r_{xx}(\ell) + \alpha E[x(n)x(n - D - \ell)] + \alpha E[x(n - D)x(n - \ell)] + \alpha^2 r_{xx}(\ell)$$

We know that $r_{xx}(\ell - D) = E[x(n)x(n - \ell - D)]$ so we want rewrite the second term:

$$r_{yy}(\ell) = r_{xx}(\ell) + \alpha r_{xx}(\ell - D) + \alpha E[x(n - D)x(n - \ell)] + \alpha^2 r_{xx}(\ell)$$

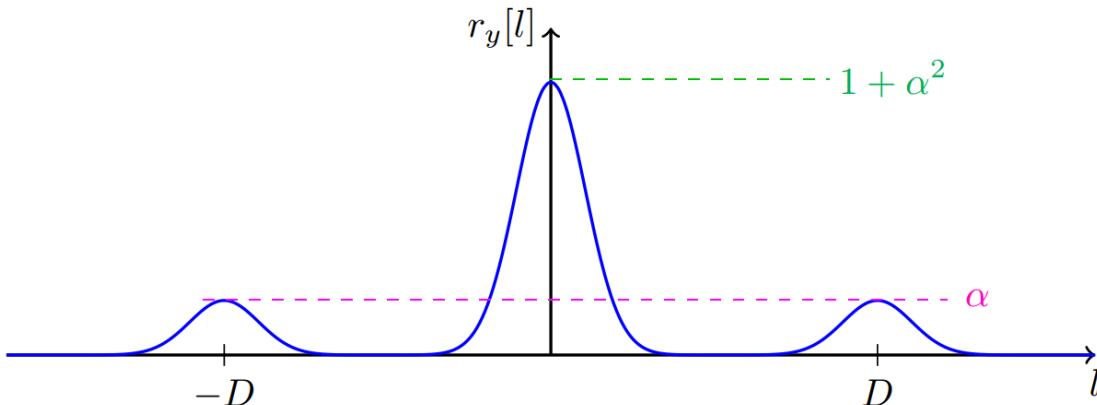
Let $m = n - D$ then $n = m + D$. Then $E[x(m)x(m - \ell + D)]$. Thus, the third term can rewritten to:

$$r_{yy}(\ell) = r_{xx}(\ell) + \alpha r_{xx}(\ell - D) + \alpha r_{xx}(\ell + D) + \alpha^2 r_{xx}(\ell)$$

To make it easier to sketch the autocorrelation $r_{yy}(\ell)$, we can rewrite it the expression as:

$$r_{yy}(\ell) = (1 + \alpha^2)r_{xx}(\ell) + \alpha r_{xx}(\ell - D) + \alpha r_{xx}(\ell + D)$$

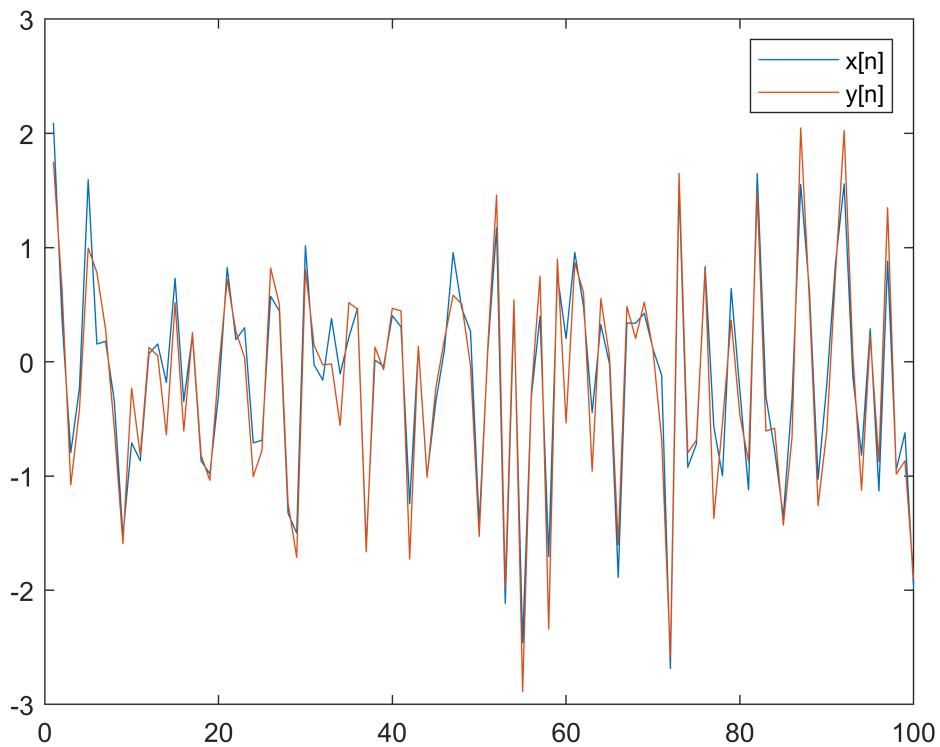
From this expression, we know that there are three peaks because each autocorrelation r_{xx} corresponds to a peak. One peak is at lag zero, another peak at lag $-D$ and the last peak at D . Therefore,



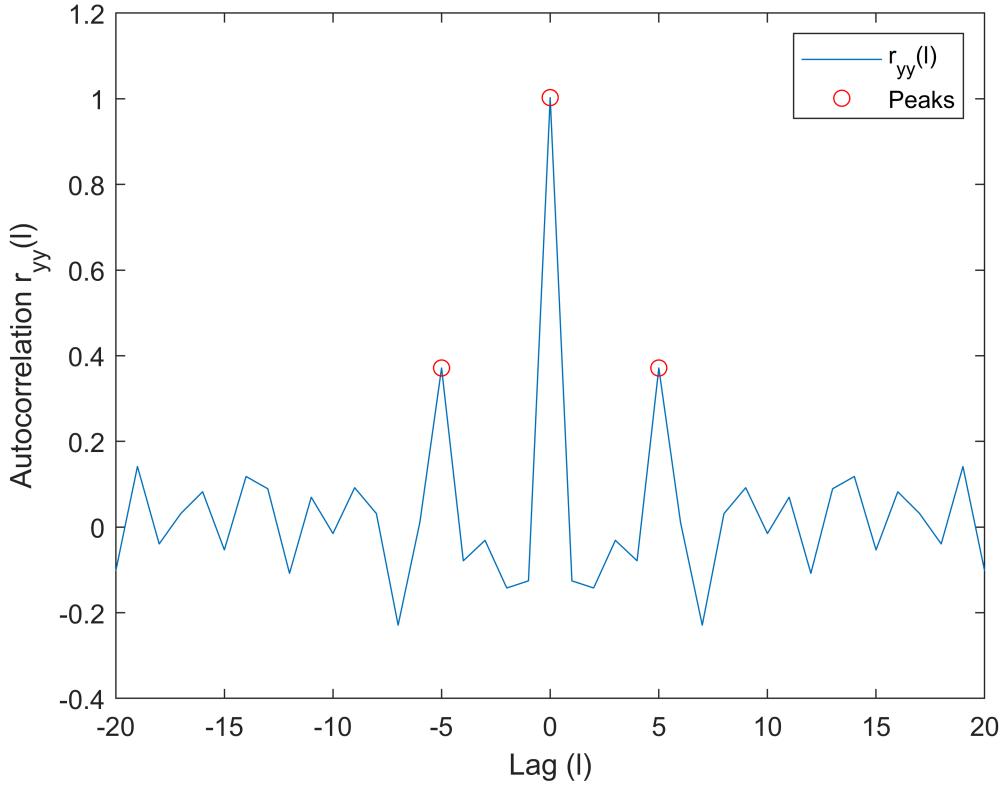
Let us check if we can see the same patterns if we simulate it in MATLAB:

```
alpha = 0.3;
D = 5;
N = 100;

n = [1:N];
x = wgn(N,1,0);
y = x + alpha*circshift(x, D);
plot(n, x, n, y)
legend('x[n]', 'y[n]')
```



```
[r_yy, lags] = xcorr(y, 20, 'biased');
plot(lags, r_yy);
hold on;
plot(0, r_yy(find(lags == 0)), 'ro');
plot(D, r_yy(find(lags == D)), 'ro');
plot(-D, r_yy(find(lags == -D)), 'ro');
legend('r_{yy}(1)', 'Peaks')
ylabel('Autocorrelation r_{yy}(1)');
xlabel('Lag (1)')
hold off;
```



2) Is possible to remove the distortion from the output signal?

2. Is it possible to remove the distortion from the $y[n]$ signal?

The distortion corresponds to a FIR filter with the system function $H(z) = 1 + \alpha z^{-D}$. In principle, we can remove this distortion with an inverse system:

$$\frac{1}{1 + \alpha z^{-D}}$$

However, if we want to implement such an inverse system, we have to build IIR filter.

We might not want to do that. The poor man's approach is as follows:

From our sketch, we can estimate the delay and attenuation by the position of the peaks and the amplitude ratio.

Then we create a FIR filter which subtract a delayed and attenuated copy of $y[n]$:

$$z[n] = y[n] - \alpha y[n - D]$$

$$z[n] = x[n] + \alpha x[n - D] - \alpha(x[n - D] + \alpha x[(n - D) - D])$$

$$z[n] = x[n] + \alpha x[n-D] - \alpha(x[n-D] + \alpha x[n-2D])$$

$$z[n] = x[n] + \alpha x[n-D] - \alpha x[n-D] - \alpha^2 x[n-2D]$$

$$z[n] = x[n] - \alpha^2 x[n-2D]$$

The result is a new signal $z[n]$ where the distortion is pushed to $2D$ and the attenuation factor is squared.

Since $\alpha < 1$, the squared attenuation α^2 is much smaller than 1. This means the distortion component is much smaller than before.

ADSI Problem 4.6: System identification using ARMA(2,2) as input

Consider a signal $x[n]$ be the output of an ARMA(2,2) process with unit-variance, white Gaussian noise as input.

$$x[n] = 2w[n] + 1w[n-1] - x[n-1] - \frac{1}{2}x[n-2]$$

The signal is passed through a FIR filter with impulse response $h[n] = \{3, -1, 1\}$ to give $y[n]$.

```
clear variables;
clear figure;
```

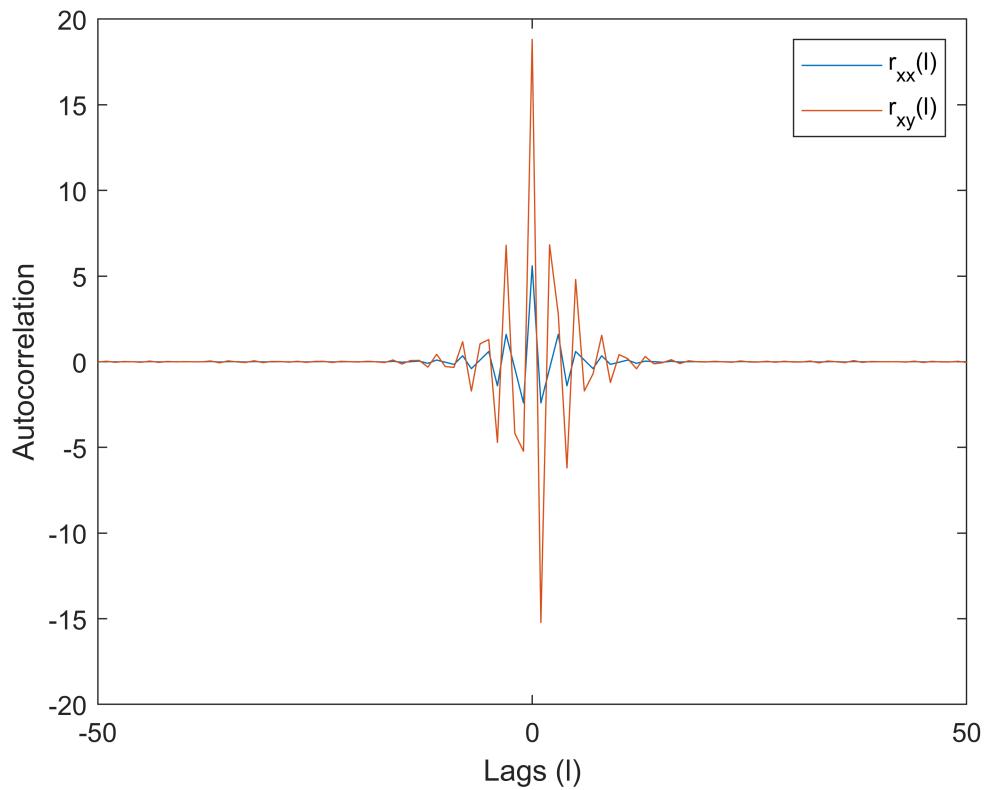
1) Simulate the autocorrelation and cross-correlation in MATLAB

1. Simulate $r_{xx}[l]$ and $r_{yx}[l]$ in MATLAB.

```
w=randn(1e6,1);
x=filter([2 1],[1 1 0.5],w);
y=filter([3 -1 1],1,x);

[rxx,lagssrxx]=xcorr(x,50,'biased');
[ryx,lagssryx]=xcorr(y,x,50,'biased');

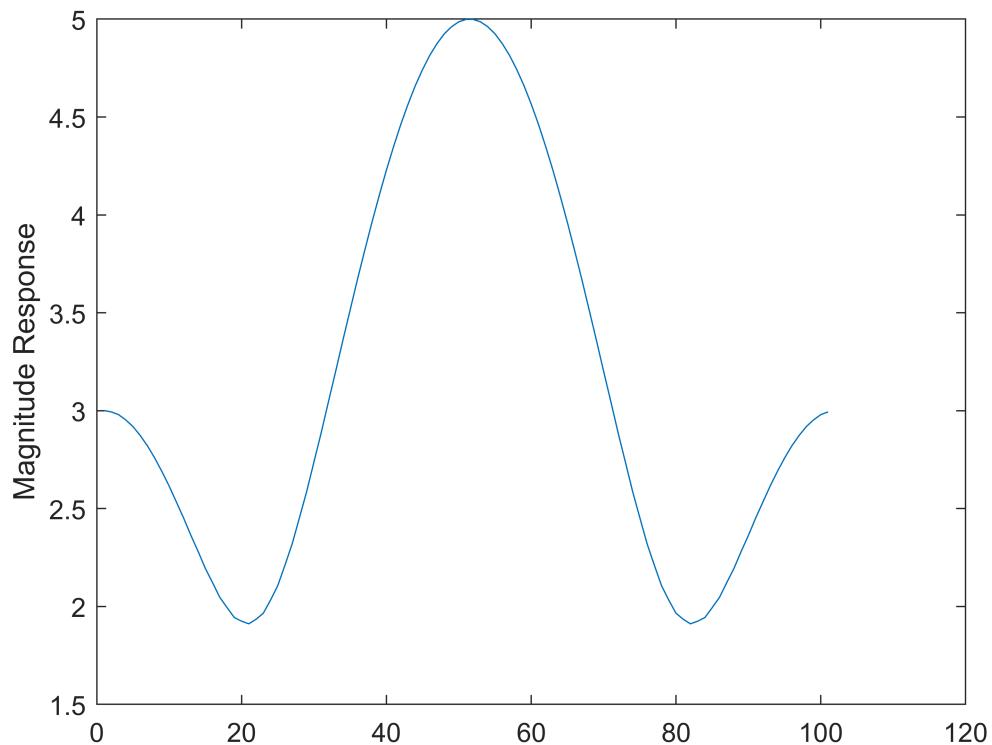
plot(lagssrxx, rxx, lagssryx, ryx)
xlabel('Lags (1)')
ylabel('Autocorrelation')
legend('r_{xx}(1)', 'r_{xy}(1)')
```



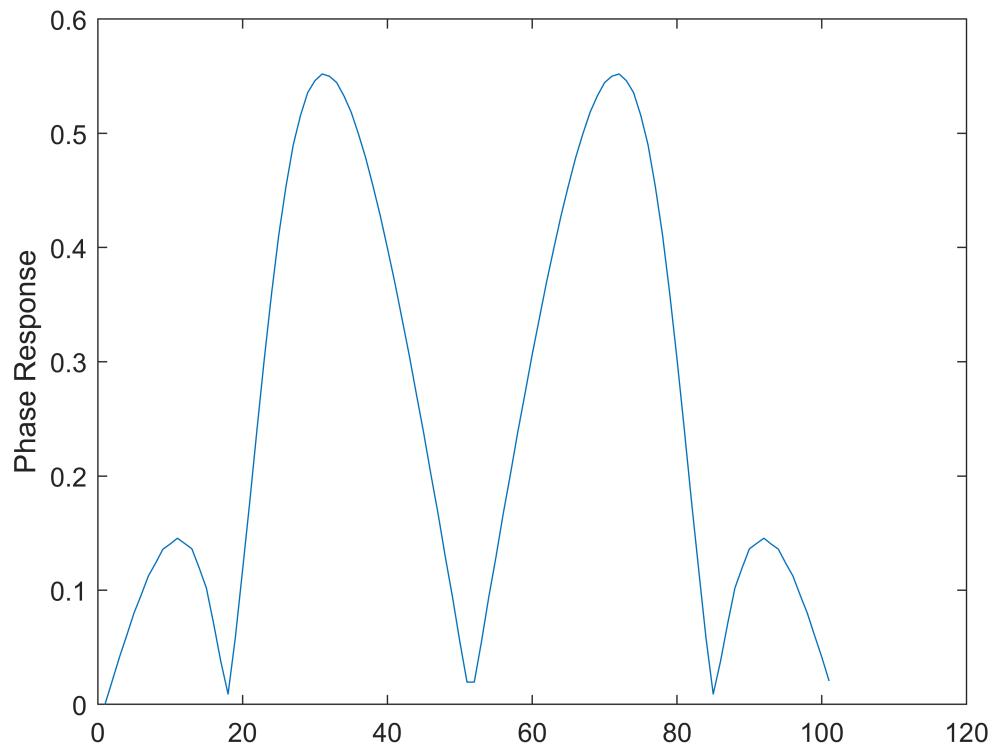
2) Compute the transfer function and impulse response from the autocorrelation

2. Compute $H(\omega)$ and $h[n]$ from $r_{xx}[l]$ and $r_{yx}[l]$ and compare with the analytical results.

```
Rxx=fft(rxx);
Ryx=fft(ryx);
H=Ryx./Rxx;
plot(abs(H))
ylabel('Magnitude Response')
```

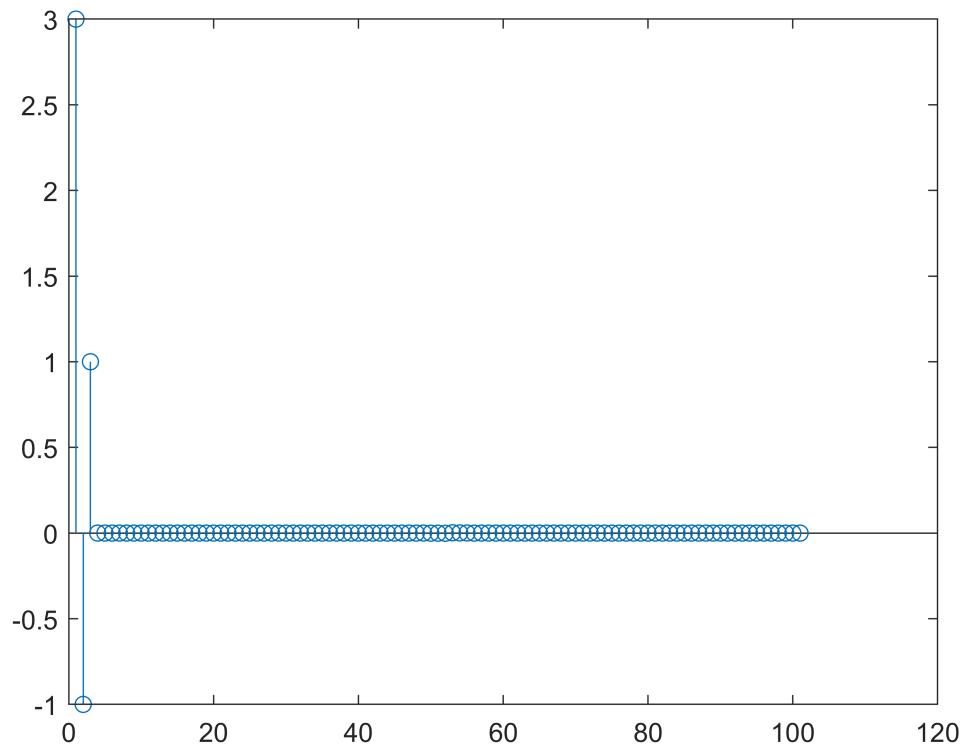


```
plot(abs(angle(H)))
ylabel('Phase Response')
```



Compute the impulse response:

```
h = ifft(H);
stem(real(h))
```



Hilbert Transform

Table of Contents

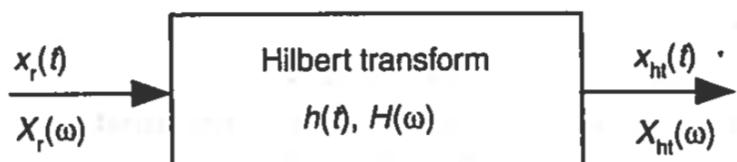
What is Hilbert Transform?	1
Why do we need it?	2
Real Signals as Complex Exponentials.....	2
Quarter Cycle Shift.....	3
Table of selected Hilbert transforms.....	4
Properties of the Hilbert transform.....	6
Envelope Extraction using Hilbert transform.....	8
FFT-based Hilbert transform.....	9
Is the Hilbert transform of a triangular wave signal is another triangular wave signal?.....	13
ADSI Problem 7.1: Study the hilbert function in MATLAB.....	15
ADSI Problem 7.2: Hilbert transform of a cosine signal.....	15
[✓] ADSI Problem 7.3: One-sided spectrum of analytical signals.....	15
[✓] ADSI Problem 7.4: Swept sine wave.....	17
[✓] 1. Design swept sine signal.....	18
[✓] 2. Compute instantaneous frequency.....	19
[?] ADSI Problem 7.5: Envelope Detection.....	22
[?] 1. Plot the signal and envelope on the same graph.....	22
[✓] 2. Describe how noise affects Hilbert transform's ability to recover the envelope.....	24
[✓] 3. Try with other two signals.....	26
[»] ADSI Problem 7.8: Amplitude modulated signal.....	28
[✓] 1. Plot 1 second of the signal and confirm that it is an amplitude modulated signal with a carrier frequency of 20 kHz.....	28
[»] 2. Use the procedure from Figure 9.8 in the note to frequency shift the signal to 15 kHz. Plot the spectrum of the signal at each step in the procedure.....	29
Functions.....	29

What is Hilbert Transform?

The discrete Hilbert transform is a process used to generate complex-valued signals from real-valued signals.

It is a method for extracting the magnitude and phase information from a real-valued signal like a EEG data:

Notation used to define the continuous Hilbert transform:



$x_r(t)$ = a real continuous time-domain input signal
 $h(t)$ = the time impulse response of a Hilbert transformer
 $x_{ht}(t)$ = the HT of $x_r(t)$, ($x_{ht}(t)$ is also a real time-domain signal)
 $X_r(\omega)$ = the Fourier transform of real input $x_r(t)$
 $H(\omega)$ = the frequency response (complex) of a Hilbert transformer
 $X_{ht}(\omega)$ = the Fourier transform of output $x_{ht}(t)$
 ω = continuous frequency measured in radians/second
 t = continuous time measured in seconds.

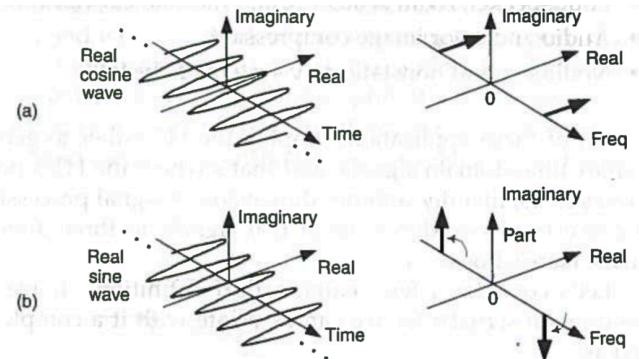


Figure 9-3 The Hilbert transform: (a) $\cos(\omega t)$; (b) its transform is $\sin(\omega t)$.

Why do we need it?

Typically, we start with a real-valued signal which can be conceptualised as:

$$x_r(t) = A \cos(\omega t)$$

Given a real-value signal, we cannot estimate or extract the phase angle and the power. What we need to represent the signal as a complex sinusoid using Euler's formula:

$$x_c(t) = A e^{j\omega t}$$

$$x_c(t) = A \cos(\omega t) + j A \sin(\omega t)$$

$$x_c(t) = x_r(t) + j x_i(t)$$

where

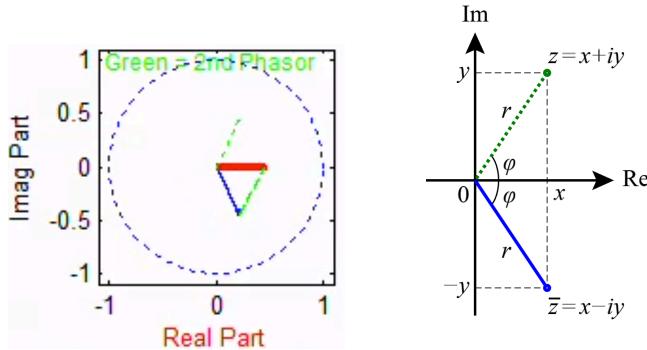
- $x_c(t)$ is known as an *analytic signal* because it has no negative-frequency spectral components. This means that $X_c(e^{j\omega})$ is zero over the negative frequency range. This is also known as a *one-sided spectrum*.
- $j x_i(t)$ is called the *phase-quadrature component*.

Using complex signals of the real signals simplifies and improves the performance of many signal processing operations.

Real Signals as Complex Exponentials

A real-valued signal can be seen as a single frequency phasor (red) that is composed of the sum of two phasors:

- a positive frequency phasor (blue) rotates counter-clockwise
- a negative frequency phasor (green) rotates clockwise



If we project the sum of the two phasors in the real part, we get the red phasor. However, if we project the sum into the imaginary part, the two phasors cancel out one another and we get zero imaginary component.

This means that a real-valued signal has two complex exponentials (complex conjugate phasors) rotating in opposite directions. The real parts are in phase and sum together, the imaginary parts are opposite polarity, and cancel out, which leaves only a real sinusoid as a result. The negative and positive frequency components **are** both necessary to produce the real signal.

A real cosine signal $x_r(t) = A \cos(\omega_0 t + \phi)$ can be expressed in terms of two complex exponentials using Eulers formula:

$$x_r(t) = A \cos(\omega_0 t + \phi)$$

$$x_r(t) = A \left(\frac{e^{j(\omega_0 t + \phi)} + e^{-j(\omega_0 t + \phi)}}{2} \right)$$

$$x_r(t) = A \left(\frac{1}{2} e^{j(\omega_0 t + \phi)} + \frac{1}{2} e^{-j(\omega_0 t + \phi)} \right)$$

$$x_r(t) = A \left(\frac{1}{2} e^{j\phi} e^{j\omega_0 t} + \frac{1}{2} e^{-j\phi} e^{-j\omega_0 t} \right)$$

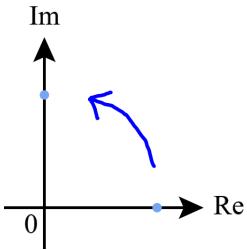
$$x_r(t) = \frac{1}{2} X e^{j\omega_0 t} + \frac{1}{2} X^* e^{-j\omega_0 t} \text{ where } X = A e^{j\phi} \text{ and } X^* = A e^{-j\phi}$$

where X^* is complex conjugate of the phasor X . This shows that a real-valued signal with frequency ω is actually composed for a positive frequency $+\omega$ complex exponential and a negative frequency $-\omega$ complex exponential.

Quarter Cycle Shift

The phase-quadrature component can be obtained by a 90 degree (*quarter-cycle*) shift on the complex plan.

The Hilbert transform of $\cos(\theta)$ is $\sin(\theta)$ i.e., a quarter-cycle shift.



The Hilbert transform rotates the Fourier coefficients in the complex space which converts the real components into the imaginary component. The quarter-cycle shifted signal is then added to the original signal.

In other words, a Hilbert transform takes each frequency component present in the original signal and shifts its phase by $-\frac{\pi}{2}$.

For a continuous time signal $x_r(t)$, the Hilbert transform is defined as:

$$x_{ht}(t) = x_r(t) * \frac{1}{\pi t} \text{ where } * \text{ is the convolution operation}$$

We can see the Hilbert transform as a FIR filter with the impulse response $h(t) = \frac{1}{\pi t}$

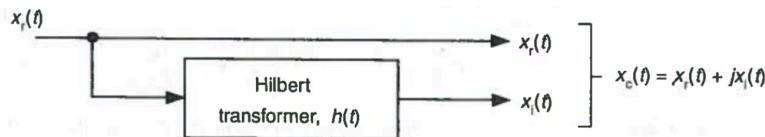


Figure 9-4 Functional relationship between the $x_c(t)$ and $x_r(t)$ signals.

Table of selected Hilbert transforms

In the following table, the frequency parameter ω is real.

Signal	Hilbert transform ^[fn 1]
$u(t)$	$H(u)(t)$
$\sin(\omega t)$ [fn 2]	$\operatorname{sgn}(\omega) \sin\left(\omega t - \frac{\pi}{2}\right) = -\operatorname{sgn}(\omega) \cos(\omega t)$
$\cos(\omega t)$ [fn 2]	$\operatorname{sgn}(\omega) \cos\left(\omega t - \frac{\pi}{2}\right) = \operatorname{sgn}(\omega) \sin(\omega t)$
$e^{i\omega t}$	$\operatorname{sgn}(\omega) e^{i\left(\omega t - \frac{\pi}{2}\right)} = -i \cdot \operatorname{sgn}(\omega) e^{i\omega t}$
$\frac{1}{t^2 + 1}$	$\frac{t}{t^2 + 1}$
e^{-t^2}	$2\pi^{-1/2} F(t)$ (see Dawson function)
Sinc function $\frac{\sin(t)}{t}$	$\frac{1 - \cos(t)}{t}$
Rectangular function $\operatorname{rect}(t) = \Pi(t) = \begin{cases} 0, & \text{if } t > \frac{1}{2} \\ \frac{1}{2}, & \text{if } t = \frac{1}{2} \\ 1, & \text{if } t < \frac{1}{2}. \end{cases}$	$\frac{1}{\pi} \ln \left \frac{t + \frac{1}{2}}{t - \frac{1}{2}} \right $
Dirac delta function $\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$	$\frac{1}{\pi t}$
Characteristic Function $\chi_{[a,b]}(t)$	$\frac{1}{\pi} \ln \left \frac{t - a}{t - b} \right $

Real signal	Hilbert transform
$a_1g_1(t) + a_2g_2(t); a_1, a_2 \in \mathbb{C}$	$a_1\hat{g}_1(t) + a_2\hat{g}_2(t)$
$h(t - t_0)$	$\hat{h}(t - t_0)$
$h(at); a \neq 0$	$\text{sgn}(a)\hat{h}(at)$
$\frac{d}{dt}h(t)$	$\frac{d}{dt}\hat{h}(t)$
$\delta(t)$	$\frac{1}{\pi t}$
e^{jt}	$-je^{jt}$
e^{-jt}	je^{-jt}
$\cos(t)$	$\sin(t)$
$\text{rect}(t)$	$\frac{1}{\pi} \ln (2t+1)/(2t-1) $
$\text{sinc}(t)$	$\frac{\pi t}{2} \text{sinc}^2(t/2) = \sin(\pi t/2) \text{sinc}(t/2)$
$1/(1+t^2)$	$t/(1+t^2)$

Properties of the Hilbert transform

(1) A signal $x_r(t)$ and its Hilbert transform $x_i(t)$ have the **same power density spectrum**.

(2) A signal $x_r(t)$ and its Hilbert transform $x_i(t)$ have the **same autocorrelation function**.

(3) A signal $x_r(t)$ and its Hilbert transform $x_i(t)$ are mutually orthogonal so we can write:

$$\int_{-\infty}^{\infty} x_r(t)x_i(t) dt = 0$$

(4) The effect of applying the Hilbert transform twice on a zero mean signal $x_r(t)$ is:

$$H[H[x_r(t)]] = -x_r(t)$$

For example, suppose $x_r(t) = \cos(t)$. Computing the Hilbert transform we get:

$$H[\cos(t)] = \cos\left(t - \frac{\pi}{2}\right)$$

Computing the Hilber transform again, we get:

$$H\left[\cos\left(t - \frac{\pi}{2}\right)\right] = \cos\left(t - \frac{\pi}{2} - \frac{\pi}{2}\right)$$

$$H\left[\cos\left(t - \frac{\pi}{2}\right)\right] = \cos(t - \pi)$$

$$H\left[\cos\left(t - \frac{\pi}{2}\right)\right] = -\cos(t)$$

(5) The energy of a Hilbert transform $x_i(t)$ is the same as the energy of the original signal $x_r(t)$. Hilbert Transform does not changes the magnitude of the input signal hence the energy is invariant

Assume that some zero-mean signal $x(k)$ with finite energy

$$\sum_{k=-\infty}^{\infty} x^2(k) = C$$

is Hilbert transformed to give $x_{HT}(k) = H[x(k)]$

Example:

$$x(t) \longrightarrow \boxed{h(t) = \frac{1}{\pi t}} \longrightarrow y(t)$$

- If $x(t) = 4\text{sinc}(2t)$, then evaluate $\int_{-\infty}^{\infty} |y(t)|^2 dt$.
- The integral is energy of $y(t)$.
- Also, $y(t)$ is Hilbert transform of $x(t)$.
- Hence, energy of $y(t)$ = energy of $x(t)$.
- Energy of $x(t) = 4\text{sinc}(2t)$, is, $E_x = \frac{4^2}{2} = 8$.

(6) The inverse Hilbert transform is $H^{-1}[x_r(t)] = -H[x_r(t)]$:

$$H(\omega) = \begin{cases} -j & \omega > 0 \\ 0 & \omega = 0 \\ j & \omega < 0 \end{cases} \Rightarrow H_{inv}(\omega) = -H(\omega) = \begin{cases} j & \omega > 0 \\ 0 & \omega = 0 \\ -j & \omega < 0 \end{cases}$$

Note: A DC term can't be recreated

Envelope Extraction using Hilbert transform

```
clear variables;
```

The books states: If a real sinewave $x_r(t)$ is amplitude modulated so its envelope contains information, then we can measure the instantaneous envelope $E(t)$ values from an analytic version of the signal using the Equation 9-4:

$$E(t) = |x_c(t)| = \sqrt{x_r(t)^2 + x_i(t)^2}. \quad (9-4)$$

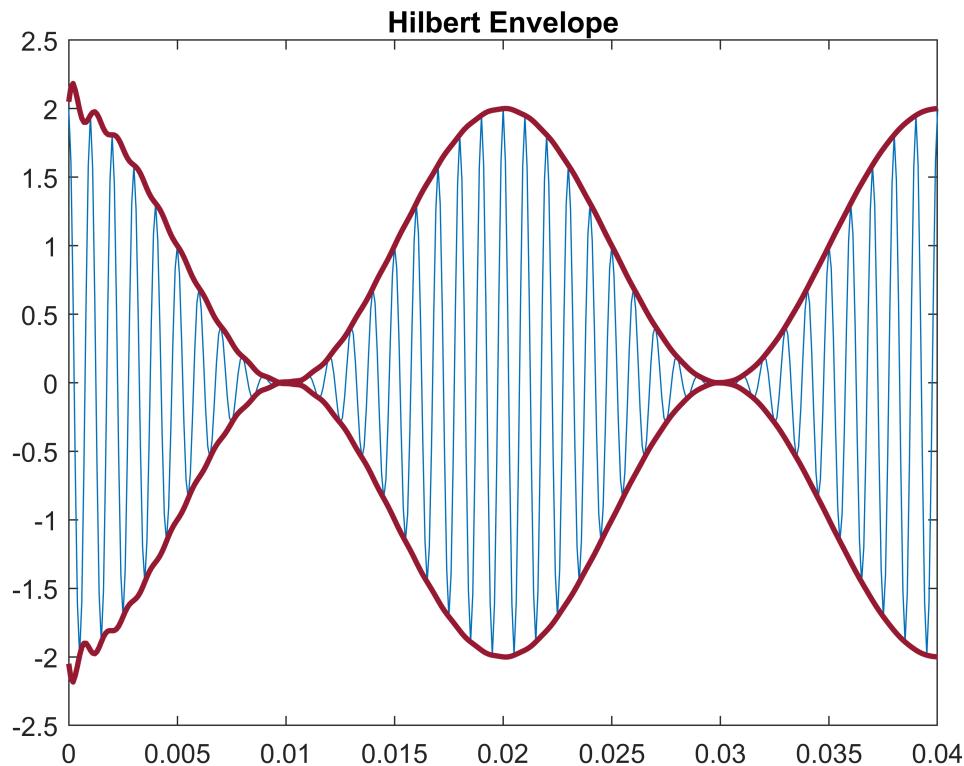
Equation 9-4 states that the envelope of the signal is equal to magnitude of $x_c(t)$.

In MATLAB the complex magnitude of a signal can be computed with the abs function.

```
t = 0:1e-4:0.1;
x = (1+cos(2*pi*50*t)).*cos(2*pi*1000*t);

y = hilbert(x);
env = abs(y);
plot_param = {'Color', [0.6 0.1 0.2], 'Linewidth', 2};

plot(t,x)
hold on
plot(t,[-1;1]*env,plot_param{1})
hold off
xlim([0 0.04])
title('Hilbert Envelope')
```



FFT-based Hilbert transform

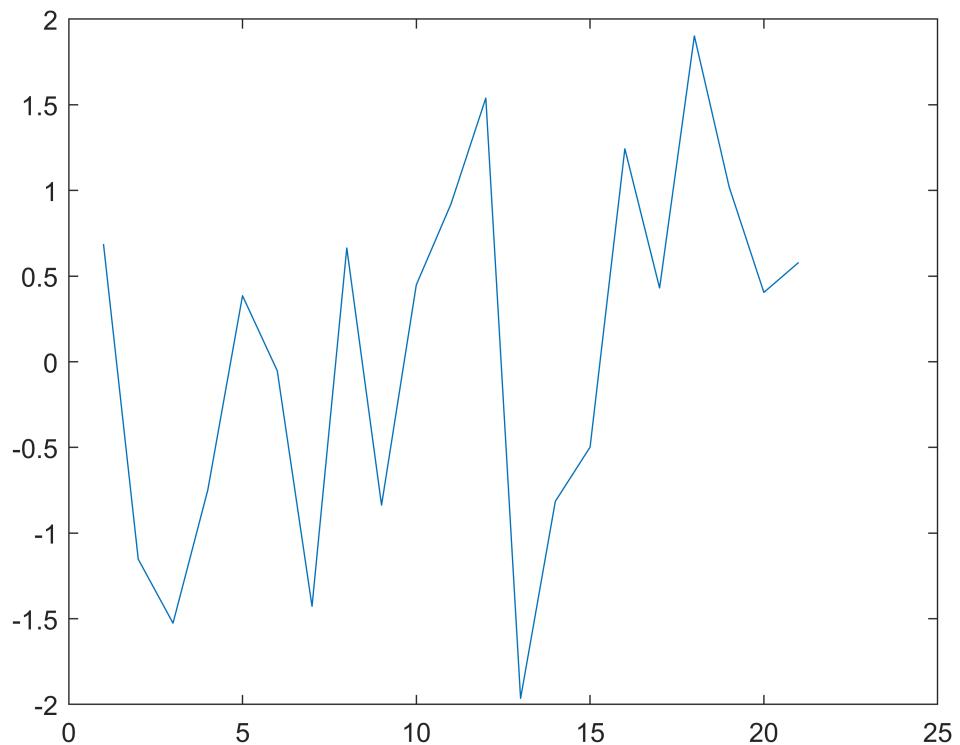
```
clear variables;
```

The FFT-based Hilbert transform can be implemented as three steps:

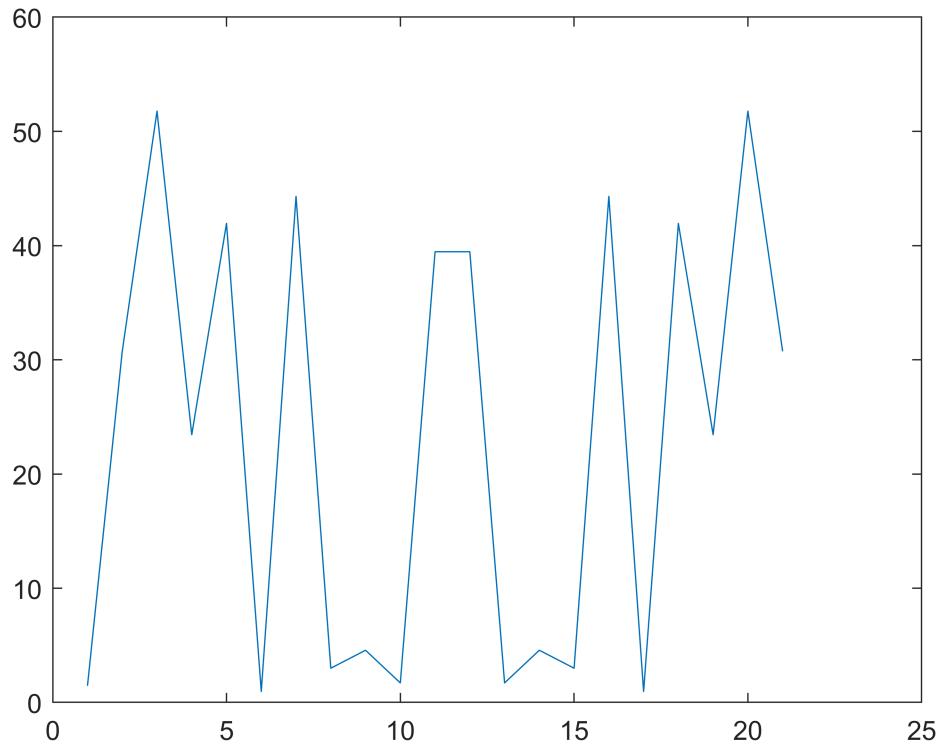
1. Take the FFT of a time-series signal
2. Rotate the Fourier coefficients
3. Take the inverse FFT of the rotated Fourier coefficients to get a time-series signal

```
n = 21;

% Generate a signal
x = randn(n, 1);
plot(x)
```



```
% Step 1: Take the FFT  
f = fft(x);  
plot(f.*conj(f))
```

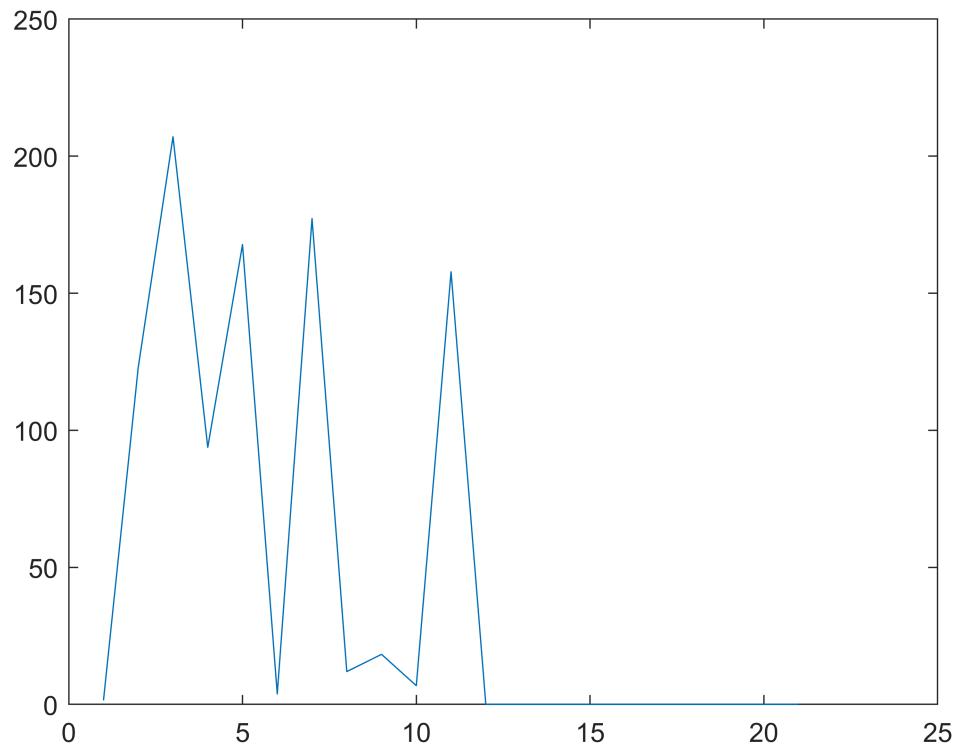


```
% Create a copy of the Fourier coefficients
complexf = 1i*f;

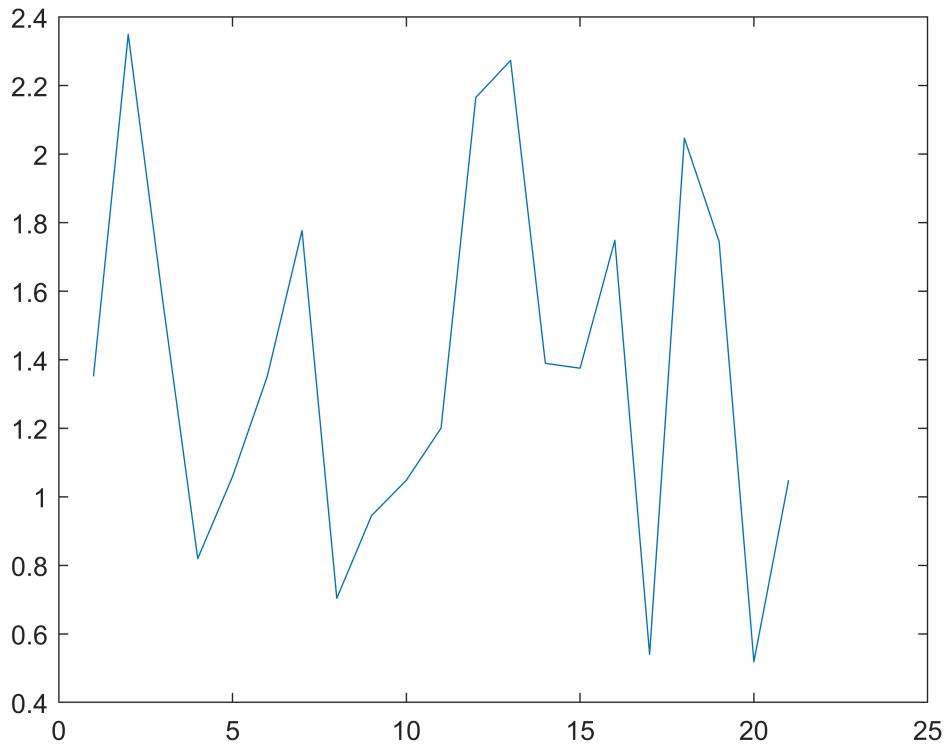
% Identify the indices for the positive and negative frequencies
% because they are rotated in different ways.
pos_freq_idx = 2:floor(n/2) + mod(n, 2);
neg_freq_idx = ceil(n/2) + 1 + ~mod(n, 2):n;

% Rotate the Fourier coefficients by computing i*A*sin(wt) component.
% Note! Positive frequencies are rotated counter-clockwise while
% negative frequencies are rotated clockwise.
f(pos_freq_idx) = f(pos_freq_idx) + -1i*complexf(pos_freq_idx);
f(neg_freq_idx) = f(neg_freq_idx) + 1i*complexf(neg_freq_idx);

plot(f.*conj(f))
```



```
hilbert_x = ifft(f);
plot(abs(hilbert_x))
```



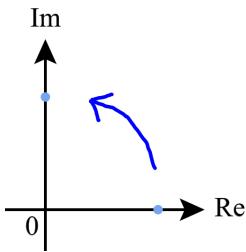
Is the Hilbert transform of a triangular wave signal is another triangular wave signal?

2. The Hilbert transform of a triangular wave signal is another triangular wave signal.

Answer: FALSE.

The Hilbert transform rotates the Fourier coefficients in the complex space which converts the real components into the imaginary component. The quarter-cycle shifted signal is then added to the original signal. In other words, a Hilbert transform takes each frequency component present in the original signal and shifts its phase by $-\frac{\pi}{2}$.

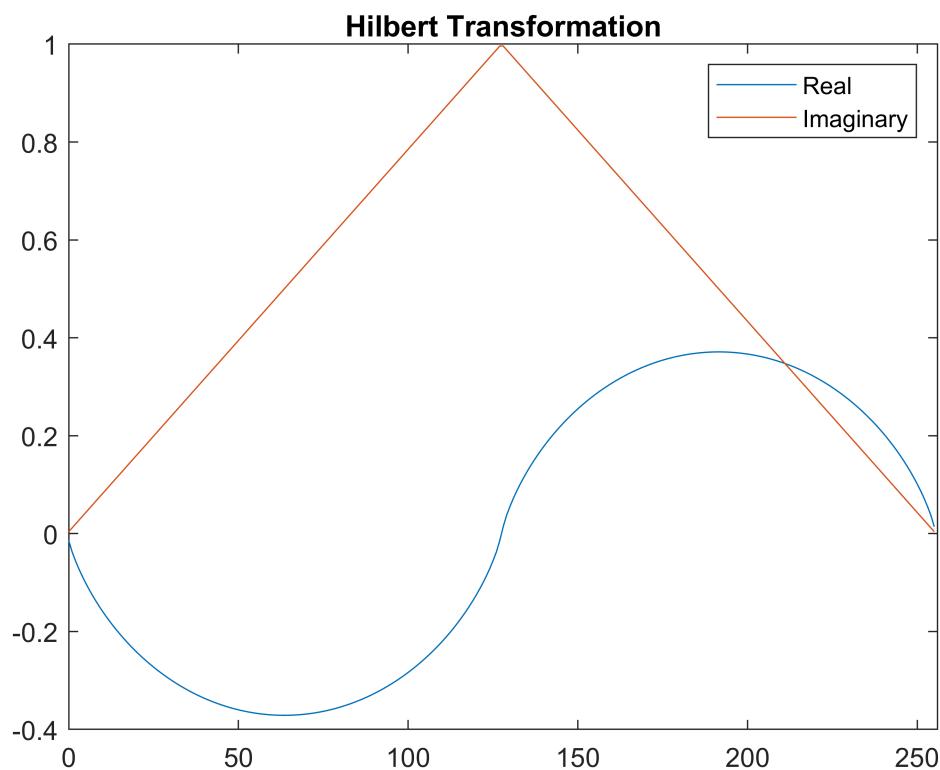
The Hilbert transform of $\cos(\theta)$ is $\sin(\theta)$ i.e., a quarter-cycle shift.



A triangular wave signal can thought of being composed of multiple frequency component. If each of these are shifted by 90 degrees then result is not a triangular wave signal.

We can experiment with it.

```
N = 256;
n = 0:N-1;
xr = triang(256);
x = hilbert(xr);
plot(n, imag(x), n, real(x))
legend('Real','Imaginary')
title('Hilbert Transformation')
xlim([0, N])
```



ADSI Problem 7.1: Study the hilbert function in MATLAB

A tremendous amount of functionality is found in MATLAB's toolboxes and Hilbert transforms are no exception. Type doc hilbert in MATLAB and read the documentation.

```
doc hilbert
```

ADSI Problem 7.2: Hilbert transform of a cosine signal

A continuous time signal is given by $x(t) = 4 + 3\cos(\omega t)$. What is the Hilbert transform of this signal?

Signal $u(t)$	Hilbert transform [fn 1] $H(u)(t)$
$\sin(\omega t)$ [fn 2]	$\text{sgn}(\omega) \sin(\omega t - \frac{\pi}{2}) = -\text{sgn}(\omega) \cos(\omega t)$
$\cos(\omega t)$ [fn 2]	$\text{sgn}(\omega) \cos(\omega t - \frac{\pi}{2}) = \text{sgn}(\omega) \sin(\omega t)$
$e^{i\omega t}$	$\text{sgn}(\omega) e^{i(\omega t - \frac{\pi}{2})} = -i \cdot \text{sgn}(\omega) e^{i\omega t}$
$\frac{1}{t^2 + 1}$	$\frac{t}{t^2 + 1}$
e^{-t^2}	$2\pi^{-1/2} F(t)$ (see Dawson function)
Sinc function $\frac{\sin(t)}{t}$	$\frac{1 - \cos(t)}{t}$
Rectangular function $\text{rect}(t) = \Pi(t) = \begin{cases} 0, & \text{if } t > \frac{1}{2} \\ \frac{1}{2}, & \text{if } t = \frac{1}{2} \\ 1, & \text{if } t < \frac{1}{2}. \end{cases}$	$\frac{1}{\pi} \ln \left \frac{t + \frac{1}{2}}{t - \frac{1}{2}} \right $
Dirac delta function $\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$	$\frac{1}{\pi t}$
Characteristic Function $\chi_{[a,b]}(t)$	$\frac{1}{\pi} \ln \left \frac{t - a}{t - b} \right $

Real signal	Hilbert transform
$a_1 g_1(t) + a_2 g_2(t); a_1, a_2 \in \mathbb{C}$	$a_1 \hat{g}_1(t) + a_2 \hat{g}_2(t)$
$h(t - t_0)$	$\hat{h}(t - t_0)$
$h(at); a \neq 0$	$\text{sgn}(a) \hat{h}(at)$
$\frac{d}{dt} h(t)$	$\frac{d}{dt} \hat{h}(t)$
$\delta(t)$	$\frac{1}{\pi t}$
e^{jt}	$-j e^{jt}$
e^{-jt}	$j e^{-jt}$
$\cos(t)$	$\sin(t)$
$\text{rect}(t)$	$\frac{1}{\pi} \ln (2t + 1)/(2t - 1) $
$\text{sinc}(t)$	$\frac{\pi t}{2} \text{sinc}^2(t/2) = \sin(\pi t/2) \text{sinc}(t/2)$
$1/(1 + t^2)$	$t/(1 + t^2)$

$$H[4 + 3 \cos(\omega t)] = H[4] + 3H[\cos(\omega t)]$$

$$= 0 + 3\text{sgn}(\omega) \sin(\omega t)$$

[✓] ADSI Problem 7.3: One-sided spectrum of analytical signals

```
clear variables;
```

Consider the signal

$$x(n) = e^{-0.001(n-255)^2} \cos(1.8n) \quad 0 \leq n \leq 511.$$

Calculate the analytical signal in MATLAB and check that the spectrum is one-sided as expected.

Recall that we start with a real-valued signal $x_r(t)$ and we want we need to represent the signal as a complex sinusoid:

$$x_c(t) = x_r(t) + j x_i(t)$$

$x_c(t)$ is known as an **analytic signal** because it has no negative-frequency spectral components.

To show that the spectrum is one-sided, we have to show that the frequency response $X_c(e^{j\omega})$ is zero over the negative frequency range.

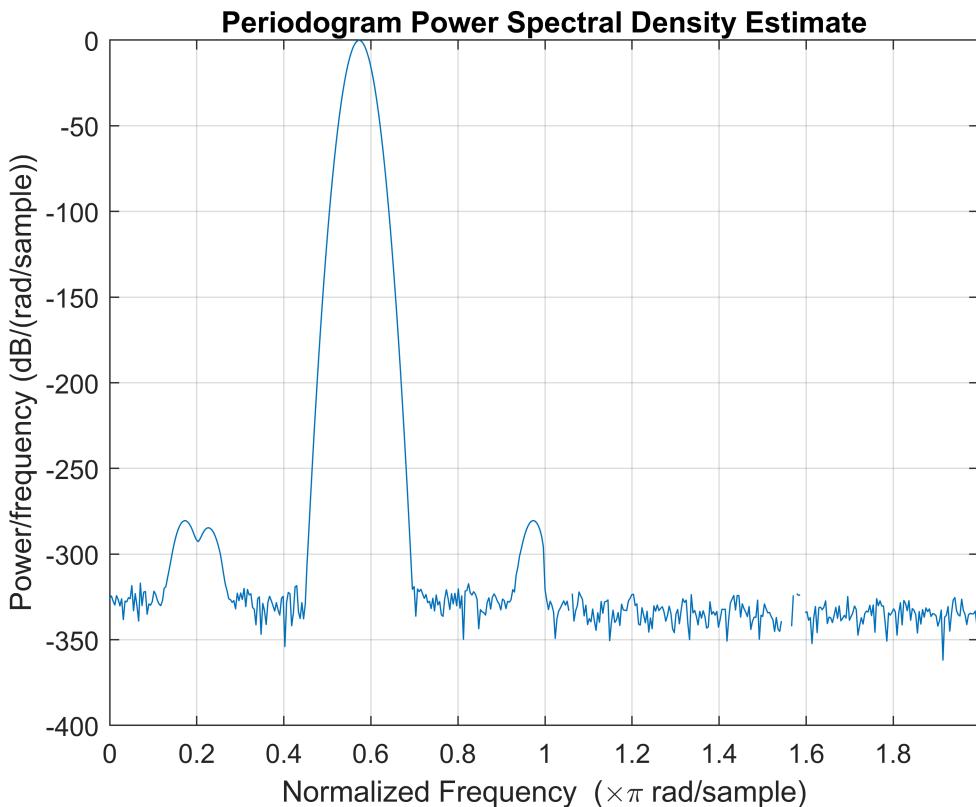
We can do this by estimating the power spectrum density (PSD) of the analytical signal using the periodogram.

In the MATLAB plot, the normalised **positive frequency range** is [0, 1] whereas the **negative frequency range** is between 1 and 2. Notice there is no power spikes in the negative frequency range, only noise.

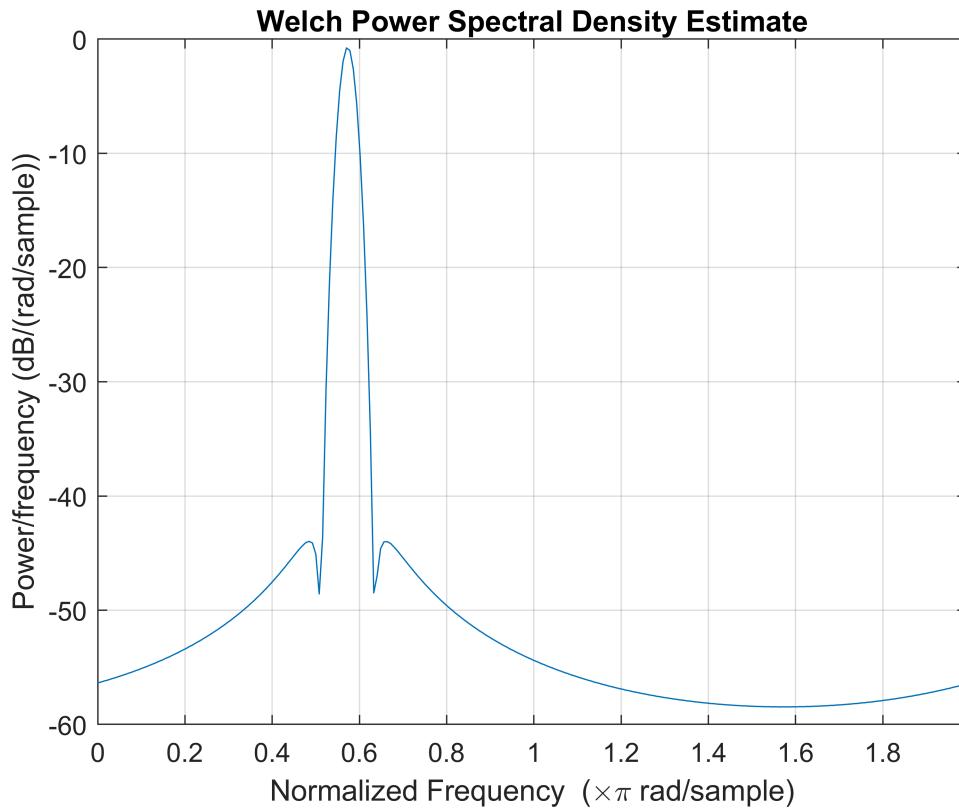
```
n = (0:511)';
x = exp(-0.001.* (n-225).^2).*cos(1.8*n);

% Compute the analytic signal of the real signal x(n)
y = hilbert(x);

% Compute and plot the PSD estimate
periodogram(y)
```



```
% PSD estimate using the Welch method
pwelch(y)
```



[✓] ADSI Problem 7.4: Swept sine wave

In (audio) measurements an exponentially swept sine wave is often used to measure transfer functions and nonlinear distortion, see e.g. Novák et al. “Nonlinear System Identification Using Exponential Swept-Sine Signal” IEEE Trans. Instrum. Measure. **59**, 2220-2229, (2010) or “Synchronized Swept-Sine: Theory, Application and Implementation”. J. Audio Eng. Soc. **63**, 787-798, (2015). In the continuous time domain an exponential swept sine wave is given by

$$s(t) = \sin \left(2\pi f_1 L \left[\exp \left(\frac{t}{L} \right) - 1 \right] \right)$$

where

$$L = \frac{T}{\ln \left(\frac{f_2}{f_1} \right)}$$

T is the duration of the sweep, f_1 and f_2 are the start and stopping frequencies of the sweep.

```
clear variables;
```

[✓] 1. Design swept sine signal

1. Design a 5 second long swept sine sampled at 48 kHz. The sine should sweep from 50 Hz to 5000 Hz. Play the sound on your pc and check that it performs as expected.

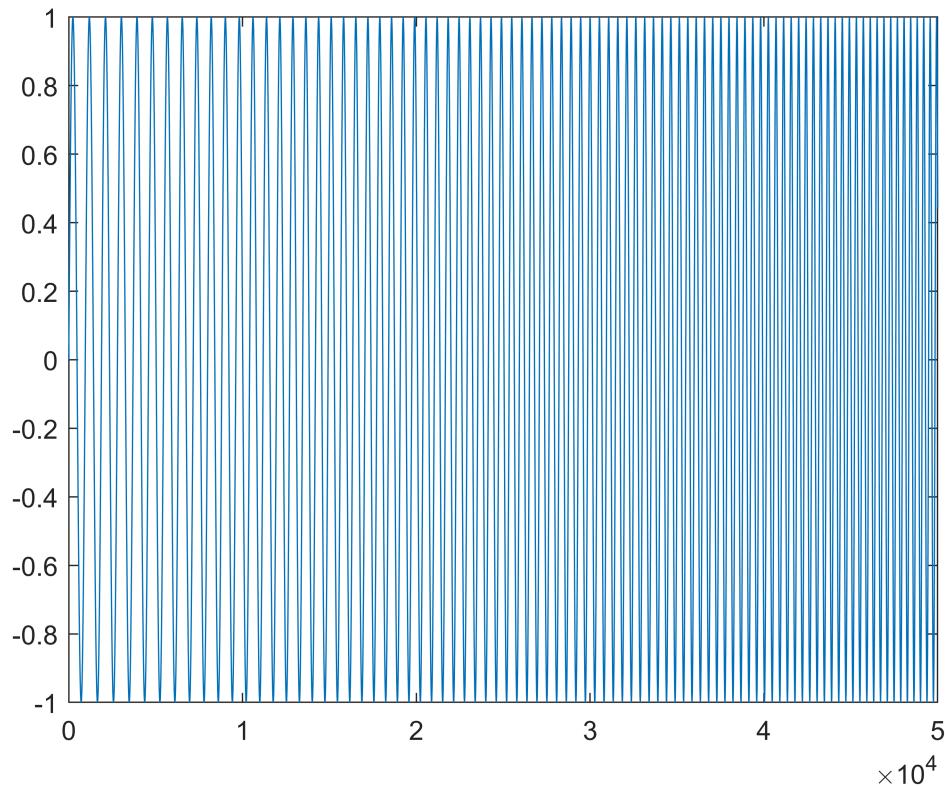
```
Ts = 48000; % Sampling rate
f1 = 50; % Starting 1/frequency
f2 = 5000; % Stopping 1/frequency
T = 5*Ts; % Duration of the sweep
L = T / log(f2/f1);

n = 0:T-1;
s = sin(2*pi * f1 * L * (exp(n/L) - 1) / Ts);

% Write to file
% audiowrite('swept_sine.wav', s, Ts);

% Play sound
soundsc(s, Ts);

% Plot it
plot(n, s)
xlim([0, 50000]);
```



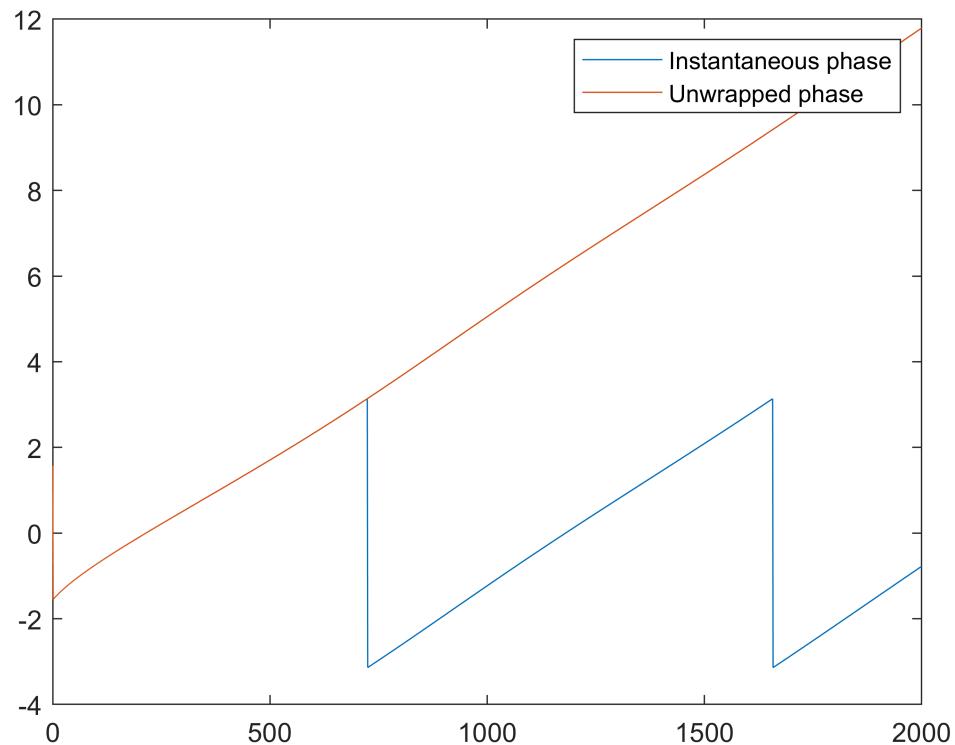
[✓] 2. Compute instantaneous frequency

2. Use equation 9.6 $F(t) = d/dt \tan^{-1} x_i(t)/x_r(t)$ to calculate the instantaneous frequency of the swept sine signal and compare with the real frequency.

$$F(t) = \frac{d}{dt} \phi(t) = \frac{d}{dt} \tan^{-1} \left(\frac{x_i(t)}{x_r(t)} \right). \quad (9-6)$$

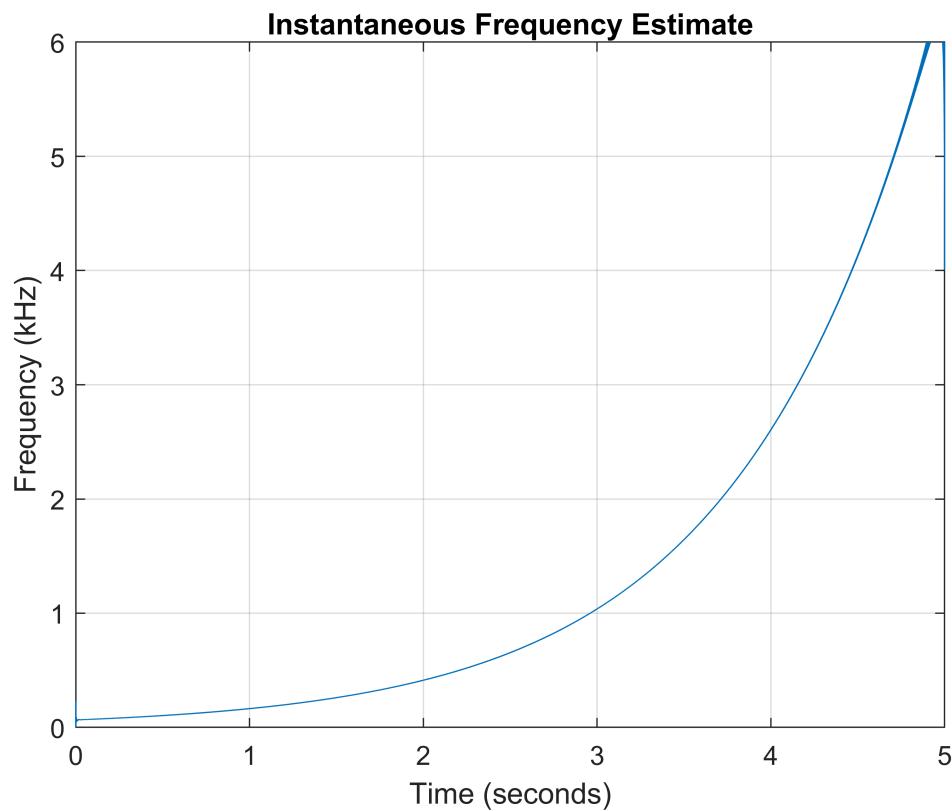
```
s_c = hilbert(s); % Compute the analytic signal

% The instantaneous phase can be computed using MATLAB's angle function
inst_phase = angle(s_c);
plot(n, inst_phase, n, unwrap(inst_phase))
legend('Instantaneous phase', 'Unwrapped phase')
xlim([0, 2000])
```

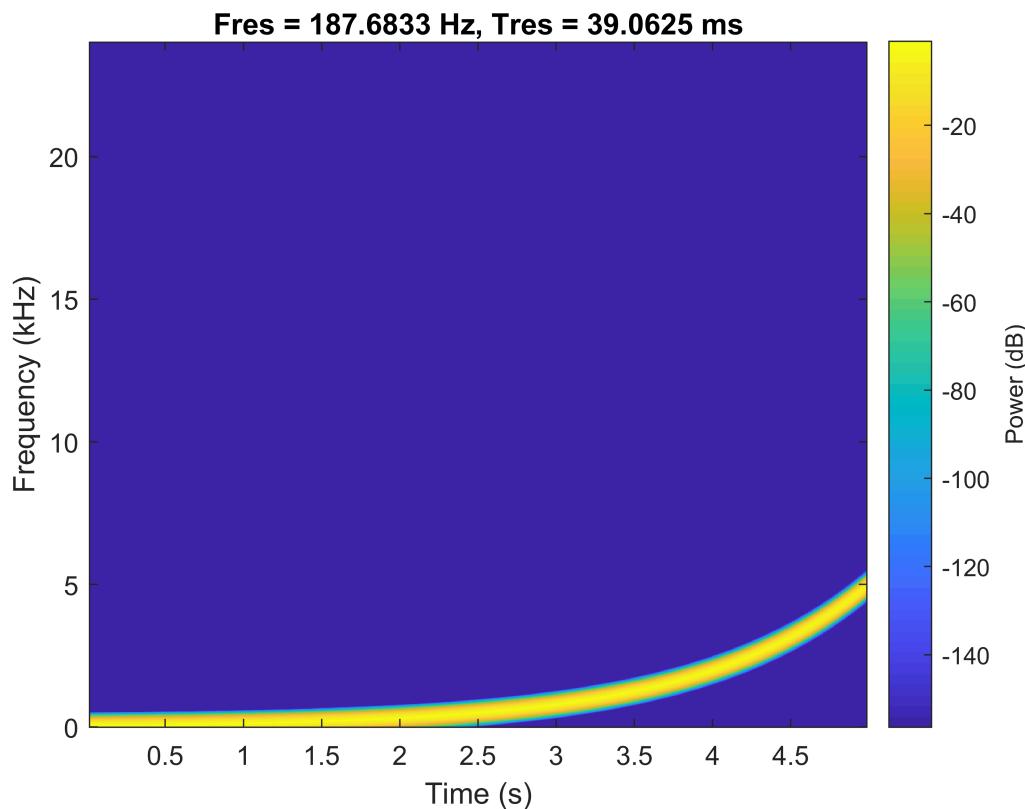


```
% Approximate the derivative of the unwrapped instantaneous phase
F = diff(unwrap(inst_phase));

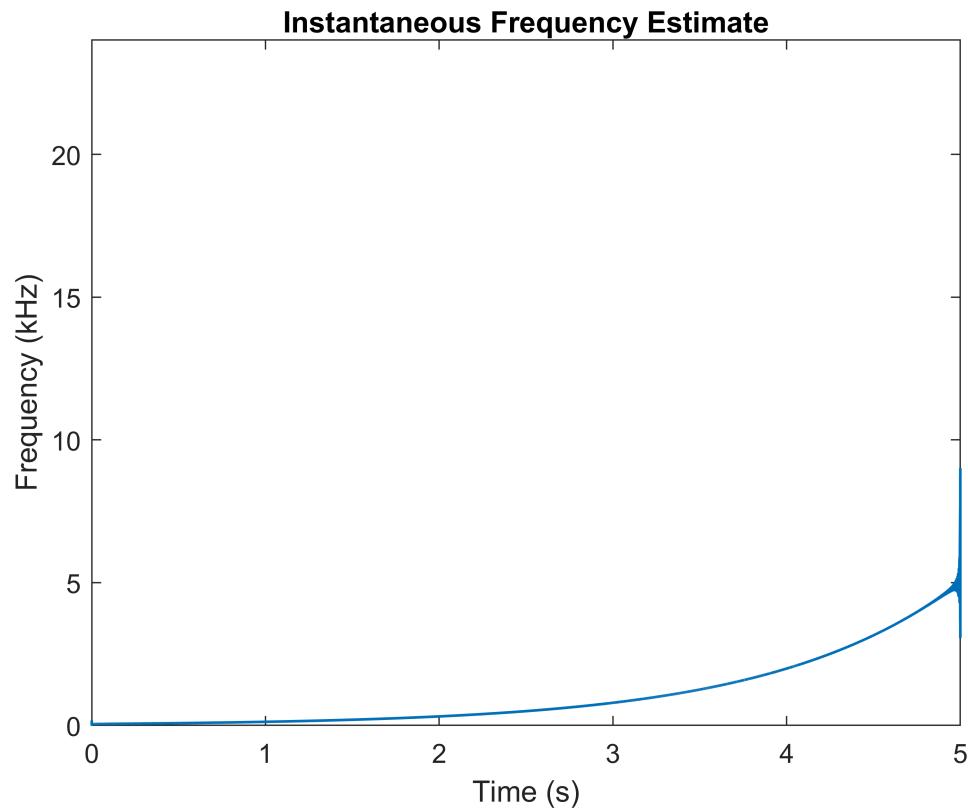
% Plot it
plot(n(2:end)/Ts, F*10)
ylim([0, 6])
grid on
ylabel('Frequency (kHz)')
xlabel('Time (seconds)')
title('Instantaneous Frequency Estimate')
```



```
% Estimate the spectrogram of the waveform using the pspectrum function.  
pspectrum(s, Ts, 'spectrogram')
```



```
% The instfreq function computes and displays the instantaneous frequency in one step.
instfreq(s, Ts, 'Method', 'hilbert')
```



[?] ADSI Problem 7.5: Envelope Detection

The textbook states that the Hilbert transform can be used to find the envelope of a signal. The validity of this statement is investigated in this problem.

Consider an exponentially decaying oscillating signal given by

$$x(n) = 4e^{-0.01n} \cos(\pi n)$$

```
clear variables;
clf
```

[?] 1. Plot the signal and envelope on the same graph

Plot this signal and use equation 9.4 to plot the envelope on the same graph. Compare the result with the expected outcome.

The book states: If a real sinewave $x_r(t)$ is amplitude modulated so its envelope contains information, then we can measure the instantaneous envelope $E(t)$ values from an analytic version of the signal using the Equation 9-4:

$$E(t) = |x_c(t)| = \sqrt{x_r(t)^2 + x_i(t)^2}. \quad (9-4)$$

Equation 9-4 states that the envelope of the signal is equal to magnitude of $x_c(t)$.

In MATLAB the complex magnitude of a signal can be computed with the abs function.

```
n = (0:511);
x_r = 4*exp(-0.001.*n).*cos(pi*n);

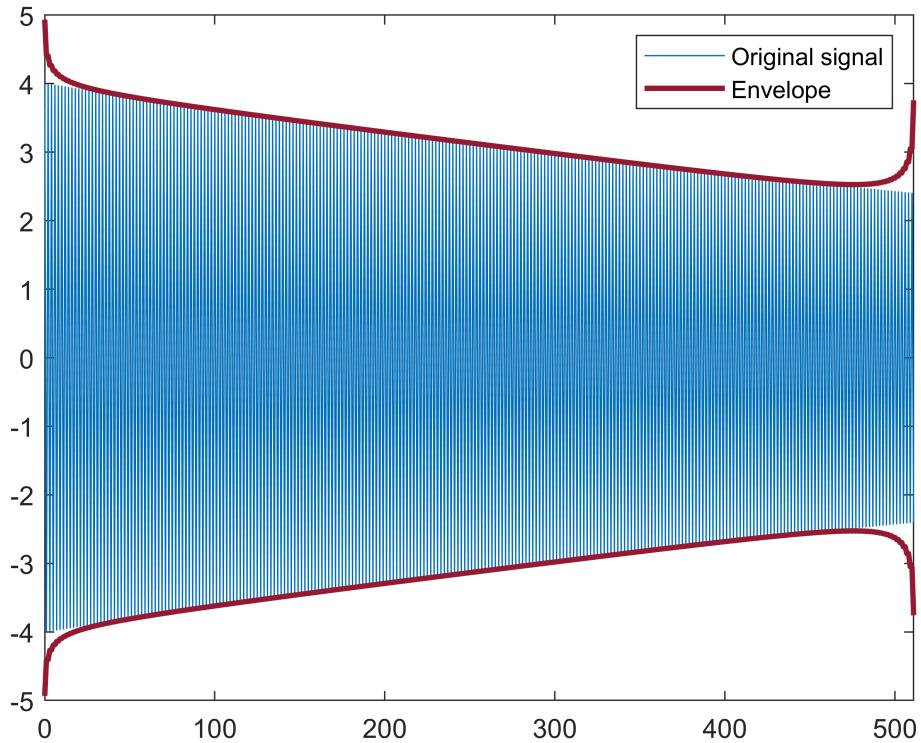
% Compute the analytic signal
x_c = hilbert(x_r);

% Find the envelope by computing the magnitude of the signal
x_envelope = abs(x_c);

% Plot the real signal
plot(n, x_r)
hold on

% Plot the envelope
plot(n, [-1;1]*x_envelope, 'Color', [0.6 0.1 0.2], 'Linewidth', 2)
hold off

legend('Original signal', 'Envelope')
xlim([0, max(n)])
```



[✓] 2. Describe how noise affects Hilbert transform's ability to recover the envelope

In real life noise is always present in measurements.

2. Add white Gaussian noise WGN $\sim (0,1)$ to the signal and repeat. Describe the influence of the noise in the ability of the Hilbert transform to recover the envelope.

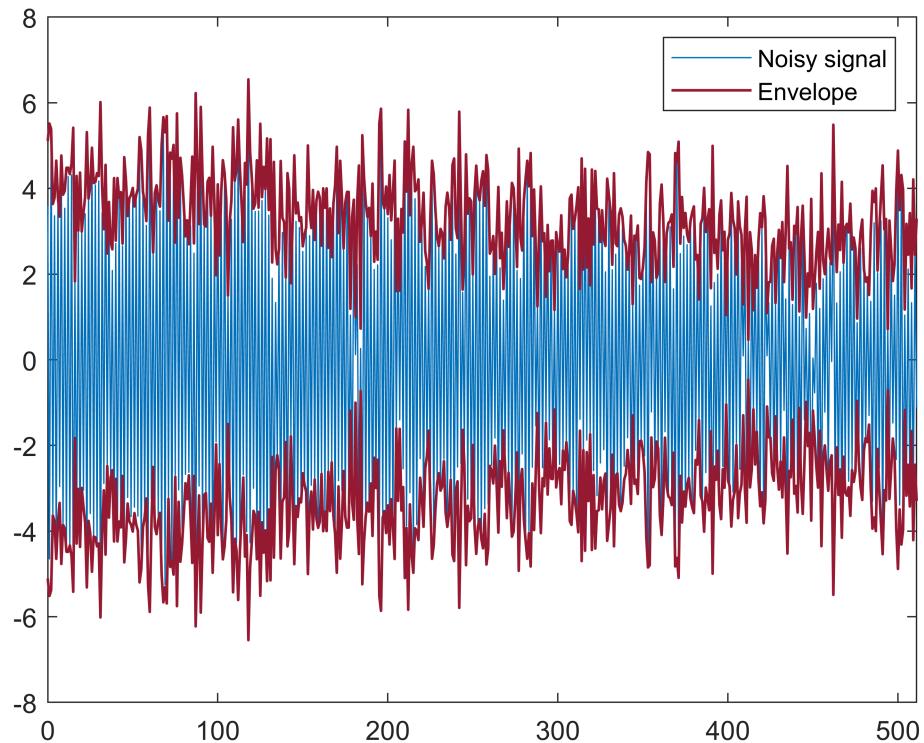
```

N = 512;
n = (0:N-1);
w = randn(N, 1)';
x_r = 4*exp(-0.001.*n).*cos(pi*n) + w;

x_c = hilbert(x_r);
x_envelope = abs(x_c);

plot(n, x_r)
hold on
plot(n, [-1;1]*x_envelope, 'Color', [0.6 0.1 0.2], 'Linewidth',1)
hold off
legend('Noisy signal', 'Envelope')
xlim([0, max(n)])

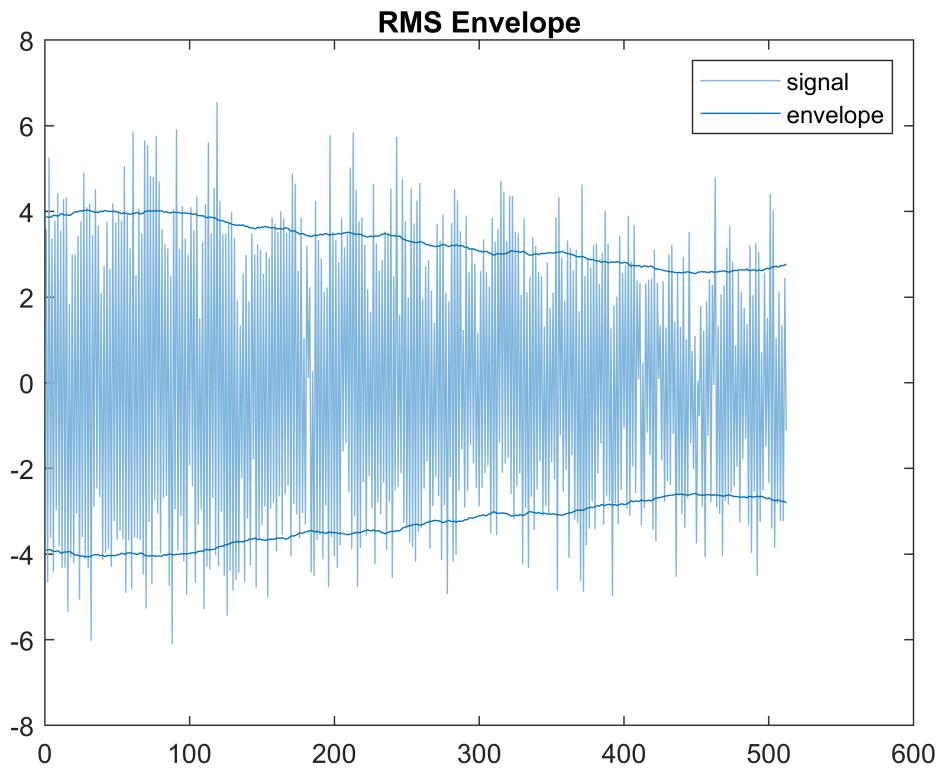
```



Hilbert transform is sensitive to noise. We observe spurious peaks in the envelope due to noise.

We can generate a moving Root-Mean-Square (RMS) average envelopes by using a sliding window of size W . Using a large window size smooths out the envelope.

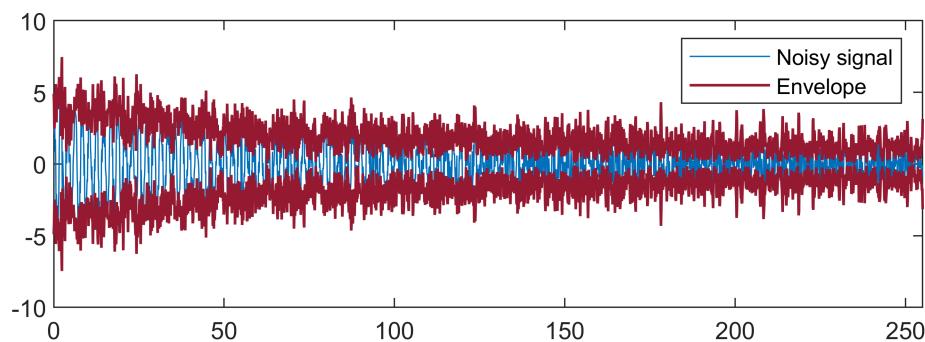
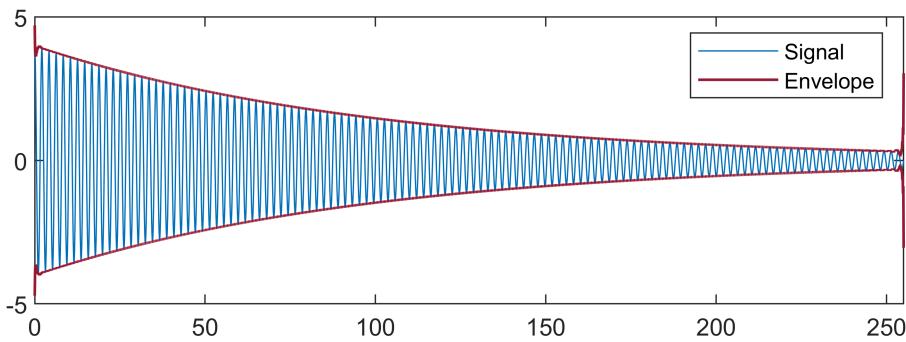
```
envelope(x_r, 100, 'rms');
```



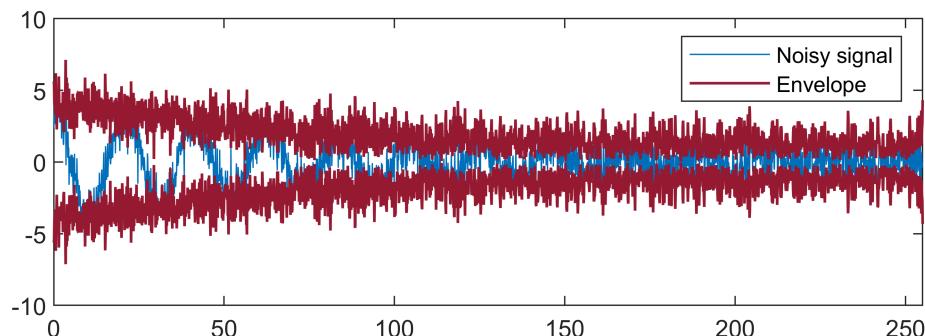
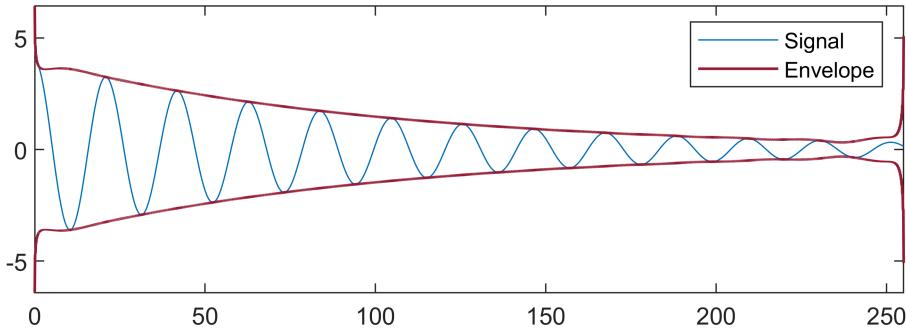
[✓] 3. Try with other two signals

3. Repeat the above two questions for the signals $x_1(n) = 4e^{-0.01n} \cos(3n)$ and $x_2(n) = 4e^{-0.01n} \cos(0.3n)$.

```
clear variables;
N = 256;
n = 0:0.1:N-1;
x = 4*exp(-0.01.*n).*cos(3*n);
plot_signal_and_envelope(n, x)
```



```
x = 4*exp(-0.01.*n).*cos(0.3*n);  
plot_signal_and_envelope(n, x)
```



[»] ADSI Problem 7.8: Amplitude modulated signal

Consider an amplitude modulated signal given by

$$x(t) = (1 + 0.2 \cos(2\pi 37t)) \cos(2\pi * 20000t)$$

```
clear variables;
```

[✓] 1. Plot 1 second of the signal and confirm that it is an amplitude modulated signal with a carrier frequency of 20 kHz.

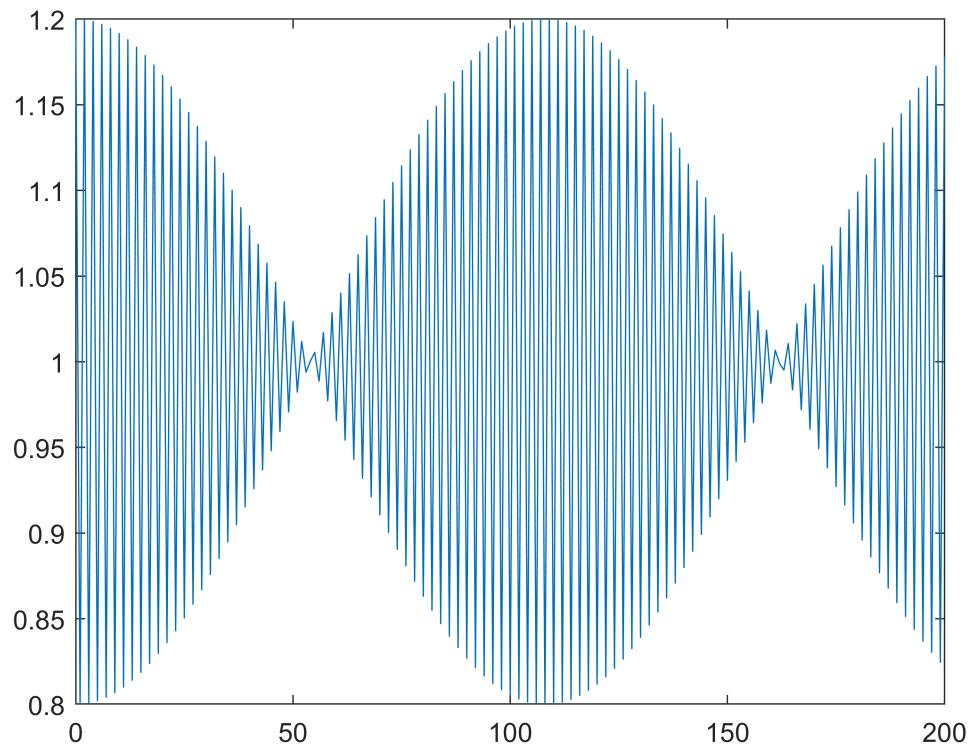
The discrete-time signal $x[n]$ of continuous-time signal $x(t)$ can be represented as:

$$x[n] = s(n \cdot T_s)$$

where

- n is an integer denoting the sample index
- T_s is the sampling period

```
Ts = 8000; % Sampling rate
n = 0:200;
x = 1 + 0.2.*cos(2*pi*37*n/Ts).*cos(2*pi*20000*n/Ts);
plot(n, x)
```



[»] 2. Use the procedure from Figure 9.8 in the note to frequency shift the signal to 15 kHz. Plot the spectrum of the signal at each step in the procedure.

Functions

```

function plot_signal_and_envelope(n, x)
w = randn(numel(n), 1)';
x_noisy = x + w;

x_c = hilbert(x);
x_envelope = abs(x_c);

x_noisy_c = hilbert(x_noisy);
x_noisy_envelope = abs(x_noisy_c);

clf
figure
ax1 = subplot(2,1,1);
plot(n, x)
hold on

```

```
plot(n, [-1;1]*x_envelope, 'Color', [0.6 0.1 0.2], 'Linewidth',1)
hold off
legend('Signal', 'Envelope')
xlim([0, max(n)])

ax2 = subplot(2,1,2);
plot(n, x_noisy)
hold on
plot(n, [-1;1]*x_noisy_envelope, 'Color', [0.6 0.1 0.2], 'Linewidth',1)
hold off
legend('Noisy signal', 'Envelope')
xlim([0, max(n)])
clear figure;
figure
end
```

Karhunen–Loève transform

Table of Contents

Optimum orthogonal transforms.....	1
Quiz: What happens if we perform a KLT on a white noise process?.....	2
Karhunen–Loève Transform Implementation.....	3
Work through the code and explain what it does.....	5
Play around with the noise parameters and comment on how the plots are altered by the noise parameters.....	6
How are the results affected by N, the number of experiments?.....	6
What happens if colored noise is used instead of white noise?	8

Optimum orthogonal transforms

KLT transform is a $p \times p$ orthogonal transform for random signals. It is also known as PCA.

The forward transform can be expressed as

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} = \begin{bmatrix} \leftarrow & \mathbf{a}_1^T & \rightarrow \\ \leftarrow & \mathbf{a}_2^T & \rightarrow \\ \vdots & \vdots & \vdots \\ \leftarrow & \mathbf{a}_p^T & \rightarrow \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \quad \text{or } \mathbf{y} = \mathbf{A}^T \mathbf{x}, \quad (14.179)$$

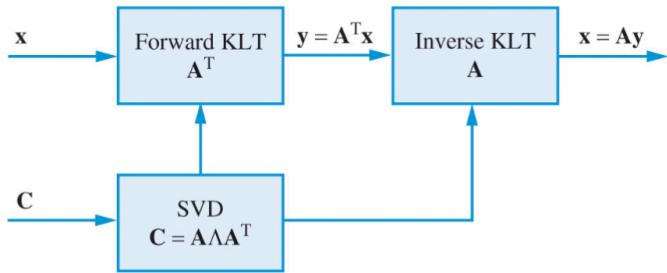
where \mathbf{A} is a $p \times p$ transform matrix.

To simplify the computation of the inverse transform, we require that the matrix \mathbf{A} is orthogonal so:

$$\mathbf{A}^T = \mathbf{A}^{-1}$$

The inverse transform provides an expansion of the input signal vector \mathbf{x} into orthogonal components using as a basis the columns of transform matrix \mathbf{A} .

The Karhunen–Loève transform We wish to find the orthogonal transform that provides a representation of the input signal vector which is optimum with respect to the mse criterion.



$$[U, \Lambda, A] = svd(C) \quad (14.205)$$

where \mathbf{C} is the covariance matrix of \mathbf{x} .

Fourier transform

$$x(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(nt) + b_n \sin(nt)), \quad -\pi \leq t \leq \pi$$

Karhunen-Loeve transform

$$x(t) = \sum_{n=1}^{\infty} z_n \phi_n(t), \quad 0 \leq t \leq T$$

Quiz: What happens if we perform a KLT on a white noise process?

```
clear variables;
```

```
C = cov(randn(100,10));  
[U, lambda, A] = svd(C);  
diag(lambda)'
```

```
ans = 1x10
    1.4367    1.3516    1.2134    0.9697    0.9281    0.7613    0.7375    0.6649 ...
```

```
X = randn(100,4);
```

```
C = cov(X - mean(X));  
[U, lambda, A] = svd(C);  
diag(lambda)'
```

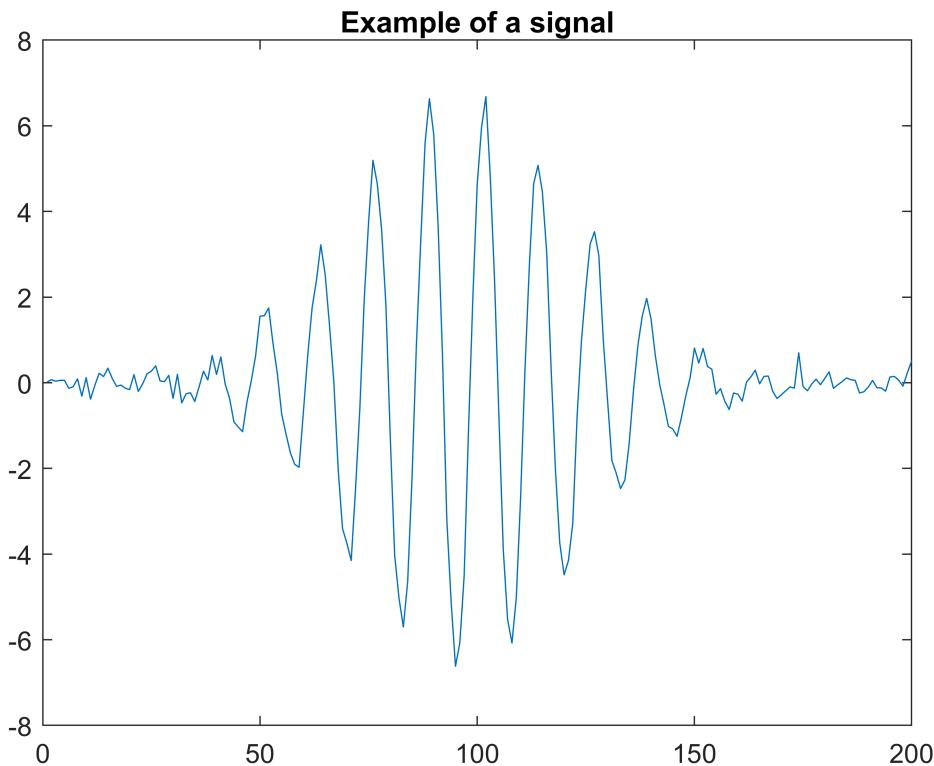
```
ans = 1x4  
1.2396    0.9595    0.8549    0.7867
```

Karhunen–Loève Transform Implementation

```
clear variables;
```

Consider the following piece of Matlab code:

```
% Settings  
n=200; % segment length  
  
t=0:n-1; % time axis (a column vector 1xn)  
t=t(:); % time axis (as row vector nx1)  
  
amplitudeNoise=10; % noise parameters  
sigmaNoise=20;  
t0Noise=20;  
whiteNoise=0.2;  
  
A=5+amplitudeNoise*rand(1);  
sigma=10+sigmaNoise*rand(1);  
t0=100+t0Noise*randn(1);  
s=A*exp( -(t-t0).^2 / (2*sigma^2)).*cos(0.5*t);  
s=s+whiteNoise*randn(n,1);  
plot(s);  
title('Example of a signal')
```



```

N=10; % Number of signals to generate
X=zeros(n,N);

%% build up an 'observed' data set
for q=1:N
    A=5+amplitudeNoise*rand(1); % The highest peak
    sigma=10+sigmaNoise*rand(1); % The "width" of pulse
    t0=100+t0Noise*randn(1); % The center of the highest peak
    s=A*exp( -(t-t0).^2 / (2*sigma^2)).*cos(0.5*t);
    X(:,q)=s+whiteNoise*randn(n,1);
end

rankX = rank(X)

```

rankX = 10

```

%% Calculate C using 14.206 and perform a KLT
C=zeros(n,n);
mean_vector = mean(X, 2); % Compute mean vector across all samples.
for q=1:N
    % x=X(:,q)-mean(X(:,q)); % Original code, seems wrong?
    x = X(:,q) - mean_vector; % My code
    C = C + x*x';
end
C=C/N;
[U, lambda, A]=svd(C);

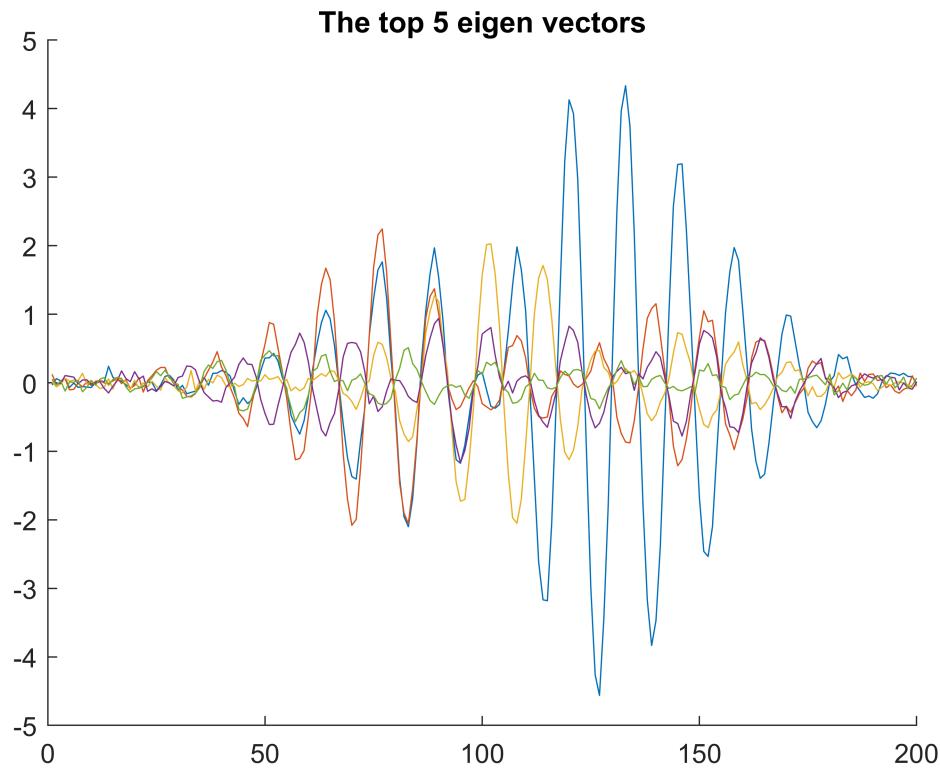
%% Plot results

```

```

figure(1)
clf(figure(1))
hold on
for q=1:5
    plot(A(:,q)*sqrt(lambda(q,q)))
end
title('The top 5 eigen vectors')
hold off

```



```

figure(2)
stem(diag(lambda))
xlabel('Component')
ylabel('Eigenvalue')

```

Work through the code and explain what it does.

Play around with the noise parameters and comment on how the plots are altered by the noise parameters.

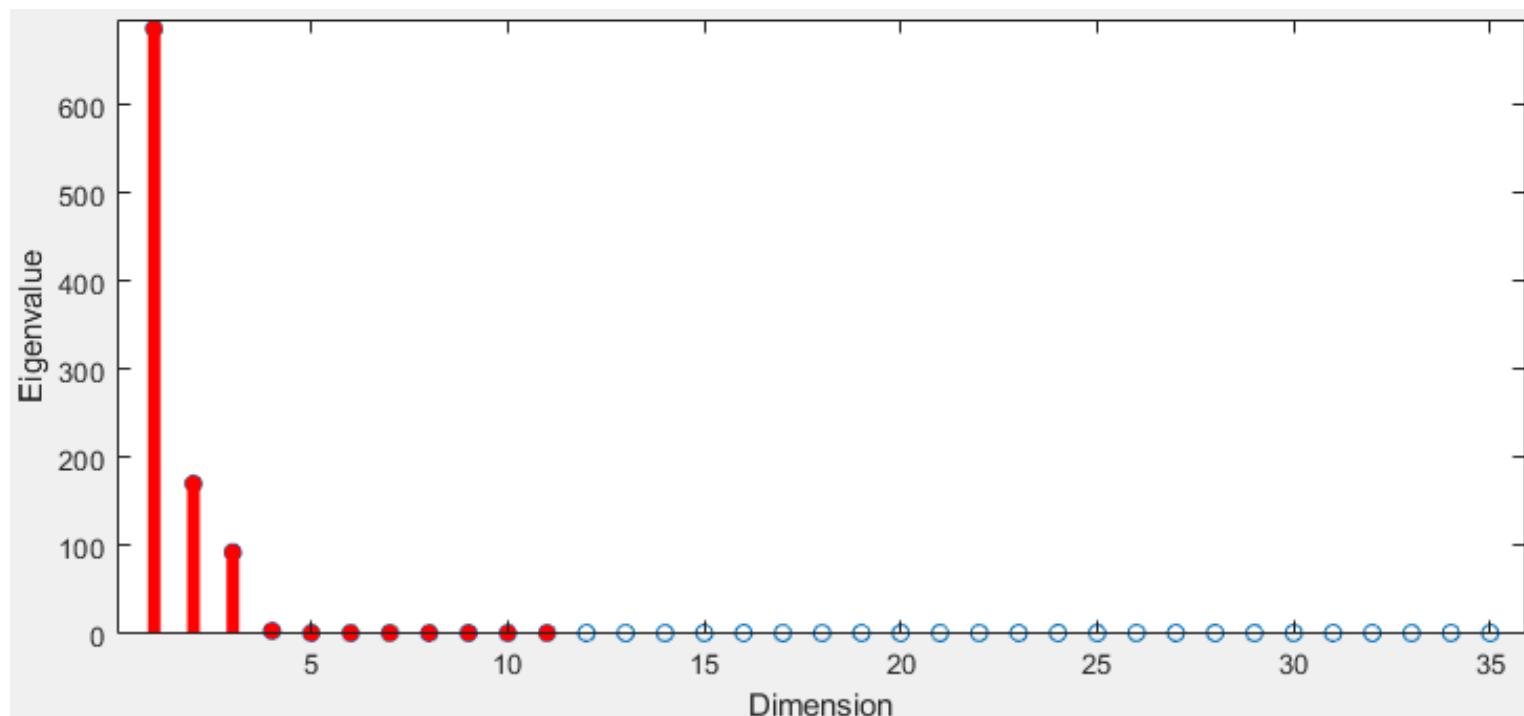
When the noise parameters are all set to zero, we are generating the same signal duplicated N times. So our X has N columns of the same $n \times 1$ vector. This means the rank of the matrix is 1.

Changing the amplitude noise parameter does not effect the rank of the matrix since changes in amplitude cause vectors to be scaled.

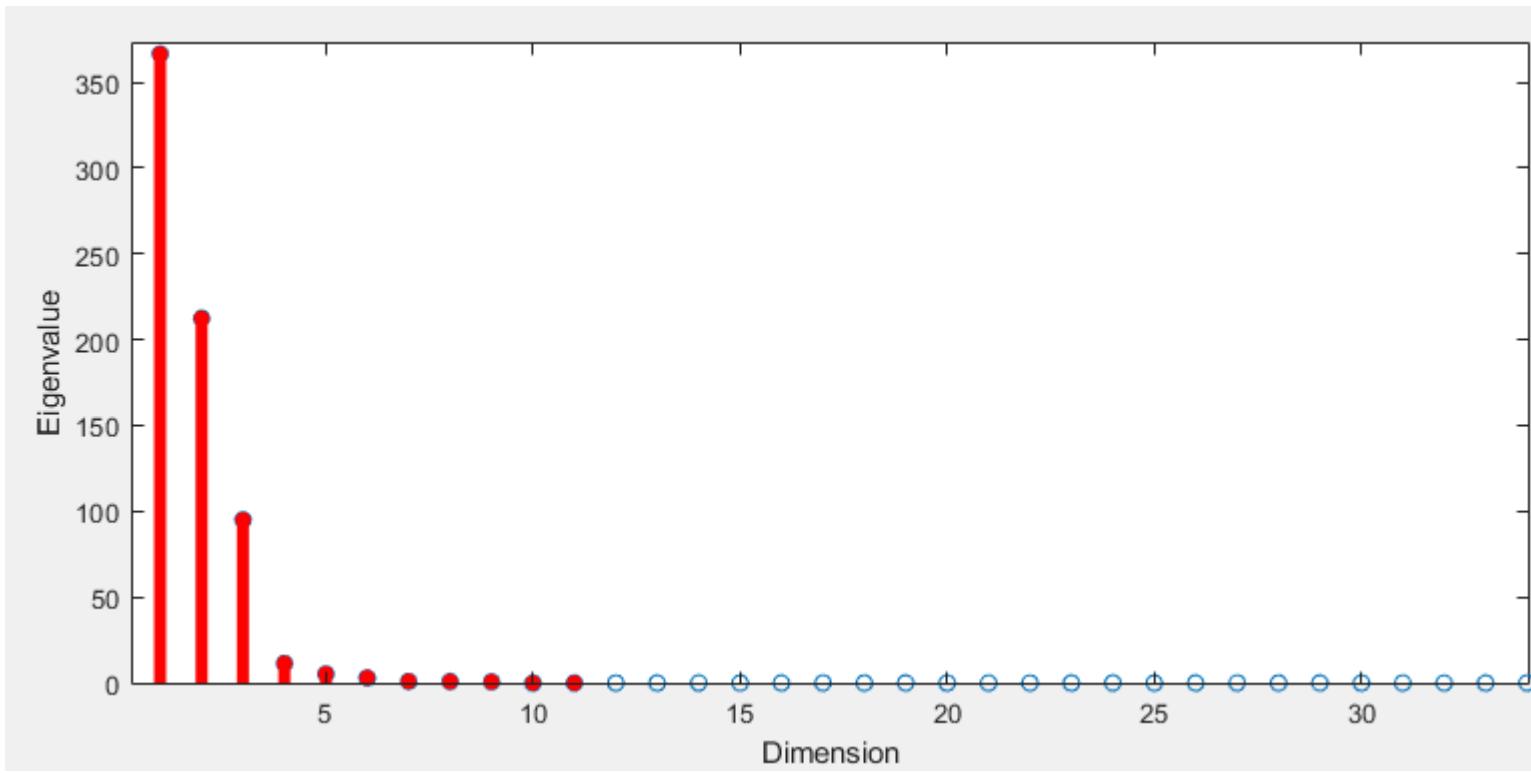
Changes in the other noise parameters causes the X matrix to be full rank (almost at every generation) since the signals are more likely to be shifted.

How are the results affected by N, the number of experiments?

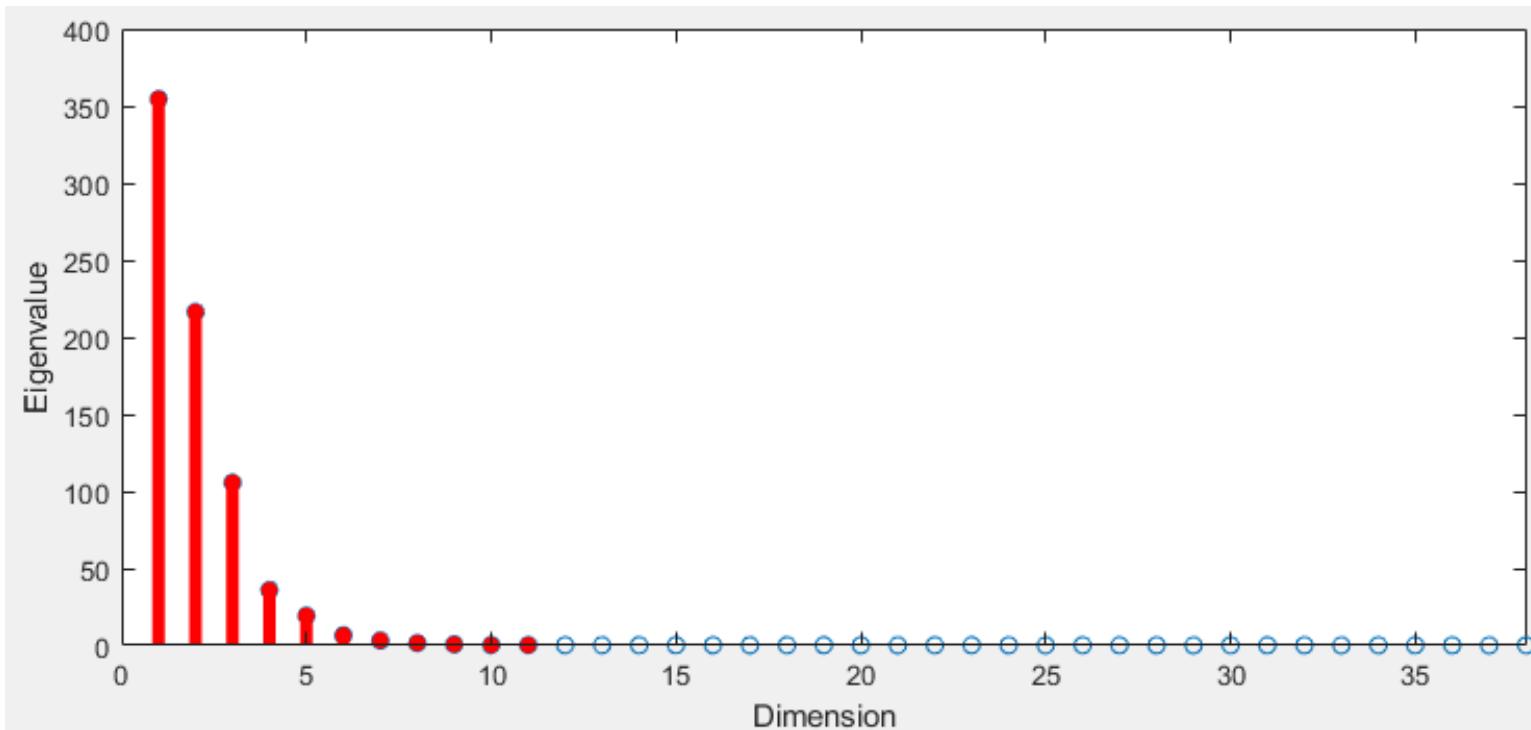
When N=5



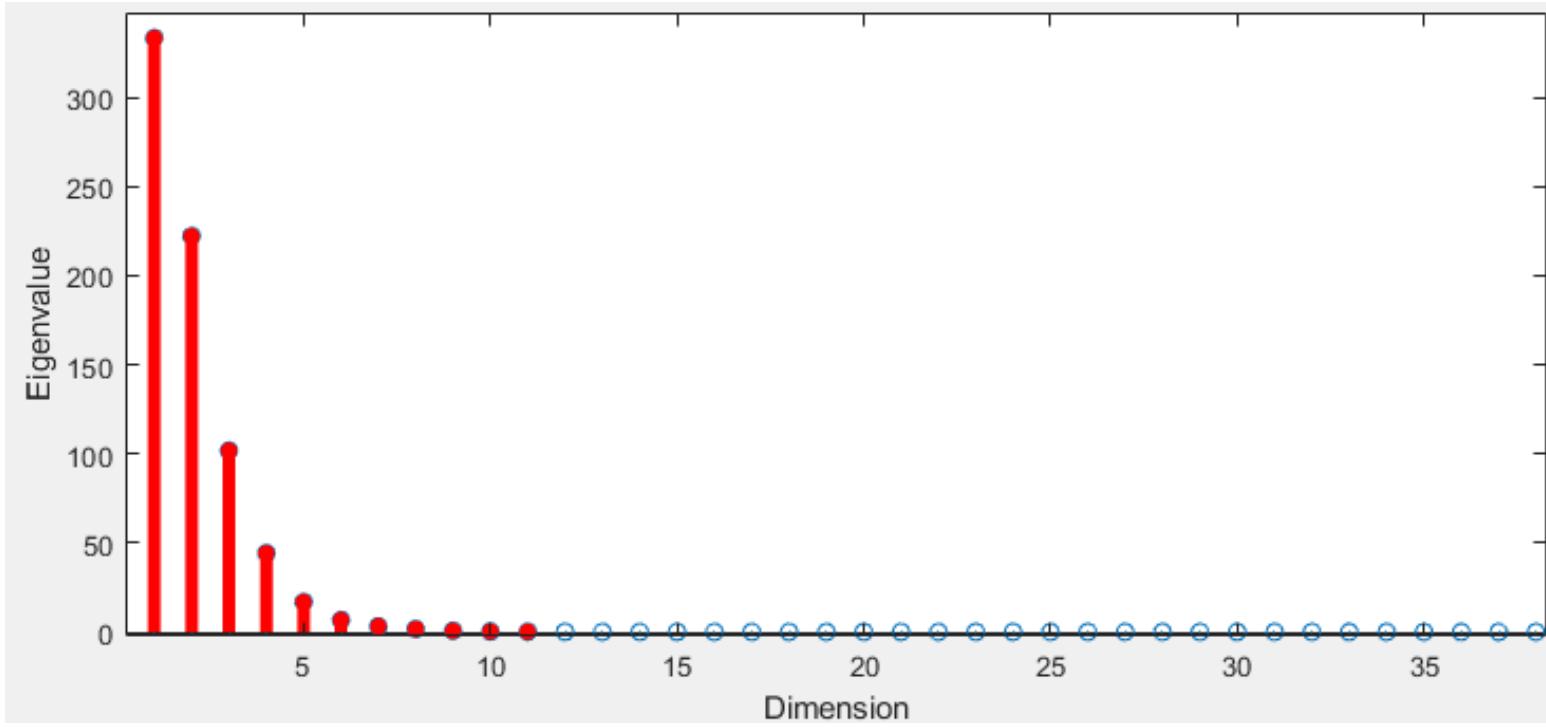
When N=10



When $N=100$

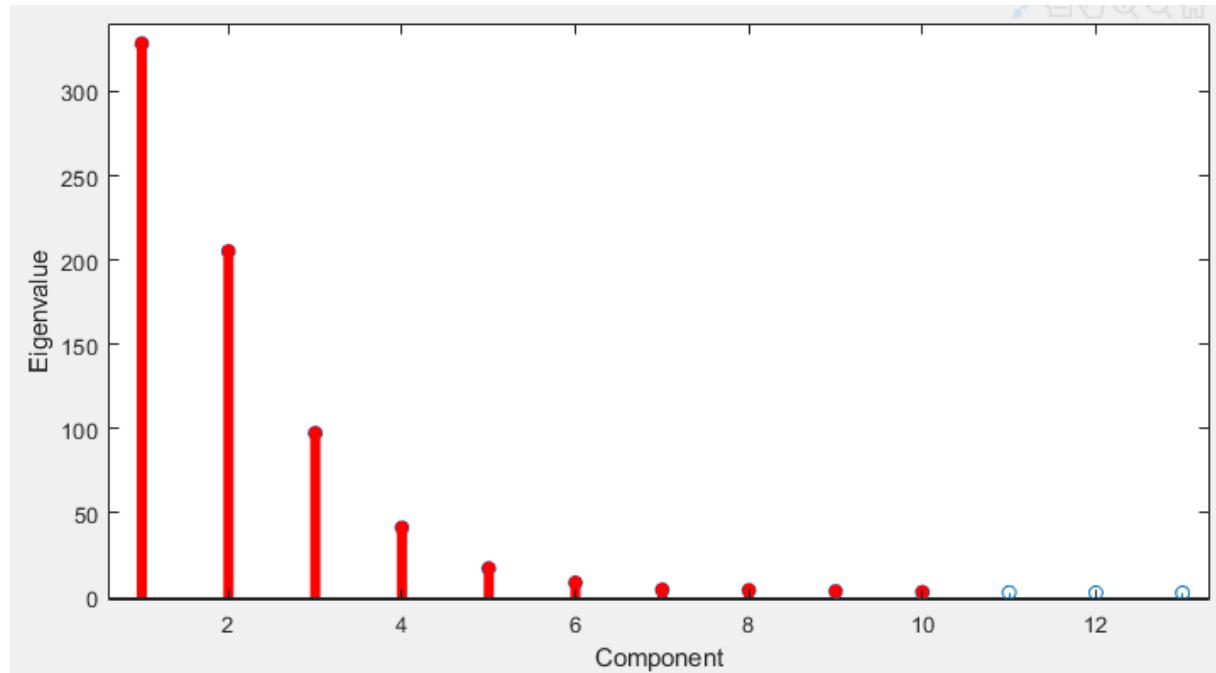


$N=1000$



What happens if colored noise is used instead of white noise?

Even with coloured noise, it seems that the largest variance within the different signals can be explained by the top 5-10 components.



```
amplitudeNoise=10; % noise parameters
```

```

sigmaNoise=20;
t0Noise=20;
colouredNoise=0.3;

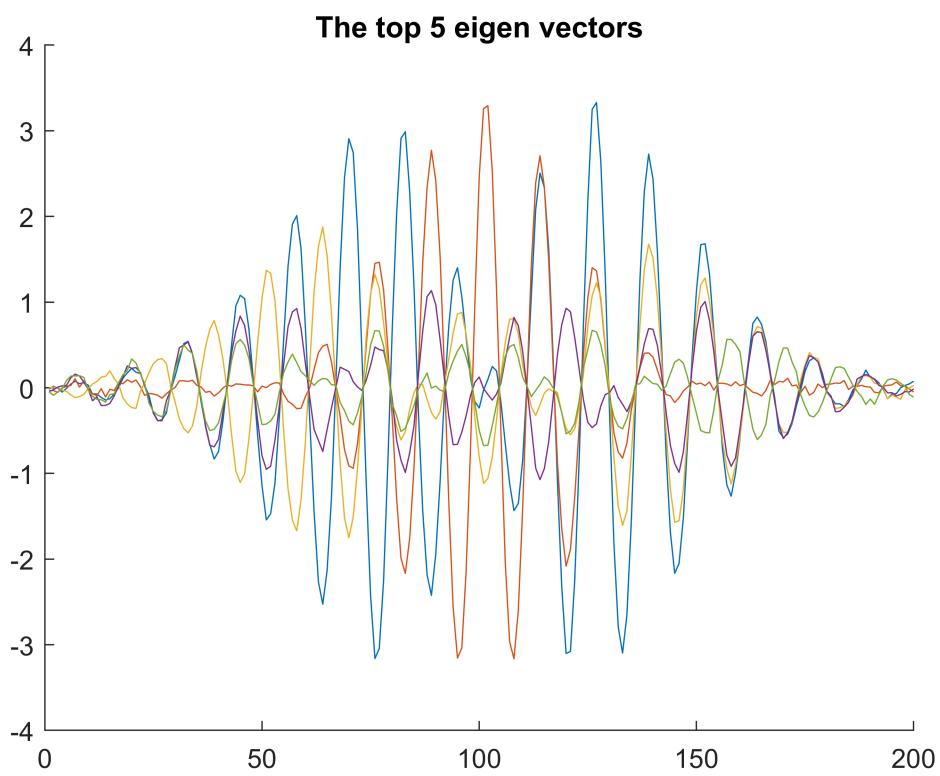
N=1000; % Number of signals to generate
X=zeros(n,N);

%% build up an 'observed' data set
for q=1:N
    A=5+amplitudeNoise*rand(1); % The highest peak
    sigma=10+sigmaNoise*rand(1); % The "width" of pulse
    t0=100+t0Noise*randn(1); % The center of the highest peak
    s=A*exp( -(t-t0).^2 / (2*sigma^2).*cos(0.5*t);
    X(:,q)=s+step(dsp.ColoredNoise('InverseFrequencyPower', colouredNoise, 'SamplesPerFrame', n));
end

%% Calculate C using 14.206 and perform a KLT
C=zeros(n,n);
mean_vector = mean(X, 2); % Compute mean vector across all samples.
for q=1:N
    % x=X(:,q)-mean(X(:,q)); % Original code, seems wrong?
    x = X(:,q) - mean_vector; % My code
    C = C + x*x';
end
C=C/N;
[U, lambda, A]=svd(C);

%% Plot results
figure(1)
clf(figure(1))
hold on
for q=1:5
    plot(A(:,q)*sqrt(lambda(q,q)))
end
title('The top 5 eigen vectors')
hold off

```



```
figure(2)
stem(diag(lambda))
xlabel('Component')
ylabel('Eigenvalue')
```

Lattice Filters

Table of Contents

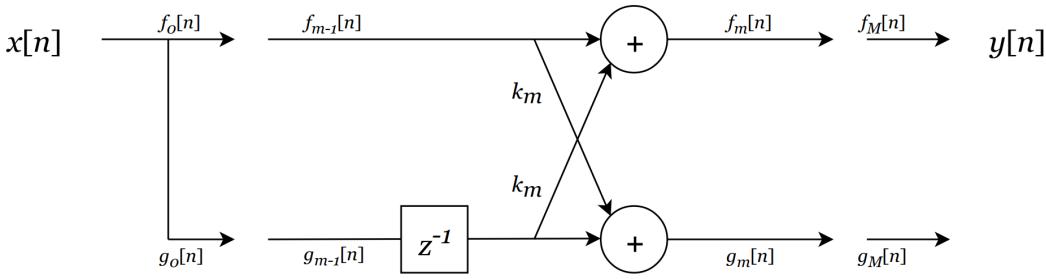
Lattice Structures.....	1
All-zero lattice structure.....	1
Single-stage filter.....	3
Two stage filter.....	3
Three stage filter.....	4
How to find reflection coefficients from the impulse response?.....	4
All-pole lattice structure.....	5
What are the advantages of lattice structures?.....	7
[✓] ADSI Problem 2.1: Implementation of All-zero lattice filters.....	7
[✓] ADSI Problem 2.2: Find impulse response of an all-zero lattice filter.....	8
[✓] ADSI Problem 2.3: Find the reflection coefficients for an all-zero lattice filter.....	10
[✓] ADSI Problem 2.5: All-zero lattice filter, find coefficients from system function (division by zero problem).....	13
[✓] ADSI Problem 2.7: Find system function an all-pole filter from reflection coefficients.....	15
[✓] ADSI Problem 2.10: Linear prediction and lattice filters.....	16
1) Compute autocorrelation from a signal.....	16
2) Compute the coefficients for the optimum 2nd-order linear predictor.....	19
3) Compare the linear predictor with a two-stage all-zero lattice filter.....	20
4) Compute reflection coefficients from the linear predictor.....	21
5) Compute prediction error.....	22
Exam 2016 Problem 3: All-pole Lattice Filter.....	23
1) Explain why a system is a digital notch filter.....	23
2) Find reflection coefficients for the all-pole lattice filter.....	24
3) Determine the group delay in the digital notch filter.....	25
Exam 2017 Problem 2: Lattice Structures.....	27
1) Determine the two equations relating outputs to inputs.....	27
2) When is the first non-zero sample observed at the output?.....	28
3) Calculate the transfer function for a one-stage lattice filter.....	29
Exam 2014 Problem 1.....	30
Functions.....	30

Lattice Structures

A lattice filter is an example of an all-pass filter typically used the analysis and synthesis of speech signals.

All-zero lattice structure

An **all-zero** lattice models an FIR system.



Each section has two inputs ($f_{m-1}[n]$ and $g_{m-1}[n-1]$) and two outputs ($f_m[n], g_m[n]$).

The m 'th section/stage can be computed as follows:

$$f_m[n] = f_{m-1}[n] + k_m g_{m-1}[n-1], \quad m = 1, 2, \dots, M \quad (9.55a)$$

$$g_m[n] = k_m f_{m-1}[n] + g_{m-1}[n-1]. \quad m = 1, 2, \dots, M \quad (9.55b)$$

The overall system input and output are given by:

$$x[n] = f_0[n] = g_0[n], \quad (9.56a)$$

$$y[n] = f_M[n]. \quad (9.56b)$$

In general, the outputs of the m th section correspond to two FIR filters with the same coefficients but in reverse order:

$$f_m[n] = \sum_{i=0}^m a_i^{(m)} x[n-i], \quad m = 1, 2, \dots, M \quad (9.64a)$$

$$g_m[n] = \sum_{i=0}^m a_{m-i}^{(m)} x[n-i]. \quad m = 1, 2, \dots, M \quad (9.64b)$$

The system functions of these all-zero FIR filters are given by:

$$A_m(z) \triangleq \frac{F_m(z)}{F_0(z)} = \sum_{i=0}^m a_i^{(m)} z^{-i}, \quad a_0^{(0)} = 1 \quad (9.65a)$$

$$B_m(z) \triangleq \frac{G_m(z)}{G_0(z)} = \sum_{i=0}^m a_{m-i}^{(m)} z^{-i} \triangleq \sum_{i=0}^m b_i^{(m)} z^{-i}. \quad (9.65b)$$

Suppose we have a FIR system of the form:

$$H(z) = \sum_{k=0}^M h[k] z^{-k}$$

We can start taking the z-transforms.

Since $z[n] = f_0[n] = g_0[n]$, taking the z-transform is straightforward:

$$X(z) = F_0(z) = G_0(z)$$

The difference equations of the m th section is:

$$\begin{aligned}f_m[n] &= f_{m-1}[n] + k_m g_{m-1}[n-1] \\g_m[n] &= k_m f_{m-1}[n] + g_{m-1}[n-1]\end{aligned}$$

Taking the z-transform of the m th section gives us:

$$\begin{aligned}F_m(z) &= F_{m-1}(z) + k_m G_{m-1}(z)z^{-1} \\G_m(z) &= k_m F_{m-1}(z) + G_{m-1}(z)z^{-1}\end{aligned}$$

We can normalise these two z-transforms as follows:

$$A_m(z) = \frac{F_m(z)}{F_0(z)} = A_{m-1}(z) + k_m B_{m-1}(z)z^{-1}$$

$$B_m(z) = \frac{G_m(z)}{G_0(z)} = k_m A_{m-1}(z) + B_{m-1}(z)z^{-1}$$

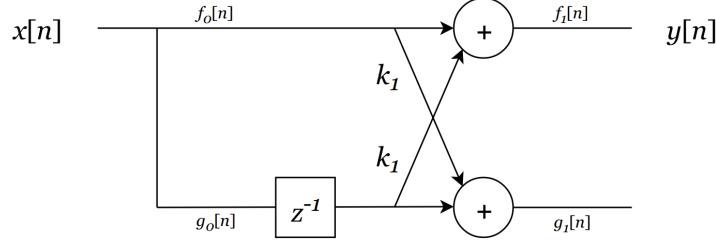
By the way we normalise, we notice that $A_0(z) = B_0(z) = 1$ because

$$A_0 = \frac{F_0(z)}{F_0(z)} = 1 \text{ and } B_0 = \frac{G_0(z)}{G_0(z)} = 1$$

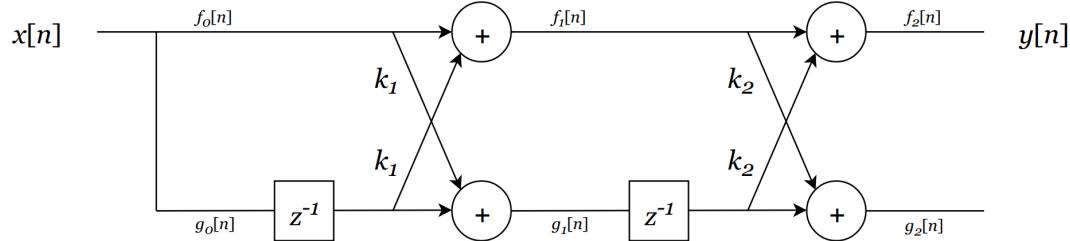
Because of the normalisation, we can write:

$$H(z) = h[0]A_M(z)$$

Single-stage filter



Two stage filter



Compute the difference equations for the two FIR systems:

$$f_2[n] = f_1[n] + k_2 g_1[n-1]$$

$$g_2[n] = k_2 f_1[n] + g_1[n-1]$$

$$f_1[n] = f_0[n] + k_1 g_0[n - 1]$$

$$g_1[n] = k_1 f_0[n] + g_0[n - 1]$$

If we substitute we get:

$$f_2[n] = f_1[n] + k_2 g_1[n - 1]$$

$$f_2[n] = f_0[n] + k_1 g_0[n - 1] + k_2(k_1 f_0[n] + g_0[n - 1])$$

Since $x[n] = f_0[n] = g_0[n]$ and $y[n] = f_2[n]$ then we can rewrite the difference equation for two stage all-zero lattice filter as:

$$y[n] = x[n] + k_1(1 + k_2)x[n - 1] + k_2x[n - 2]$$

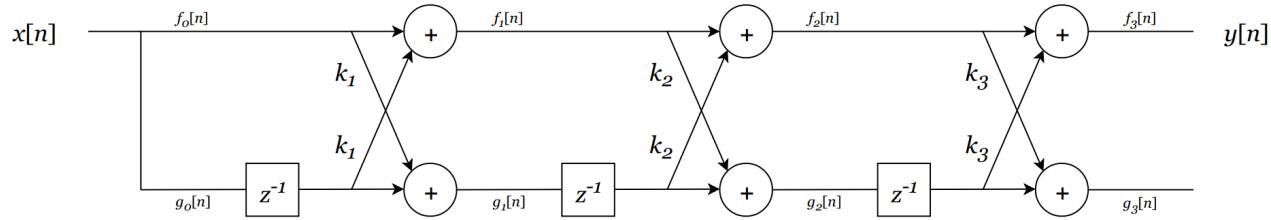
Taking the z-transform, we have:

$$Y(z) = X(z) + k_1(1 + k_2)z^{-1}X(z) + k_2z^{-2}X(z)$$

We can compute the transfer function:

$$H(z) = \frac{Y(z)}{X(z)} = 1 + k_1(1 + k_2)z^{-1} + k_2z^{-2}$$

Three stage filter



How to find reflection coefficients from the impulse response?

Suppose we want to determine the reflection coefficients $k_m, m = 1, 2, \dots, M$ from the following an M th-order normalised FIR filter:

$$H(z) = \sum_{k=0}^M h[k]z^{-k}$$

First, we can define the filter coefficients as follows:

$$a_k = \frac{h[k]}{h[0]} \text{ where } k = 0, 1, \dots, M$$

The recursive algorithm works as follows:

1. Compute $A_M(z) = \frac{H(z)}{h[0]}$
2. Compute $k_M = a_M$ where a_M is the last coefficients of $A_M(z)$. For example, if $A_M(z) = 1 + 0.06z^{-1} - 0.42z^{-2} + 0.5z^{-3}$ then $k_M = 0.5$
3. Compute $B_M(z)$ by flipping the coefficients of $A_M(z)$

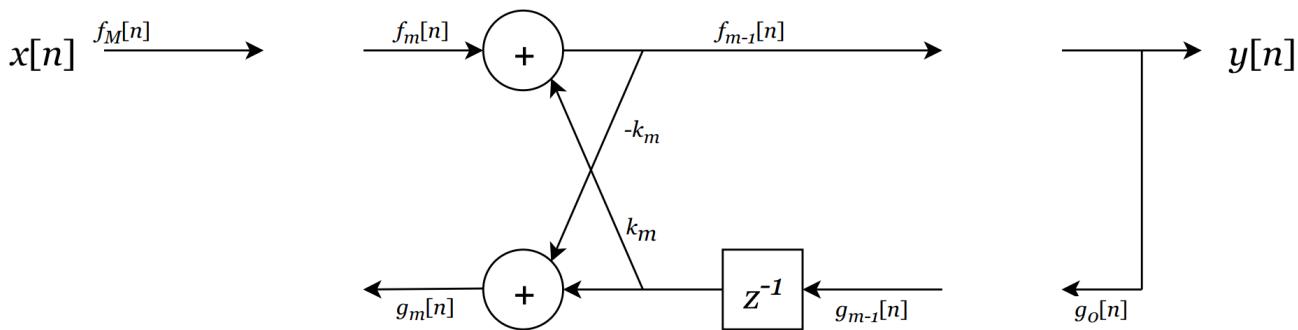
4. Set $m = M$
5. Compute $A_{m-1}(z) = \frac{1}{1 - k_m^2} [A_m(z) - k_m B_m(z)]$. Notice that this is where the algorithm fails because if $k_m = 1 \rightarrow k_m^2 = 1$ then we will have division by zero.
6. Compute $k_{m-1} = a_{m-1}$ where a_{m-1} is the last coefficients of $A_{m-1}(z)$
7. Compute $B_{m-1}(z)$ by flipping the coefficients of $A_{m-1}(z)$. Alternatively, compute $B_{m-1}(z) = z^{-m+1} A_{m-1}\left(\frac{1}{z}\right)$
8. Set $m = m - 1$
9. Go to step 5 if $m \neq 0$
10. We know that $A_0(z) = B_0(z) = 1$

See page 513

All-pole lattice structure

An **all-pole** lattice models an IIR system.

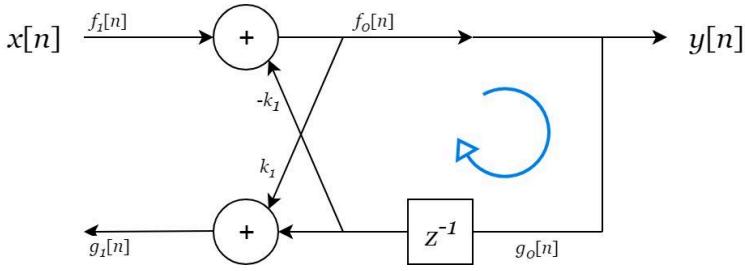
The figure below shows how an all-pole lattice structure looks like



$$f_{m-1}[n] = f_m[n] - k_m g_{m-1}[n - 1]$$

$$g_m[n] = k_m f_{m-1}[n] + g_{m-1}[n - 1]$$

When $M = 1$



The difference equation for when $M = 1$ is given as:

$$f_0[n] = f_1[n] - k_1 g_0[n - 1]$$

Since $f_0[n] = g_0[n] = y[n]$ and $f_1[n] = x[n]$ then we have the following difference equation

$$y[n] = x[n] - k_1 y[n - 1]$$

This is the difference equation for a IIR filter!

If we take the z-transform, we get:

$$Y(z) = X(z) - k_1 Y(z)z^{-1}$$

We can compute the transfer function:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 + k_1 z^{-1}} = \frac{1}{A(z)}$$

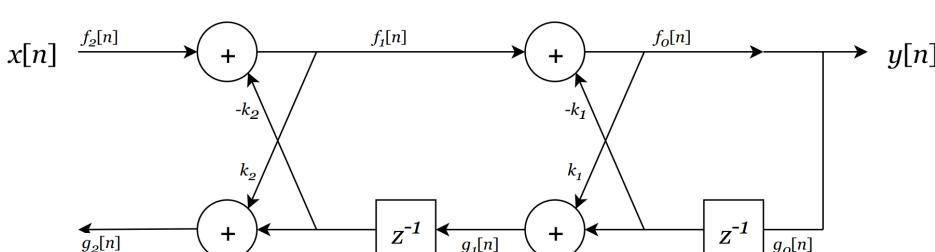
Let us look at the other system $g_1[n]$:

$$g_1[n] = k_1 f_0[n] + g_0[n - 1]$$

$$g_1[n] = k_1 y[n] + y[n - 1]$$

It is clear that $g_1[n]$ is a FIR filter of the output.

When $M = 2$



We compute the difference equations from the output:

$$f_0[n] = f_1[n] - k_1 g_0[n - 1]$$

$$g_1[n] = k_1 f_0[n] + g_0[n - 1]$$

The next stage:

$$f_1[n] = f_2[n] - k_2 g_1[n-1]$$

$$g_2[n] = k_2 f_1[n] + g_1[n-1]$$

We can substitute:

$$f_0[n] = f_2[n] - k_2 g_1[n-1] - k_1 g_0[n-1]$$

$$f_0[n] = f_2[n] - k_2(k_1 f_0[n-1] + g_0[n-2]) - k_1 g_0[n-1]$$

Since $f_0[n] = g_0[n] = y[n]$ and $f_2[n] = x[n]$ then we have:

$$y[n] = x[n] - k_2(k_1 y[n-1] + y[n-2]) - k_1 y[n-1]$$

$$y[n] = x[n] - k_1(1 + k_2)y[n-1] - k_2 y[n-2]$$

If we take the z-transform, we get:

$$Y(z) = X(z) - k_1(1 + k_2)Y(z)z^{-1} - k_2 Y(z)z^{-2}$$

The transfer function becomes:

$$\frac{Y(z)}{X(z)} = \frac{1}{1 + k_1(1 + k_2)z^{-1} + k_2 z^{-2}}$$

What are the advantages of lattice structures?

When is an all-pole filter stable?

The all-zero and all-pole lattice structures require twice the number of multiplications per output sample as the direct forms. However, they have two unique advantages compared to direct form structures.

The first advantage follows from the following theorem:

Theorem 9.4.1 The roots of the polynomial $A_M(z)$ are inside the unit circle if and only if

$$|k_m| < 1. \quad m = 1, 2, \dots, M \tag{9.87}$$

Advantage 1: If the lattice parameters satisfy Eq. (9.87) then the all-zero lattice filter is **minimum-phase** and the all-pole filter is **stable**.

Advantage 2: Lattice structures are insensitive to quantization of the k -parameters.

[✓] ADSI Problem 2.1: Implementation of All-zero lattice filters

Work your way through the MATLAB code in Figure 9.26 and make sure you understand what is going on.

```

function [y] = azlatfilt(k, x, G)
% AZLATFILT      Function for implementation of an all-zero FIR lattice filter
% From Figure 9.26 (p 515)

M = length(k);

% Create an array for the sequence f_m[n]
f = zeros(1,M);

% Create an array for the sequence g_m[n]
g = f;

% Create an array for the sequence g_m[n-1]
oldg = zeros(1,M);

% Keeps track of x[n-1]
oldx = 0;
x = G*x;
y = zeros(size(x));

for n=1:length(x)
    % Compute f1[n] = x[n] + k1x[n-1] -> Equation (9.57a)
    f(1) = x(n)+k(1)*oldx;

    % Compute g1[n] = k1x[n] + x[n-1] -> Equation (9.57b)
    g(1) = k(1)*x(n)+oldx;
    oldx = x(n);
    for m = 2:M
        % Compute: fm[n] = fm?1[n] + km gm?1[n ? 1] -> Equation (9.55a)
        f(m) = f(m-1)+k(m)*oldg(m-1);

        % Compute: gm[n] = km fm?1[n] + gm?1[n ? 1] -> Equation (9.55b)
        g(m) = k(m)*f(m-1)+oldg(m-1);

        % Delay
        oldg(m-1) = g(m-1);
    end

    % y[n] = fM[n] -> Equation (9.56b)
    y(n) = f(M);
end
end

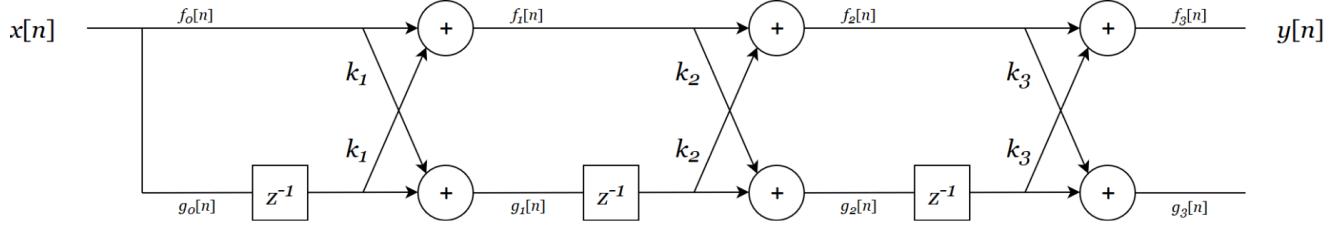
```

[✓] ADSI Problem 2.2: Find impulse response of an all-zero lattice filter

Consider an all-zero (FIR) lattice filter with $k_1 = 0.65$, $k_2 = -0.34$ and $k_3=0.8$.

1. Find the impulse response of the filter by tracing an impulse $x(n) = \delta(n)$ through the filter.

Since the lattice filter has three reflection coefficients, it is a 3-stage all-zero lattice filter.



Here are the formulas needed to trace through the filter.

$$x[n] = f_0[n] = g_0[n],$$

$$y[n] = f_M[n].$$

$$f_m[n] = f_{m-1}[n] + k_m g_{m-1}[n-1], \quad m = 1, 2, \dots, M$$

$$g_m[n] = k_m f_{m-1}[n] + g_{m-1}[n-1]. \quad m = 1, 2, \dots, M$$

The input is $x[n] = \delta[n] = [1, 0, 0, 0]$

$n = 1$

$$f_0(n) = g_0(n) = x(1) = 1$$

$$f_1(n) = f_0(n) + k_1 g_0(n-1) = 1 + 0 = 1$$

$$g_1(n) = k_1 f_0(n) + g_0(n-1) = 0.65 + 0 = 0.65$$

$$f_2(n) = f_1(n) + k_2 g_1(n-1) = 1 - 0.34 \cdot 0 = 1$$

$$g_2(n) = k_2 f_1(n) + g_1(n-1) = -0.34 \cdot 1 + 0 = -0.34$$

$$f_3(n) = f_2(n) + k_3 g_2(n-1) = 1 + 0.8 \cdot 0.0 = \underline{1}$$

$$g_3(n) = k_3 f_2(n) + g_2(n-1) = 0.8 \cdot 1 + 0 = 0.8$$

$n = 2$

$$f_0(n) = g_0(n) = x(2) = 0$$

$$f_1(n) = 0 + 0.65 \cdot 1 = 0.65$$

$$g_1(n) = 0 + 1 = 1$$

$$f_2(n) = 0.65 - 0.34 \cdot 0.65 = 0.429$$

$$g_2(n) = -0.34 \cdot 0.65 + 0.65 = 0.429$$

$$f_3(n) = 0.429 + 0.8 \cdot -0.34 = \mathbf{0.157}$$

$$g_3(n) = 0.8 \cdot 0.429 - 0.34 = 0.0032$$

$n = 4$

$$f_0(n) = g_0(n) = x(2) = 0$$

$$f_1(n) = 0$$

$$g_1(n) = 0$$

$$f_2(n) = 0$$

$$g_2(n) = 0$$

$$f_3(n) = \mathbf{0.8}$$

$$g_3(n) = 1$$

We can also use the MATLAB function **azlatfilt** given in Figure 9.26 in the book.

```

k = [0.65, -0.34, 0.8];
x = [1, 0, 0, 0];
G = 1;
y = azlatfilt(k, x, G)

```

```

y = 1x4
1.0000    0.1570    0.0032    0.8000

```

The impulse response of the given filter is:

$$h[n] = [1, 0.157, 0.0032, 0.8]$$

[✓] ADSI Problem 2.3: Find the reflection coefficients for an all-zero lattice filter

An all-zero lattice filter has the following system function

$$H(z) = 1 + \frac{13}{24}z^{-1} + \frac{5}{8}z^{-2} + \frac{1}{3}z^{-3}$$

- Find the reflection coefficients k_1 , k_2 and k_3 for the corresponding lattice filter.

Can be computed using MATLAB function ***tf2latc*** to convert transfer function filter parameters to lattice filter for:

```

b = [1, 13/24, 5/8, 1/3];
a = 1;
k = tf2latc(b, a)

```

```

k = 3x1
0.2500
0.5000
0.3333

```

We can also try the algorithm (Figure 9.24) from the book

```

k = fir2lat(b)'

```

```

k = 3x1
0.2500
0.5000
0.3333

```

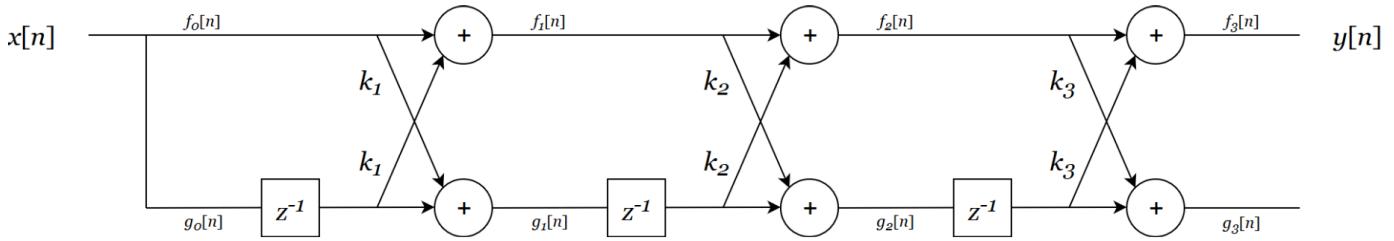
We can also do it by hand using the recursive algorithm:

The recursive algorithm works as follows:

- Compute $A_M(z) = \frac{H(z)}{h[0]}$

2. Compute $k_M = a_M$ where a_M is the last coefficients of $A_M(z)$. For example,
if $A_M(z) = 1 + 0.06z^{-1} - 0.42z^{-2} + 0.5z^{-3}$ then $k_M = 0.5$
3. Compute $B_M(z)$ by flipping the coefficients of $A_M(z)$
4. Set $m = M$
5. Compute $A_{m-1}(z) = \frac{1}{1 - k_m^2} [A_m(z) - k_m B_m(z)]$. Notice that this is where the algorithm fails because
if $k_m = 1 \rightarrow k_m^2 = 1$ then we will have division by zero.
6. Compute $k_{m-1} = a_{m-1}$ where a_{m-1} is the last coefficients of $A_{m-1}(z)$
7. Compute $B_{m-1}(z)$ by flipping the coefficients of $A_{m-1}(z)$. Alternatively, compute $B_{m-1}(z) = z^{-m+1} A_{m-1}\left(\frac{1}{z}\right)$
8. Set $m = m - 1$
9. Go to step 5 if $m \neq 0$
10. We know that $A_0(z) = B_0(z) = 1$

In this problem, we have a 3-stage all-zero lattice filter:



We are given its transfer function:

$$H(z) = 1 + \frac{13}{24}z^{-1} + \frac{5}{8}z^{-2} + \frac{1}{3}z^{-3}$$

Since this is a FIR system of the form:

$$H(z) = \sum_{k=0}^M h[k]z^{-k}$$

We know that $M = 3$, $h[k] = \left[1, \frac{13}{24}, \frac{5}{8}, \frac{1}{3}\right]$ and $h[0] = 1$.

Step 1: Compute $A_3(z)$

$$A_M(z) = A_3(z) = \frac{H(z)}{h[0]} = \frac{H(z)}{1} = 1 + \frac{13}{24}z^{-1} + \frac{5}{8}z^{-2} + \frac{1}{3}z^{-3}$$

Step 2: Compute k_3

The last coefficients of $A_3(z)$ is $\frac{1}{3}$ so $k_3 = \frac{1}{3}$

Step 3: Compute $B_3(z)$

Once we have $A_M(z)$, we can compute $B_M(z)$ by simply flipping the coefficients of $A_M(z)$.

$$B_M(z) = B_3(z) = \frac{1}{3} + \frac{5}{8}z^{-1} + \frac{13}{24}z^2 + z^{-3}$$

Step 4: Set $m = M$

$$m = 3$$

Step 5: Compute $A_2(z)$

Using the formula $A_{m-1}(z) = \frac{1}{1 - k_m^2} [A_m(z) - k_m B_m(z)]$ we can compute $A_2(z)$

$$A_2(z) = \frac{1}{1 - k_3^2} [A_3(z) - k_3 B_3(z)]$$

$$A_2(z) = \frac{1}{1 - \left(\frac{1}{3}\right)^2} \left[\left(1 + \frac{13}{24}z^{-1} + \frac{5}{8}z^{-2} + \frac{1}{3}z^{-3}\right) - \frac{1}{3} \left(\frac{1}{3} + \frac{5}{8}z^{-1} + \frac{13}{24}z^2 + z^{-3}\right) \right]$$

$$A_2(z) = \frac{9}{8} \left[1 - \frac{1}{3} \cdot \frac{1}{3} + \left(\frac{13}{24} - \frac{1}{3} \cdot \frac{5}{8}\right)z^{-1} + \left(\frac{5}{8} - \frac{1}{3} \cdot \frac{13}{24}\right)z^{-2} + \left(\frac{1}{3} - \frac{1}{3}\right)z^{-3} \right]$$

$$A_2(z) = \frac{9}{8} \left[\frac{8}{9} + \frac{1}{3}z^{-1} + \frac{4}{9}z^{-2} + 0z^{-3} \right]$$

$$A_2(z) = 1 + \frac{3}{8}z^{-1} + \frac{1}{2}z^{-2}$$

Step 6: Compute k_2

We can read off the last coefficient of $A_2(z)$ which is $k_2 = \frac{1}{2}$

Step 7: Compute $B_2(z)$

Once we have $A_2(z)$, we can compute $B_2(z)$ by simply flipping the coefficients of $A_2(z)$.

$$B_2(z) = \frac{1}{2} + \frac{3}{8}z^{-1} + z^{-2}$$

Step 8: Set $m = 2$

Step 9: Go to step 5

Step 5: Compute $A_1(z)$

We can compute $A_1(z)$ as follows

$$A_1(z) = \frac{1}{1 - k_2^2} [A_2(z) - k_2 B_2(z)]$$

$$A_1(z) = \frac{1}{1 - \left(\frac{1}{2}\right)^2} \left[\left(1 + \frac{3}{8}z^{-1} + \frac{1}{2}z^{-2}\right) - \frac{1}{2} \left(\frac{1}{2} + \frac{3}{8}z^{-1} + z^{-2}\right) \right]$$

$$A_1(z) = \frac{4}{3} \left[1 - \frac{1}{2} \cdot \frac{1}{2} + \left(\frac{3}{8} - \frac{1}{2} \cdot \frac{3}{8} \right) z^{-1} + \left(\frac{1}{2} - \frac{1}{2} \right) z^{-2} \right]$$

$$A_1(z) = \frac{4}{3} \left[\frac{3}{4} + \frac{3}{16} z^{-1} + 0 z^{-2} \right]$$

$$A_1(z) = 1 + \frac{1}{4} z^{-1}$$

Step 6: Compute k_1

We can read off the last coefficient of $A_1(z)$ which is $k_1 = \frac{1}{4}$

We are done! The result is $k_1 = \frac{1}{4}$, $k_2 = \frac{1}{2}$ and $k_3 = \frac{1}{3}$

The reflection coefficients are $k_1 = \frac{1}{4}$, $k_2 = \frac{1}{2}$, $k_3 = \frac{1}{3}$

[✓] ADSI Problem 2.5: All-zero lattice filter, find coefficients from system function (division by zero problem)

A filter has the system function

$$H(z) = 1 + 2z^{-1} + z^{-2}$$

- Calculate k_1 and k_2 for the corresponding lattice filter and draw the lattice structure.

```
clear variables;
```

We cannot use a recursive algorithm if one of the reflection coefficients is equal to 1. In this problem, $k_2 = 1$ so if we attempt to compute $A_1(z)$, it will lead to division by zero:

$$A_1(z) = \frac{1}{1 - k_2^2} [A_2(z) - k_2 B_2(z)] = \frac{1}{1 - 1} [A_2(z) - k_2 B_2(z)]$$

We can try out it out in MATLAB function:

```
b = [1, 2, 1];
a = 1;
% tf2latc(b) % This fails because one coeff. is 1
```

Instead we can use a cascade of two single-stage filters.

First, we factorise the original FIR filter to two cascade filters. This can be done by finding the roots and using the formula:

$$H(z) = (1 - r_1 z^{-1})(1 - r_2 z^{-1}) \cdots (1 - r_M z^{-1}) \text{ where } r_i \text{ are the roots of the transfer function.}$$

Find the roots:

roots(b)

```
ans = 2x1
-1
-1
```

Factorise the original FIR filter into two cascading filters:

$$H(z) = (1 - (-1)z^{-1})(1 - (-1)z^{-1}) = (1 + z^{-1})(1 + z^{-1})$$

Recall that that a single stage all-zero filter has following difference equation:

$$y[n] = x[n] + k_1 x[n-1]$$

Taking the z-transform we get:

$$Y_{az}(z) = X_{az}(z) + k_1 X_{az}(z)z^{-1}$$

The system function of a single stage all-zero lattice filter is therefore:

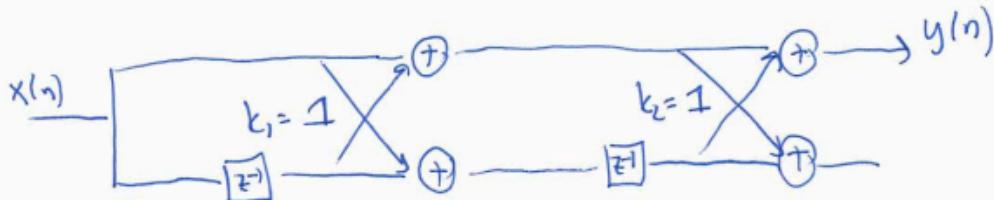
$$H_{az}(z) = \frac{Y_{az}(z)}{X_{az}(z)} = 1 + k_1 z^{-1}$$

It turns out that we can cascade two single stage all-zero lattice filters the same way as we cascade FIR filters.

instead we can factorize $H(z) = A_2(z)$

$$H(z) = (1 + z^{-1})(1 + z^{-1})$$

so we can instead implement as a cascade of two single stages with $k_1 = 1$
and it turns out that it also works in a single two-stage lattice



So in our case, we have $k_1 = 1$ and $k_2 = 1$.

Let us check if we can get the impulse response of the original filter.

```
k = [1, 1];
lat2fir(k, 1)
```

```
ans = 1x3
    1     2     1
```

```
latc2tf(k) % Same as lat2fir
```

```
ans = 1x3
    1     2     1
```

Success!

[✓] ADSI Problem 2.7: Find system function an all-pole filter from reflection coefficients

Determine the system function for an all-pole filter with the reflection coefficients $k_1 = 0.6$, $k_2 = 0.3$, $k_3 = 0.5$ and $k_4 = 0.9$.

```
clear variables;
```

Use MATLAB to compute the transfer function:

```
k = [0.6, 0.3, 0.5, 0.9];
[num, dem] = latc2tf(k, 'allpole')
```

```
num = 1x5
    1     0     0     0     0
dem = 1x5
    1.0000    1.3800    1.3110    1.3370    0.9000
```

The system function is:

$$H(z) = \frac{1}{1 + 1.38z^{-1} + 1.311z^{-2} + 1.337z^{-3} + 0.9z^{-4}}$$

Alternatively, do it by hand using following two formulas:

$$f_{m-1}[n] = f_m[n] - k_m g_{m-1}[n-1]$$

$$g_m[n] = k_m f_{m-1}[n] + g_{m-1}[n-1]$$

We start computing the difference equations of the last section:

$$f_0[n] = f_1[n] - k_1 g_0[n-1]$$

$$g_1[n] = k_1 f_0[n] + g_0[n-1]$$

Then we compute the next stage:

$$f_1[n] = f_2[n] - k_2 g_1[n - 1]$$

$$g_2[n] = k_2 f_1[n] + g_1[n - 1]$$

Next, we can substitute $f_1[n]$ into the expression $f_0[n]$:

$$f_0[n] = f_2[n] - k_2 g_1[n - 1] - k_1 g_0[n - 1]$$

$$f_0[n] = f_2[n] - k_2(k_1 f_0[n - 1] + g_0[n - 2]) - k_1 g_0[n - 1]$$

And so on.

Finally, we will get a large expression in which we replace:

- $g_0[n]$ and $f_0[n]$ with $y[n]$
- $f_4[n]$ with $x[n]$

[✓] ADSI Problem 2.10: Linear prediction and lattice filters

The idea behind linear prediction is than the signal $\hat{x}(n)$ can be estimated as a weighted linear combination of the p previous samples

$$\hat{x}(n) = - \sum_{k=1}^p a_p(k) x(n - k)$$

```
clear variables;
```

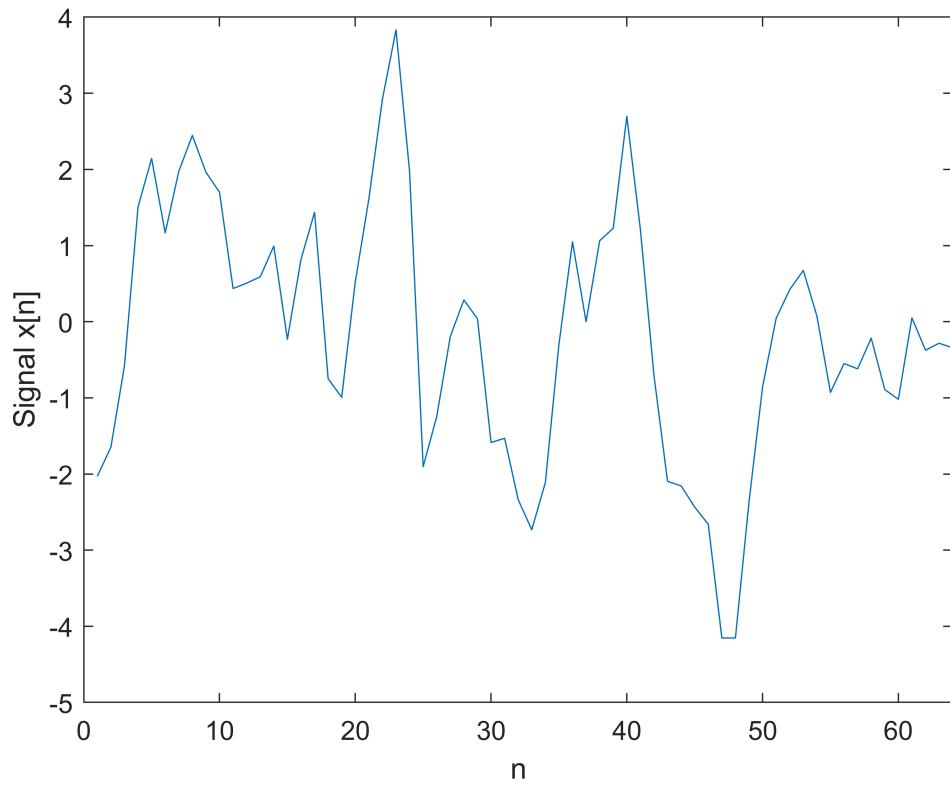
1) Compute autocorrelation from a signal

The file `signal1.dat` contains 64 samples from an artificial signal.

1. Load `signal1.dat` into MATLAB. Create an autocorrelation of the signal using `xcorr` and plot it. What does the autocorrelation tell you about the signal?

Let us first plot the signal.

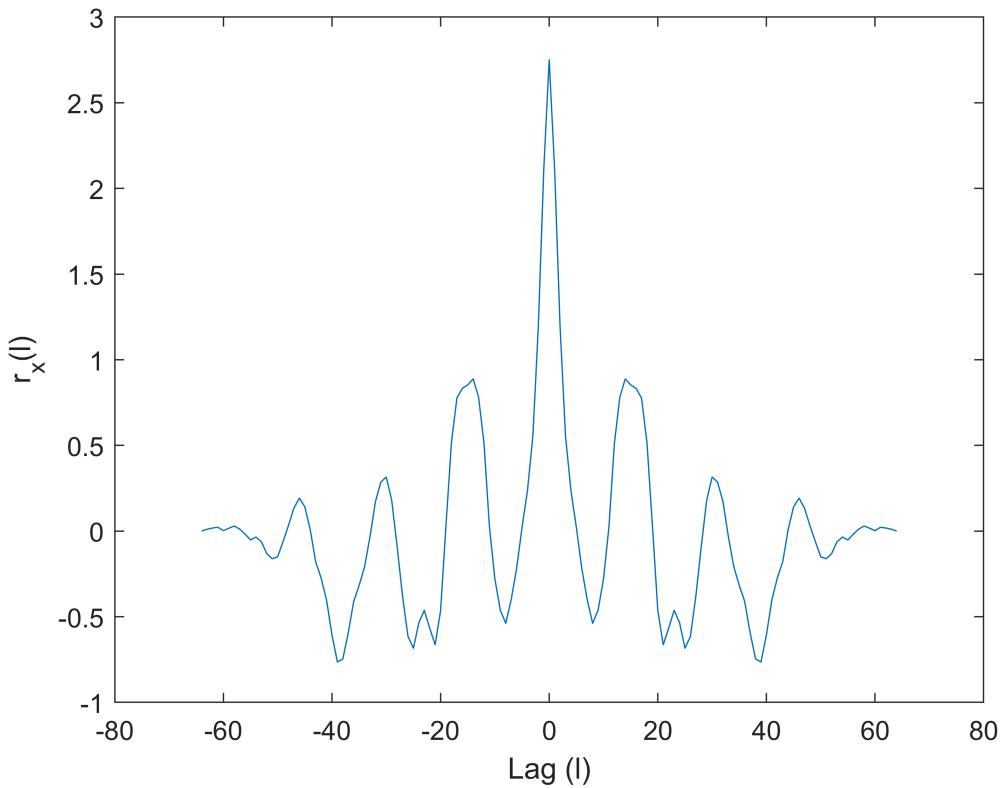
```
x = importdata('signal1.dat');
plot(x);
xlabel('n');
ylabel('Signal x[n]');
xlim([0, numel(x)]);
```



Autocorrelation is the cross-correlation between a signal $x[n]$ with the same signal shifted by some amount $x[n - k]$. So if we start with $k = 1$, we compute the correlation between $x[n]$ and $x[n - 1]$. For $k = 2$, we compute the correlation between $x[n]$ and $x[n - 2]$ etc.

Autocorrelation is symmetrical around zero.

```
[r,lags] = xcorr(x,64, 'biased');
plot(lags,r);
xlabel('Lag (1)');
ylabel('r_x(1)');
```



From the plot, we see that when the signal is shifted e.g. by 4, the correlation is low.

```
r(4)
```

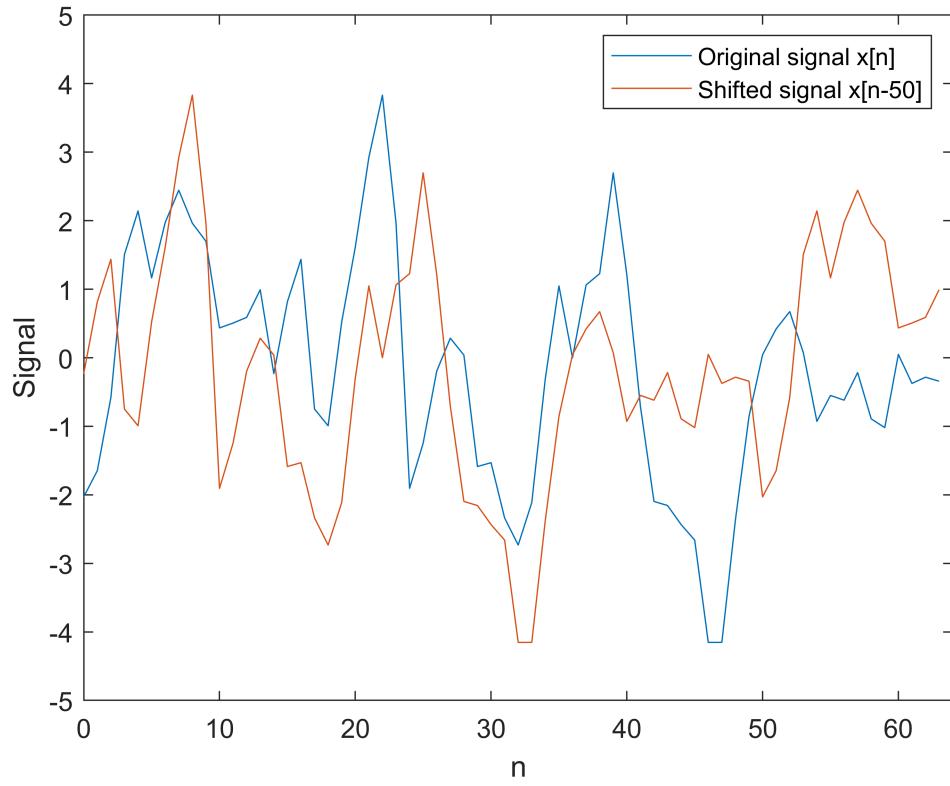
```
ans = 0.0223
```

```
r(50)
```

```
ans = 0.8533
```

This means that the original signal $x[n]$ and the shifted signal $x[n - 4]$ are not very similar. The correlation is higher when the signal is shifted by 50:

```
n = linspace(0, 63, 64)';
x_shifted_50 = circshift(x, 50);
plot(n, x, n, x_shifted_50);
legend('Original signal x[n]', 'Shifted signal x[n-50]')
xlabel('n');
ylabel('Signal');
xlim([0, numel(x)]);
ylim([-5, 5])
```



Shifting the signal by 64 yields the highest correlation. This makes sense because if we shift a signal of length 64 by 64 then we get the original signal $x[n] = x[n - 64]$.

`r(64+1)`

ans = 2.7507

2) Compute the coefficients for the optimum 2nd-order linear predictor

Based on the autocorrelation values we can calculate the coefficients for the optimum p th order linear predictor with the normal equations (which we will derive later in the course)

$$r_x(l) = - \sum_{k=1}^p a_p(k) r_x(l-k), \quad l = 1, \dots, p$$

Let $p = 2$.

2. Compute $a_2(1)$ and $a_2(2)$.

When $p = 2$ then we get following two equations:

$$r_x(1) = -a_2(1)r_x(0) - a_2(2)r_x(-1)$$

$$r_x(2) = -a_2(1)r_x(1) - a_2(2)r_x(0)$$

In exercise 1) we computed the correlation values r_x . So we know the values for $r_x(-1)$, $r_x(0)$, $r_x(1)$ and $r_x(2)$. This means that we have to solve two equations with two unknowns.

Let us rewrite it:

$$r_x(1) + a_2(1)r_x(0) + a_2(2)r_x(-1) = 0$$

$$r_x(2) + a_2(1)r_x(1) + a_2(2)r_x(0) = 0$$

Solve it in MATLAB:

```
syms a2_1 a2_2

% The autocorrelation is centered around zero
% Find the index where n=0
centerIx = find(lags==0);

% Retrieve the values r(-1), r(0), r(1) and r(2) in order
rx_m1 = r(centerIx - 1);
rx_0 = r(centerIx + 0);
rx_1 = r(centerIx + 1);
rx_2 = r(centerIx + 2);

eq1 = rx_1 + a2_1*rx_0 + a2_2*rx_m1;
eq2 = rx_2 + a2_1*rx_1 + a2_2*rx_0;

sol = solve([eq1, eq2], [a2_1, a2_2]);

a1 = sol.a2_1;
a2 = sol.a2_2;
% Print the value of a2(1)
vpa(sol.a2_1)
```

ans = -1.0554484225972357733576990520746

```
% Print the value of a2(2)
vpa(sol.a2_2)
```

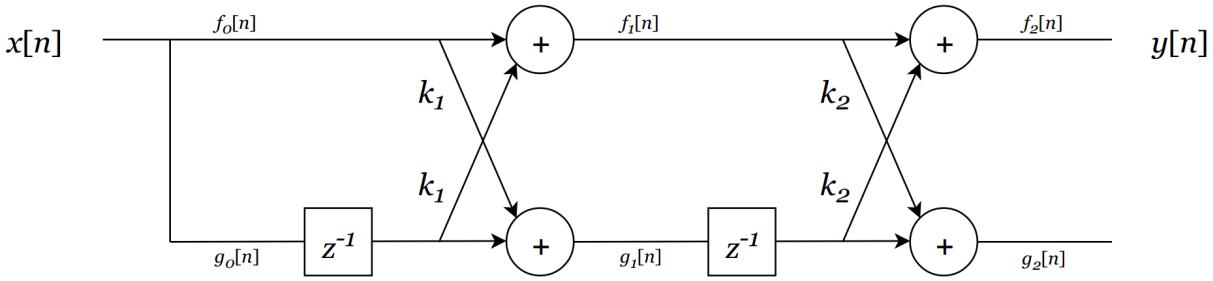
ans = 0.37530590522885965170506927041028

The coefficients of the predictor are $a_2(1) \approx -1.0554$ and $a_2(2) \approx 0.3753$

3) Compare the linear predictor with a two-stage all-zero lattice filter

Next, consider a two-stage all-zero lattice filter.

3. What is the relation between the output of the filter, $f_2(n)$, and the above linear prediction?



We know that the output of a two-stage all-zero lattice filter is:

$$y[n] = x[n] + k_1(1 + k_2)x[n - 1] + k_2x[n - 2]$$

The p'th order linear predictor has the following form:

$$\hat{x}[n] = \sum_{k=1}^p a_k \cdot [n - k]$$

So the second-order linear predictor has the form:

$$\hat{x}[n] = a_1x[n - 1] + a_2x[n - 2]$$

Notice that 2nd order linear predictor looks very much like the difference equation for the two-stage all-zero lattice filter.

There is a one-to-one correspondance between their coefficients where $a_1 = k_1(1 + k_2)$ and $a_2 = k_2$.

4) Compute reflection coefficients from the linear predictor

- Compute the reflection coefficients k_1 and k_2 from the $a_2(1)$ and $a_2(2)$ values found above.

From 3) we found that $a_1 = k_1(1 + k_2)$ and $a_2 = k_2$ so:

$$k_2 = a_2$$

$$k_1 = \frac{a_1}{1 + a_2}$$

In 2) we found that values for a_1 and a_2 . We will use these to compute k_1 and k_2 :

```
k1 = a2/(1+a2);
k2 = a2;
vpa(k1)
```

```
ans = 0.27288903785111455734521239333629
```

```
vpa(k2)
```

```
ans = 0.37530590522885965170506927041028
```

So we have $k_1 \approx 0.272889$ and $k_2 \approx 0.3753$

5) Compute prediction error

5. Send the signal through a 2nd order all-zero lattice filter and analyze the prediction error. Is the signal predictable? Repeat for a higher order predictor.

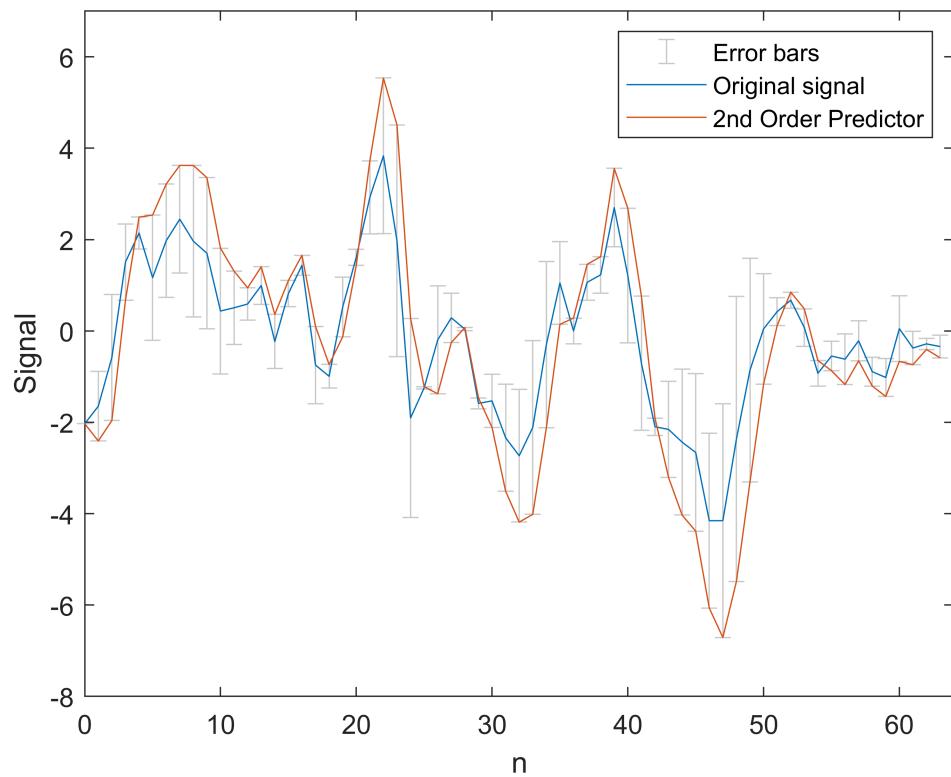
```
figure;

% Compute x_hat and err
n = linspace(0, 63, 64);
x = importdata('signal1.dat');
x_hat = azlatfilt([k1, k2], x, 1);
err = x_hat - x;

% Plot
errorbar(n, x, err, 'LineStyle','none', 'Color', [0.8, 0.8, 0.8]);

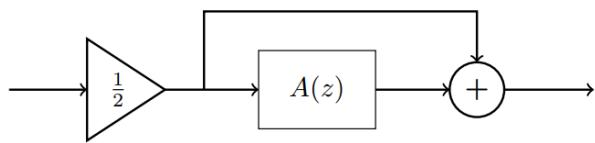
hold on;
plot(n, x);
plot(n, x_hat);
hold off;

xlabel('n');
ylabel('Signal');
xlim([0, numel(x)]);
ylim([-8, 7]);
legend('Error bars', 'Original signal', '2nd Order Predictor');
```



Exam 2016 Problem 3: All-pole Lattice Filter

Consider the digital signal processing system shown below where $A(z)$ is a 2nd order all-pass filter.



1) Explain why a system is a digital notch filter

Since $A(z)$ is an all-pass filter, all frequencies pass un-attenuated but a frequency dependent phase shift is imposed on the signal.

Assume an input signal $x(n) = \alpha \cos(\omega n)$. If the two signals at the summing point are in phase the output is just equal to the input. For any other phase shift, the two signals will partly cancel out with the degree of cancellation depending on the phase shift.

Since $A(z)$ is an all-pass filter all frequencies pass un-attenuated but a frequency dependent phase shift is imposed on the signal. Assume an input signal $x(n) = \alpha \cos(\omega n)$. If the two signals at the summing point are in phase the output is just equal to the input. For any other phase shift, the two signals will partly cancel out with the degree of cancellation depending on the phase shift. For a phase shift of $(2z + 1)\pi$, $z \in \mathbb{Z}$ the two signals will be equal but with opposite sign and the output becomes zero giving the notch filter. Mathematically, this is seen from

$$y(n) = \frac{1}{2}\alpha \cos(\omega n) + \frac{1}{2}\alpha \cos(\omega n + \phi) = \alpha \cos\left(\frac{\phi}{2}\right) \cos\left(\omega n + \frac{\phi}{2}\right).$$

2) Find reflection coefficients for the all-pole lattice filter

The all-pass filter can be implemented as an all-pole lattice filter. Assume that

$$A(z) = \frac{0.3 + 0.91z^{-1} + z^{-2}}{1 + 0.91z^{-1} + 0.3z^{-2}}.$$

2. Find $A_2(z)$, $A_1(z)$ and the reflection coefficients for the all-pole lattice filter.

The recursive algorithm works as follows:

1. Compute $A_M(z) = \frac{H(z)}{h[0]}$
2. Compute $k_M = a_M$ where a_M is the last coefficients of $A_M(z)$.
3. Compute $B_M(z)$ by flipping the coefficients of $A_M(z)$
4. Set $m = M$
5. Compute $A_{m-1}(z) = \frac{1}{1 - k_m^2} [A_m(z) - k_m B_m(z)]$.
6. Compute $k_{m-1} = a_{m-1}$ where a_{m-1} is the last coefficients of $A_{m-1}(z)$
7. Compute $B_{m-1}(z)$ by flipping the coefficients of $A_{m-1}(z)$. Alternatively, compute $B_{m-1}(z) = z^{-m-1} A_{m-1}\left(\frac{1}{z}\right)$
8. Set $m = m - 1$
9. Go to step 5 if $m \neq 0$
10. We know that $A_0(z) = B_0(z) = 1$

the difference equations for an all-pole system and an all-zero system with

$$H_{\text{ap}}(z) = \frac{Y(z)}{X(z)} = \frac{1}{A_2(z)}, \quad H_{\text{az}}(z) = \frac{G_2(z)}{Y(z)} = z^{-2}A_2(1/z), \quad (9.84)$$

where

$$A_2(z) \triangleq 1 + a_1^{(2)}z^{-1} + a_2^{(2)}z^{-2}. \quad (9.85)$$

From the system function of the all-pass filter, we know that:

$$A_2(z) = 1 + 0.91z^{-1} + 0.3z^{-2} \text{ and } k_2 = a_2^{(2)} = 0.3$$

We compute B_2 :

$$B_2 = 0.3 + 0.91z^{-1} + z^{-2}$$

We compute A_1 :

$$A_1(z) = \frac{1}{1 - k_2^2} [A_2(z) - k_2 B_2(z)]$$

$$A_1(z) = \frac{1}{1 - 0.3^2} [(1 + 0.91z^{-1} + 0.3z^{-2}) - 0.3(0.3 + 0.91z^{-1} + z^{-2})]$$

$$A_1(z) = \frac{1}{1 - 0.3^2} [1 + 0.91z^{-1} + 0.3z^{-2} - 0.09 - 0.2730z^{-1} - 0.3z^{-2}]$$

$$A_1(z) = 1.0989[0.91 + 0.6370z^{-1}]$$

$$A_1(z) = 1 + 0.7z^{-1}$$

with $k_1 = a_1^{(1)} = 0.7$

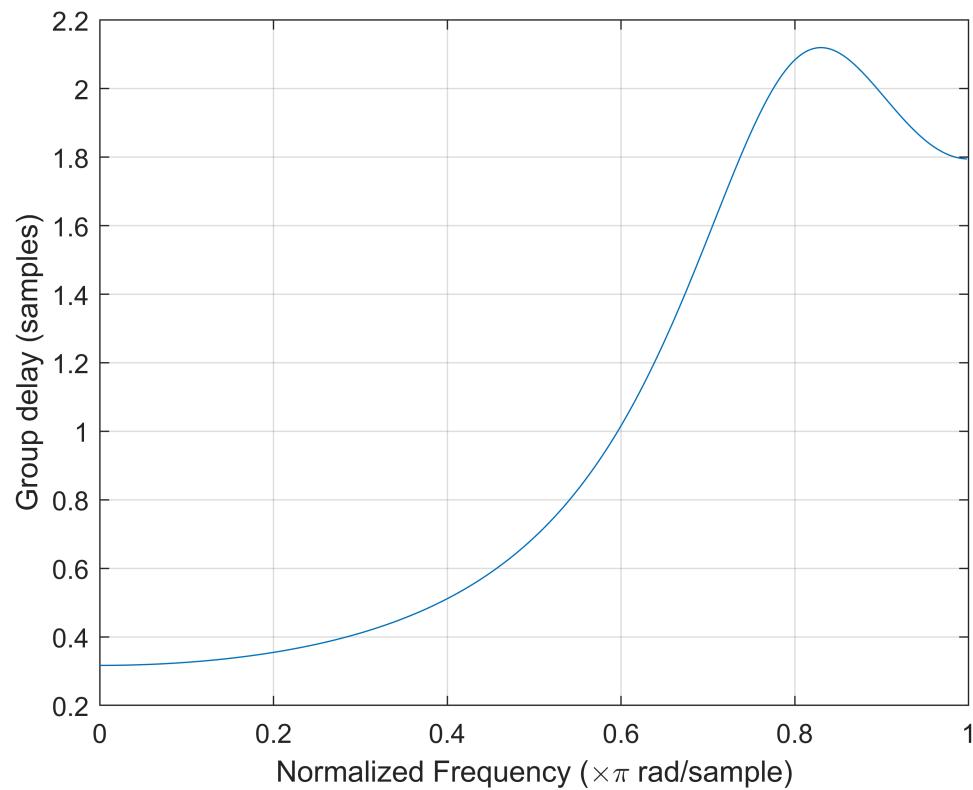
3) Determine the group delay in the digital notch filter

The transfer function of the entire system is needed to determine the group delay

$$\begin{aligned} H(z) &= \frac{1}{2} \left(1 + \frac{0.3 + 0.91z^{-1} + z^{-2}}{1 + 0.91z^{-1} + 0.3z^{-2}} \right) \\ &= \frac{1}{2} \left(\frac{1 + 0.91z^{-1} + 0.3z^{-2}}{1 + 0.91z^{-1} + 0.3z^{-2}} + \frac{0.3 + 0.91z^{-1} + z^{-2}}{1 + 0.91z^{-1} + 0.3z^{-2}} \right) \\ &= \frac{0.65 + 0.91z^{-1} + 0.65z^{-2}}{1 + 0.91z^{-1} + 0.3z^{-2}}. \end{aligned}$$

The group delay is then found using Matlab:

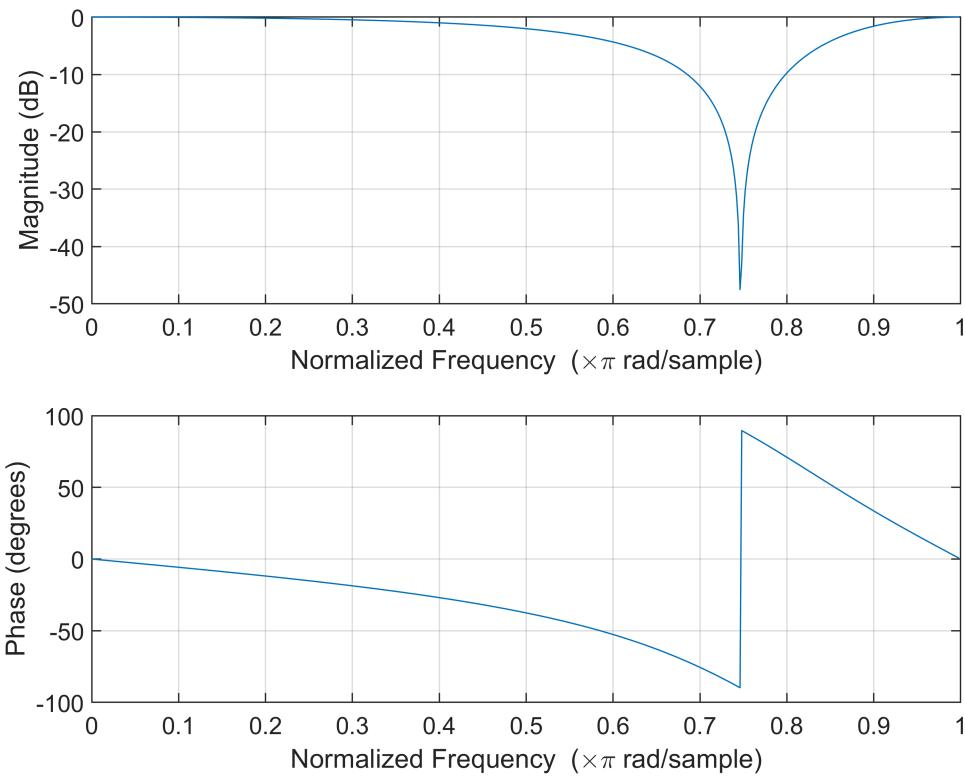
```
grpdelay([0.65 0.91 0.65],[1 0.91 0.3])
```



From which it is seen that the group delay at $\omega = \frac{\pi}{4}$ is 0.3788.

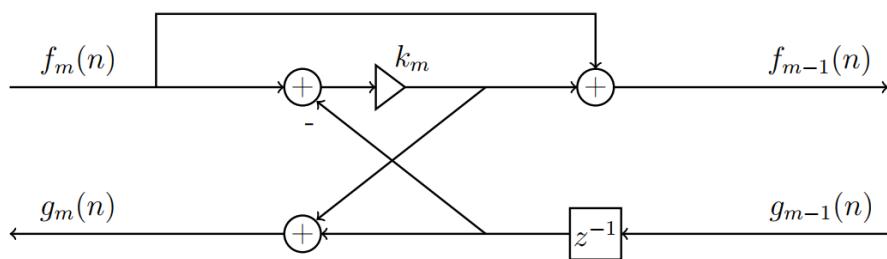
For completeness, the performance of the notch filter can be plotted with freqz.

```
freqz([0.65 0.91 0.65],[1 0.91 0.3])
```



Exam 2017 Problem 2: Lattice Structures

Many different lattice structure have been investigated over the years, often tailored to specific requirements. One particular example is based on minimizing hardware requirements and uses only one multiplication per stage. The structure of the m 'th stage of a one-multiplier lattice filter is shown below. The input and output is connected as for standard all-pole filters.



```
clear variables;
```

- 1) Determine the two equations relating outputs to inputs

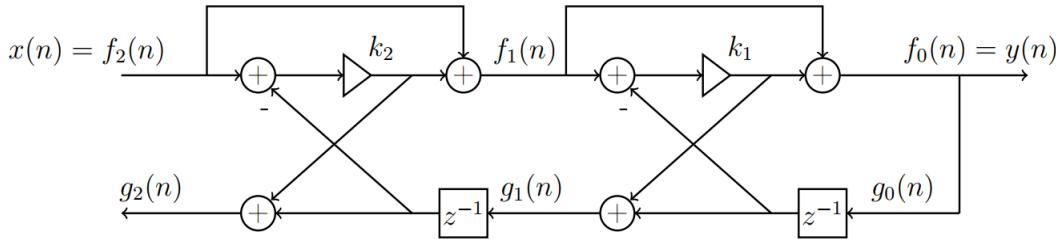
1. Determine the two equations relating outputs $f_{m-1}(n)$ and $g_m(n)$ to inputs $f_m(n)$ and $g_{m-1}(n)$.

$$\begin{aligned} f_{m-1}(n) &= f_m(n) + k_m(f_m(n) - g_{m-1}(n)) \\ &= f_m(n) + k_m f_m(n) - k_m g_{m-1}(n-1) \\ &= (1 + k_m)f_m(n) - k_m g_{m-1}(n-1) \\ g_m(n) &= g_{m-1}(n-1) + k_m(f_m(n) - g_{m-1}(n)) \\ &= g_{m-1}(n-1) + k_m f_m(n) - k_m g_{m-1}(n) \\ &= k_m f_m(n) + (1 - k_m)g_{m-1}(n-1) \end{aligned}$$

2) When is the first non-zero sample observed at the output?

2. If the input of a two-stage, one-multiplier lattice filter with reflection coefficients k_1 and k_2 is excited by an impulse, $x(n) = \delta(n)$, when is the first non-zero sample observed at the output $y(n)$?

The two-stage, one multiplier lattice looks like this:



We found that the equation relating output to input is:

$$f_{m-1}(n) = (1 + k_m)f_m(n) - k_m g_{m-1}(n-1)$$

Since we want the first output, we do not concern ourselves with the $g_{m-1}(n-1)$.

First, we compute $f_1(n)$:

$$f_1(n) = (1 + k_2)f_2(n) = (1 + k_2)x(n)$$

Next, we compute $f_0(n)$:

$$f_0(n) = (1 + k_1)f_1(n) = (1 + k_1)(1 + k_2)x(n) = y(n)$$

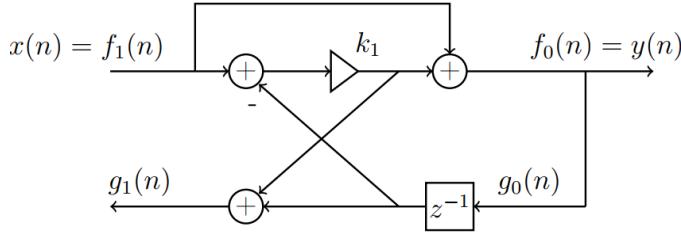
Since there is a direct path without any delays from input $x(n)$ to the output $y(n)$, the first non-zero sample appears instantaneously at the output.

$$y(n) = (1 + k_1)(1 + k_2)x(n)$$

3) Calculate the transfer function for a one-stage lattice filter

3. Calculate the transfer function $H_{G_1}(z) = G_1(z)/X(z)$ for a one-stage, one-multiplier, lattice filter with reflection coefficients k_1 and show that it corresponds to an all-pass filter.

The one-stage, one multiplier lattice looks like this.



The equations relating the inputs and outputs are:

$$f_0(n) = (1 + k_1)f_1(n) - k_1g_0(n - 1)$$

$$g_1(n) = k_1f_1(n) + (1 - k_1)g_0(n - 1)$$

To calculate transfer functions, the two equations are first z -transformed:

$$F_0(z) = (1 + k_1)F_1(z) - k_1z^{-1}G_0(z) \quad \text{Eq. (1)}$$

$$G_1(z) = k_1F_1(z) + (1 - k_1)z^{-1}G_0(z) \quad \text{Eq. (2)}$$

We know that $F_1(z) = X(z)$ and $F_0(z) = G_0(z) = Y(z)$.

We use the identity $F_0(z) = G_0(z)$ to rewrite our Eq. (1):

$$G_0(z) = (1 + k_1)F_1(z) - k_1z^{-1}G_0(z)$$

$$G_0(z) + k_1z^{-1}G_0(z) = (1 + k_1)F_1(z)$$

$$(1 + k_1z^{-1})G_0(z) = (1 + k_1)F_1(z)$$

$$G_0(z) = \frac{(1 + k_1)F_1(z)}{1 + k_1z^{-1}}$$

Substitutde the new expression into Eq. (2)

$$G_1(z) = k_1F_1(z) + \frac{(1 - k_1)(1 + k_1)z^{-1}}{1 + k_1z^{-1}}F_1(z)$$

We know that $F_1(z) = X(z)$:

$$G_1(z) = k_1X(z) + \frac{(1 - k_1)(1 + k_1)z^{-1}}{1 + k_1z^{-1}}X(z)$$

$$G_1(z) = X(z) \left(k_1 + \frac{(1 - k_1)(1 + k_1)z^{-1}}{1 + k_1 z^{-1}} \right)$$

$$\frac{G_1(z)}{X(z)} = k_1 + \frac{(1 - k_1)(1 + k_1)z^{-1}}{1 + k_1 z^{-1}}$$

$$\frac{G_1(z)}{X(z)} = \frac{k_1(1 + k_1 z^{-1})}{1 + k_1 z^{-1}} + \frac{(1 - k_1)(1 + k_1)z^{-1}}{1 + k_1 z^{-1}}$$

$$\frac{G_1(z)}{X(z)} = \frac{k_1(1 + k_1 z^{-1}) + (1 - k_1)(1 + k_1)z^{-1}}{1 + k_1 z^{-1}}$$

```
syms k z
expand(k*(1+k*z)+(1-k)*(1+k)*z)
```

$$\text{ans} = k + z$$

$$\frac{G_1(z)}{X(z)} = \frac{k_1 + z^{-1}}{1 + k_1 z^{-1}}$$

An allpass filter has the form:

$$H_k(z) = z^{-1} \frac{1 - p_k^* z}{1 - p_k z^{-1}} = \frac{z^{-1} - p_k^*}{1 - p_k z^{-1}} \quad (5.157)$$

Higher order allpass systems can be obtained by cascading multiple first-order sections, as

$$H_{\text{ap}}(z) = e^{j\beta} \prod_{k=1}^N \frac{z^{-1} - p_k^*}{1 - p_k z^{-1}}, \quad (5.158)$$

Exam 2014 Problem 1

Functions

```

function [k,G] = fir2lat(h)
% Converts FIR filter coefficients to lattice coefficients.
% From Figure 9.24 (p. 514)
G = h(1);
a = h/G;
M = length(h)-1;
k(M) = a(M+1);
for m = M:-1:2
    b = fliplr(a);
    a = (a-k(m)*b)/(1-k(m)^2); a = a(1:m);
    k(m-1) = a(m);
end
end

function [h] = lat2fir(k, G)
% Converts lattice coefficients to FIR filter coefficients.
% From Figure 9.25 (p. 515)
a = 1;
b = 1;
M = length(k);
for m = 1:1:M
    a = [a,0]+k(m)*[0,b];
    b = fliplr(a);
end
h = G*a;
end

function [y] = azlatfilt(k, x, G)
% AZLATFILT      Filters x with an all-zero lattice filter with
%                 coefficients k. From Figure 9.26 (p 515)

M = length(k);

% Create an array for the sequence f_m[n]
f = zeros(1,M);

% Create an array for the sequence g_m[n]
g = f;

% Create an array for the sequence g_m[n-1]
oldg = zeros(1,M);

% Keeps track of x[n-1]
oldx = 0;
x = G*x;
y = zeros(size(x));

for n=1:length(x)
    % Compute f1[n] = x[n] + k1x[n-1] -> Equation (9.57a)
    f(1) = x(n)+k(1)*oldx;

    % Compute g1[n] = k1x[n] + x[n-1] -> Equation (9.57b)
    g(1) = k(1)*x(n)+oldx;
    oldx = x(n);

```

```

for m = 2:M
    % Compute: fm[n] = fm?1[n] + km gm?1[n ? 1] -> Equation (9.55a)
    f(m) = f(m-1)+k(m)*oldg(m-1);

    % Compute: gm[n] = km fm?1[n] + gm?1[n ? 1] -> Equation (9.55b)
    g(m) = k(m)*f(m-1)+oldg(m-1);

    % Delay
    oldg(m-1) = g(m-1);
end

% y[n] = fM[n] -> Equation (9.56b)
y(n) = f(M);
end
end

function y = aplatfilt(k,G,x)
% All-pole Lattice Filter Implementation
% From Figure 9.28

M = length(k); g = zeros(1,M); f = g;
oldy = 0; x = G*x; y = zeros(size(x));
for n = 1:length(x)
    f(M) = x(n);
    for i = 1:M-1
        m = M+1-i;
        f(m-1) = f(m)-k(m)*g(m-1);
        g(m) = k(m)*f(m-1)+g(m-1);
    end
    y(n) = f(1)-k(1)*oldy;
    g(1) = k(1)*y(n)+oldy;
    oldy = y(n);
end
end

```

Linear Predictors

Table of Contents

Forward Linear Predictors.....	1
All-pole signal modelling.....	2
[✓] ADSI Problem 6.4: Linear interpolation, estimate missing samples.....	3
1) Use mean squared error to derive coefficients based on the ACRS.....	3
[✓] ADSI Problem 6.6: Levinson-Durbin and linear prediction.....	4
[✓] 1) Plot the autocorrelation function.....	5
[✓] 2) Compute reflection coefficients for m'th order optimum linear predictors.....	5
[?] ADSI Problem 6.7: Linear prediction of a sine signal.....	7
[✓] 1) Determine the autocorrelation function for $x(n)$	8
[?] 2) Determine the 2nd order forward linear prediction filter.....	8
[>] 3) Find the system function for the filter and locate the zeros.....	9
[>] 4) Determine the frequency response, plot it and comment on the result.....	9
[>] 5) Calculate the prediction error.....	10
[?] ADSI Problem 6.8: Autocorrelation function and linear prediction.....	10

Forward Linear Predictors

Problem: we want predict the current signal value $\hat{x}[n]$ given a set of p previous samples of a wide-sense stationary process:

$$\hat{x}[n] = \sum_{k=1}^p h_k x[n-k] = \mathbf{h}^T \mathbf{x}[n-1], \quad (14.122)$$

This is called one-step *forward linear predictor*.

The normal equations for the optimum linear predictor are:

$$\mathbf{R}\mathbf{h} = \mathbf{r}, \quad (14.125)$$

where \mathbf{R} , which is a symmetric Toeplitz matrix, and the vector \mathbf{r} are defined by

$$\mathbf{R} \triangleq \begin{bmatrix} r[0] & r[1] & \dots & r[p-1] \\ r[1] & r[0] & \dots & r[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r[p-1] & r[p-2] & \dots & r[0] \end{bmatrix} \quad \text{and} \quad \mathbf{r} \triangleq \begin{bmatrix} r[1] \\ r[2] \\ \vdots \\ r[p] \end{bmatrix}. \quad (14.126)$$

The minimum squared error is given by Eq. 14.127:

$$J_0 = r[0] - \mathbf{h}^T \mathbf{r} = r[0] - \mathbf{r}^T \mathbf{R}^{-1} \mathbf{r}. \quad (14.127)$$

All-pole signal modelling

Suppose the input signal $x[n]$ is an AR(p) model i.e.:

$$x[n] = - \sum_{k=1}^p a_k x[n-k] + z[n] = -\mathbf{a}^T \mathbf{x}[n-1] + z[n], \quad (14.128)$$

where $z[n] \sim \text{WN}(0, \sigma_z^2)$

The value that we want to predict $x[n]$ can be written as:

$$x[n] = \sum_{k=1}^p h_k x[n-k] + e[n] = \mathbf{h}^T \mathbf{x}[n-1] + e[n]. \quad (14.131)$$

Setting $\mathbf{h} = -\mathbf{a}$, the prediction error can be expressed as:

$$e[n] = x[n] + \sum_{k=1}^p a_k x[n-k] = x[n] + \mathbf{a}^T \mathbf{x}[n-1], \quad (14.132)$$

This shows that the prediction error $e[n]$ is the output of a filter with the following system function:

$$A(z) = 1 + \sum_{k=1}^p a_k z^{-k}. \quad (14.133)$$

This system $A(z)$ is known as the ***prediction error filter*** or the ***analysis filter***.

If the input $x[n]$ is an AR(p) process, the output of $A(z)$ is a white noise process $z[n]$

[✓] ADSI Problem 6.4: Linear interpolation, estimate missing samples

Sometimes it happens that a datapoint is missing from some signal acquisition due to sensor failure, transmission errors etc. Assume that we have a long stationary sequence $\{x[n]\}_{n=0}^{N-1}$ where the j 'th sample is missing i.e.

$$\{x[n]\} = \{x[0], x[1], \dots, x[j-1], x[j+1], x[j+2], \dots, x[N-2], x[N-1]\}$$

We want estimate the missing datapoint as a linear combination of the two neighbouring samples

$$\hat{x}[j] = c_1 x[j-1] + c_2 x[j+1]$$

1. Use our standard mean square error approach to derive equations for c_1 and c_2 based on the autocorrelation $r_{xx}(l)$.

1) Use mean squared error to derive coefficients based on the ACRS

We want to estimate a missing sample $x(j)$ as a linear combination of two neighbouring samples:

$$\hat{x}(j) = c_1 x(j-1) + c_2 x(j+1)$$

To find the coefficients c_1 and c_2 using the mean squared error method, we need to take following steps:

1. Find an expression for the error: $e(n)$
2. Square the error quantity: $e^2(n)$
3. Take the expectation of squared error: $E[e^2(n)]$ to find the Mean Squared Error
4. Find the minimum by setting the partial derivatives of MSE to zero and solving the equation

Step 1: The error of the estimate is:

$$e(j) = x(j) - \hat{x}(j)$$

$$e(j) = x(j) - (c_1 x(j-1) + c_2 x(j+1))$$

$$e(j) = x(j) - c_1 x(j-1) - c_2 x(j+1)$$

Step 2: square the error

$$e^2(j) = (x(j) - c_1 x(j-1) - c_2 x(j+1))(x(j) - c_1 x(j-1) - c_2 x(j+1))$$

```
syms c1 c2 x_j x_jm1 x_jp1
expand((x_j - c1*x_jm1 - c2*x_jp1)^2)
```

$$\text{ans} = c_1^2 x_{jm1}^2 + 2 c_1 c_2 x_{jm1} x_{jp1} - 2 c_1 x_j x_{jm1} + c_2^2 x_{jp1}^2 - 2 c_2 x_j x_{jp1} + x_j^2$$

$$e^2(j) = c_1^2 x^2(j-1) + 2c_1 c_2 x(j-1)x(j+1) - 2c_1 x(j)x(j-1) + c_2^2 x^2(j+1) - 2c_2 x(j)x(j+1) + x^2(j)$$

Step 3: Take the expectation of the squared error:

$$E[e^2(j)] = E[c_1^2 x^2(j-1) + 2c_1 c_2 x(j-1)x(j+1) - 2c_1 x(j)x(j-1) + c_2^2 x^2(j+1) - 2c_2 x(j)x(j+1) + x^2(j)]$$

$$E[e^2(j)] = c_1^2 E[x^2(j-1)] + 2c_1 c_2 E[x(j-1)x(j+1)] - 2c_1 E[x(j)x(j-l)]$$

$$+ c_2^2 E[x^2(j+1)] - 2c_2 E[x(j)x(j+1)] + E[x^2(j)]$$

From the expression, we observe different autocorrelation values:

$$\begin{aligned} E[e^2[n]] &= r_x[0] - c_1 r_x[1] - c_2 r_x[1] \\ &\quad - c_1 r_x[1] + c_1^2 r_x[0] + c_1 c_2 r_x[2] \\ &\quad - c_2 r_x[1] + c_1 c_2 r_x[2] + c_2^2 r_x[0] \end{aligned}$$

Step 4: The minimum is found by taking the partial derivatives, setting it zero and solving the two equations:

$$\frac{\partial E[e^2[n]]}{\partial c_1} = 0, \quad \frac{\partial E[e^2[n]]}{\partial c_2} = 0,$$

The partial derivatives become:

$$\begin{aligned} -2r_x[1] + 2c_1 r_x[0] + 2c_2 r_x[2] &= 0 \\ -2r_x[1] + 2c_1 r_x[2] + 2c_2 r_x[0] &= 0 \end{aligned}$$

Which can be written as a matrix equation:

$$\begin{bmatrix} r_x[0] & r_x[2] \\ r_x[2] & r_x[0] \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} r_x[1] \\ r_x[1] \end{bmatrix}$$

Working through the equations the solution is:

$$c_1 = c_2 = \frac{r_x[1](r_x[0] - r_x[2])}{r_x^2[0] - r_x^2[2]}$$

[✓] ADSI Problem 6.6: Levinson-Durbin and linear prediction

The autocorrelation function of an AR(2) process with two complex conjugated poles at $p = r_p e^{\pm j\omega_p}$ can be calculated analytically and is given by

$$r_{xx}(l) = \frac{r_p^l (\sin((l+1)\omega_p) - r_p^2 \sin((l-1)\omega_p))}{(1 - r_p^2) \sin(\omega_p)(1 - 2r_p^2 \cos(2\omega_p) + r_p^4)} \quad \text{for } l \geq 0$$

Assume that $r_p = 0.9$ and $\omega_p = \pi/16$.

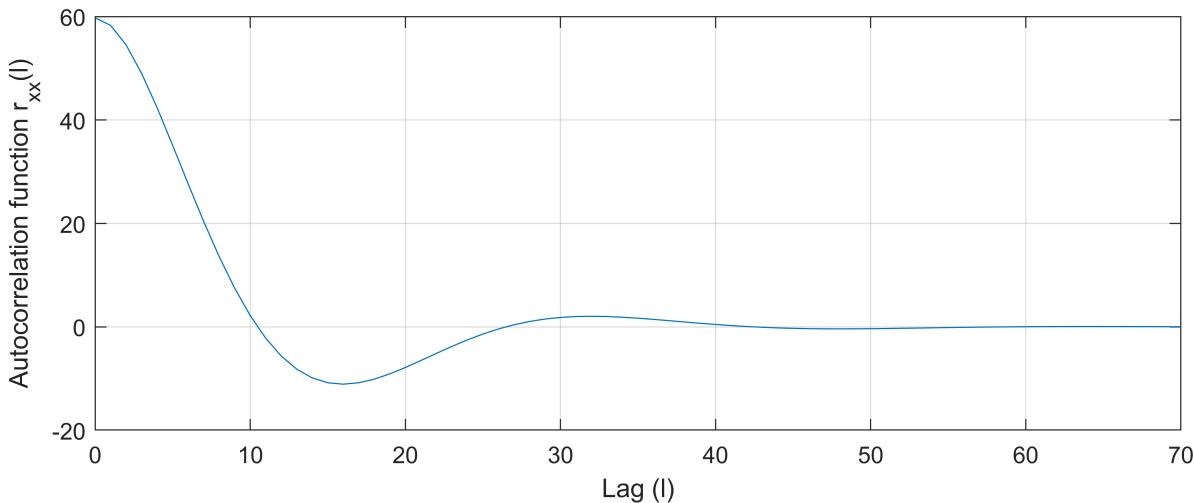
[✓] 1) Plot the autocorrelation function.

```
rp = 0.9;
wp = pi/16;

l = 0:70;

nominator = rp.^l.* (sin((l + 1)*wp) - rp^2*sin((l - 1)*wp));
denominator = (1 - rp^2) * sin(wp) * (1 - 2*rp^2*cos(2*wp) + rp^4);
r_xx = nominator / denominator;

figure('position', [0, 0, 800, 300])
plot(l, r_xx)
xlabel('Lag (l)')
ylabel('Autocorrelation function r_{xx}(l)')
grid on;
```



[✓] 2) Compute reflection coefficients for m'th order optimum linear predictors

Use the above autocorrelation function and the Levinson-Durbin recursion to calculate reflection coefficients and minimum mean square errors for m 'th order optimum linear predictors for $m = 1$ to $m = 6$. Are the results in agreement with your anticipations and Eq. (14.149)?

First, we use the MATLAB function `levinson(r_xx, m)` to compute the reflection coefficients and the corresponding errors for the different m 'th order linear predictors:

```
M = 6;
err = zeros(M, 1); % Array to store the errors

for m=1:M
    [a, e, k] = levinson(r_xx, m);
    % `a` is the coefficients of the AR(m) model
    % `e` is the prediction error of m'th order optimum linear predictor
    % `k` is the reflection coefficients of the linear predictor
```

```

err(m) = e;
display(strcat('Coefficients for a ', 32, num2str(m), '-th order predictor', 32))
display(k)
end

Coefficients for a 1-th order predictor
k = -0.9754
Coefficients for a 2-th order predictor
k = 2x1
-0.9754
 0.8100
Coefficients for a 3-th order predictor
k = 3x1
-0.9754
 0.8100
 -0.0000
Coefficients for a 4-th order predictor
k = 4x1
-0.9754
 0.8100
 -0.0000
 -0.0000
Coefficients for a 5-th order predictor
k = 5x1
-0.9754
 0.8100
 -0.0000
 -0.0000
 0.0000
Coefficients for a 6-th order predictor
k = 6x1
-0.9754
 0.8100
 -0.0000
 -0.0000
 0.0000
 -0.0000

```

It is important to note that the reflection coefficients $\{k_3, k_4, k_5, k_6\}$ are zero. This is to be expected as the reflection coefficients is dependent on the AR(q) model. Since we have an AR(2) model, the corresponding reflection coefficients become zero.

Let us consider the error of each of the predictors. Eq. 14.149 states that the prediction error can be calculated based on the previous error and current reflection coefficient:

$$J_{m+1} = J_m + \beta_{m+1} k_{m+1} = (1 - k_{m+1}^2) J_m. \quad (14.149)$$

where $J_0 = r_{xx}(0)$ and $\beta_1 = r_{xx}(1)$

The squared error should be 1 once the order of the predictor matches or exceeds the AR(q) model.

```
J0 = r_xx(1)
```

```
J0 = 59.7579
```

```
J1 = (1 - k(1)^2) * J0
```

```
J1 = 2.9078
```

```
J2 = (1 - k(2)^2) * J1
```

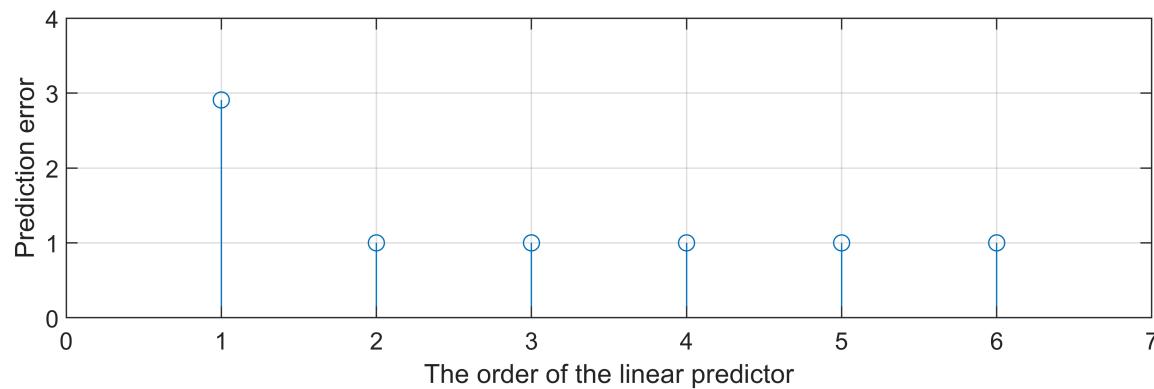
```
J2 = 1.0000
```

```
J3 = (1 - k(3)^2) * J2
```

```
J3 = 1.0000
```

Let us plot the order of the linear predictor versus the prediction error for an AR(2) model.

```
figure('position', [0, 0, 700, 200])
stem(err)
xlabel('The order of the linear predictor')
ylabel('Prediction error')
xlim([0, M+1])
ylim([0, 4])
grid on;
```



[?] ADSI Problem 6.7: Linear prediction of a sine signal

This problem addresses linear prediction on a simple harmonic signal where the results can be compared with our intuitive understanding.

Let a discrete time signal be given by

$$x(n) = \sqrt{2} \sin(\omega_0 n + \phi)$$

Where the phase ϕ is uniformly distributed between 0 and 2π .

```
clear variables;
```

[✓] 1) Determine the autocorrelation function for $x(n)$

The autocorrelation function of a real **sine signal** $z(n) = A \sin(\omega n + \phi)$ where A and ω are real constants and $\phi \sim U(0, 2\pi)$ is:

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

As $A = \sqrt{2}$, we have:

$$r_{xx}(\ell) = -\frac{(\sqrt{2})^2}{2} \cos(\omega_0\ell)$$

$$r_{xx}(\ell) = -\cos(\omega_0\ell)$$

[?] 2) Determine the 2nd order forward linear prediction filter

Write down the normal equation for the forward linear prediction filter and determine the filter coefficients for a 2nd order filter. For mathematical convinience we set $\omega_0 = \frac{\pi}{3}$.

The forward linear predictor can predict "future" sample $x(n)$ given the previous p samples $x(n-1), x(n-2), \dots, x(n-p)$. Such an predictor can be modelled as a FIR filter:

$$\hat{x}[n] = \sum_{k=1}^p h_k x[n-k] = \mathbf{h}^T \mathbf{x}[n-1], \quad (14.122)$$

where $\mathbf{h} \triangleq [h_1 \ h_2 \ \dots \ h_p]^T$ and $\mathbf{x}[n-1] \triangleq [x[n-1] \ x[n-2] \ \dots \ x[n-p]]^T$

The normal equation for the optimum p 'th order linear predictor is given by Eq. 14.125:

$$\mathbf{R}\mathbf{h} = \mathbf{r}, \quad (14.125)$$

The normal equation for the second order filter is:

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \end{bmatrix}$$

```
M = 2;
w0 = pi/3;
ell = 1:M+1;

r_xx = -cos(w0*ell);
R = toeplitz(r_xx(1:M));
r = r_xx(2:M+1)';
h_opt = R\r
```

```
h_opt = 2x1
1015 x
-2.2365
```

-2.2365

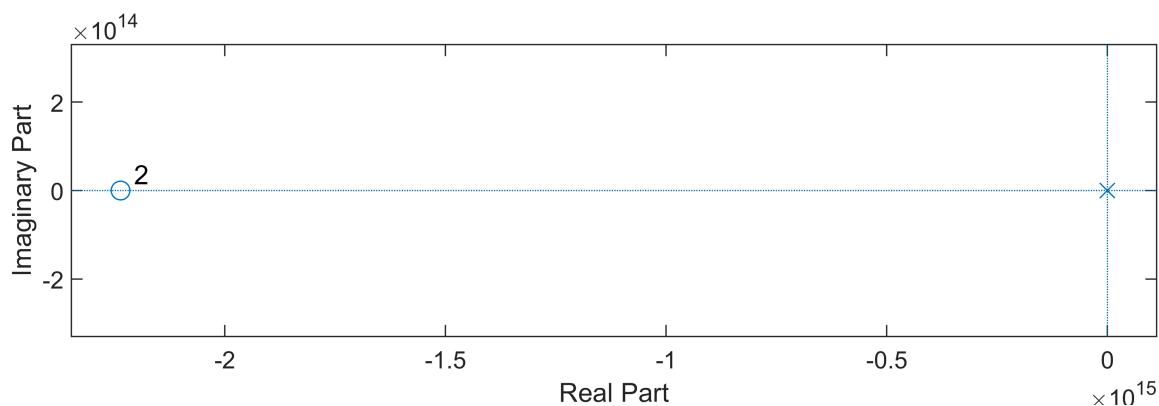
[?] The coefficients seem too large.

[»] 3) Find the system function for the filter and locate the zeros

Find the system function $H(z)$ for the filter and locate the zeros.

$$H(z) = h_1 z^{-1} + h_2 z^{-2}$$

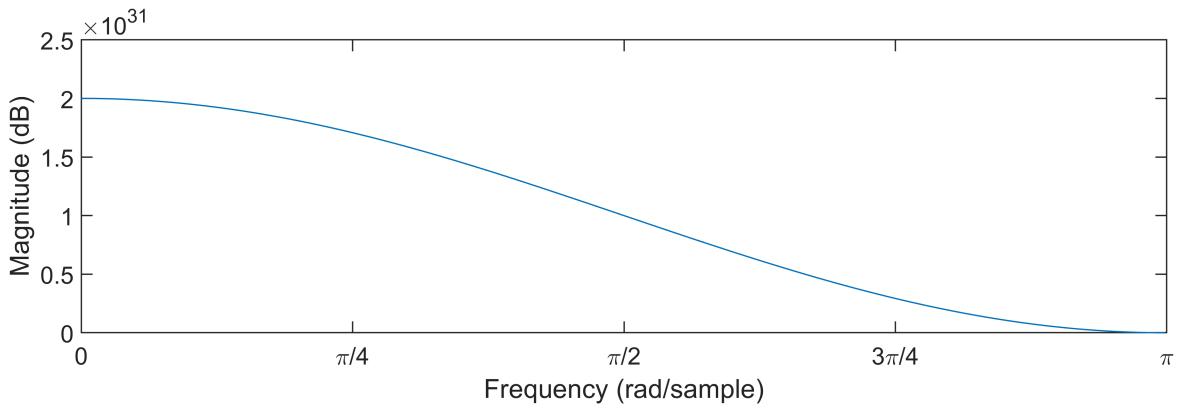
```
zplane(h_opt, 1)
```



[»] 4) Determine the frequency response, plot it and comment on the result

Determine the frequency response $H(\omega)$. Plot it and comment on the result.

```
[H, w] = freqz(h_opt, 1);
plot(w, H.*conj(H));
set(gca, 'XTick', 0:pi/4:pi)
set(gca, 'XTickLabel', {'0', '\pi/4', '\pi/2', '3\pi/4', '\pi'})
xlabel('Frequency (rad/sample)')
ylabel('Magnitude (dB)')
xlim([0, pi]);
```



[»] 5) Calculate the prediction error

The minimum square error for an optimum linear predictor is given by Eq. 14.127:

$$J_0 = r[0] - \mathbf{h}^T \mathbf{r} = r[0] - \mathbf{r}^T \mathbf{R}^{-1} \mathbf{r}. \quad (14.127)$$

```
r_xx(1)
```

```
ans = -0.5000
```

```
mse = r_xx(1) - h_opt'*r
```

```
mse = 3.3547e+15
```

[?] ADSI Problem 6.8: Autocorrelation function and linear prediction

Assume that for a given sequence of data $\{x(n)\}$ the autocorrelation function has been calculated and used to solve the normal equations so that the optimum p 'th order linear predictor was found. Now, an amplifier is placed in the signal chain so that the signal is $\{c \cdot x(n)\}$. How does the autocorrelation function and the linear predictor change?

In this problem, the autocorrelation function $r_{xx}(\ell)$ is already computed for signal $x(n)$.

We want to find the autocorrelation function for the amplified signal $y(n) = c \cdot x(n)$:

$$r_{yy}(\ell) = E[y(n)y(n-\ell)]$$

$$r_{yy}(\ell) = E[c x(n) c x(n-\ell)]$$

$$r_{yy}(\ell) = c^2 E[x(n)x(n-\ell)]$$

$$r_{yy}(\ell) = c^2 r_{xx}(\ell)$$

So, the autocorrelation sequence of the amplified signal is multiplied by c^2

The normal equation for the optimum linear predictor is given by Eq. 14.125:

$$\mathbf{R}\mathbf{h} = \mathbf{r}, \quad (14.125)$$

Even when the autocorrelation sequence is multiplied by some factor, the impulse response of the linear predictor \mathbf{h} will remain the same. Therefore, the linear predictor will still work the same way.

[?] **Are there other explanations?**

Matched Filters

Table of Contents

Matched Filters.....	2
Examples.....	4
White noise process.....	4
Problems.....	5
QUIZ: What is the assumption of the received signal?.....	5
QUIZ: How long should the matched filter be?.....	5
Quiz: Is it okay to choose a normalisation factor of 1.....	6
QUIZ: How can we interpret in the frequency domain?.....	6
True/False: Matched filters are only applicable when the noise is white.....	6
ADSI Problem 6.1: Matched filters.....	6
1) Plot a realisation of the signal with the added noise.....	7
2) Plot the power density spectrum of the noise and spectrum of the signal.....	8
3) Calculate the matched filter.....	11
[?] 4) Does the frequency response of the matched filter make sense?.....	13
[✓] 5) How efficient is the matched filter?.....	13
[✓] 6) What happens to the optimum SNR when the noise is twice as powerful?.....	14
[✓] 7) Compute SNR using a 6-tap moving average filter.....	15
Problem 14.14: Matched filters, compare short vs long signals.....	16
1) Explain why, in the absence of noise, the output is the ACRS of the desired signal.....	17
2) Determine the matched filter of a short cosine signal.....	17
3) Determine the matched filter of a long cosine signal.....	19
4) Which signal can be detected more easily by visual inspection?.....	21
Problem 14.35: Matched Filter and white noise.....	22
[❖] a) Compare the impulse responses with input signals.....	23
b) Compute the correlation of input and impulse response.....	24
c) Modify the signals so the relations are zero.....	24
Problem 14.37: Matched Filter and Square Pulse Signals.....	25
a) Process $x_1[n]$ through a matched filter.....	25
b) Process $x_2[n]$ through a matched filter.....	27
c) Discuss the results in (a) and (b)......	29
Exam 2017 Problem 3: Detect presence of signal using matched filter.....	30
1) Design a matched filter and determine the improvement in SNR.....	30
2) Discuss the improvement of SNR if a high-frequency signal is used.....	32
Exam 2018 Problem 3: Improve SNR with a matched filter.....	33
1) Design a matched filter to improve the SNR.....	33
2) Can SNR be improved by using a longer signal?.....	34
Exam 2018 Problem 3: Matched Filters.....	35
[✓] 1) Design a matched filter to improve the signal to noise ratio and comment on the improvement.....	35
[✓] 2) Can the signal to noise ratio be improved by using more than two blocks?.....	36
Exam 2017 Problem 3: Detect presence of signal using matched filter.....	37
1) Design a matched filter for detecting the presence of the signal and determine the improvement in signal to noise ratio.....	37
2) Discuss the improvement of SNR if another signal is used.....	38
Exam 2016 Problem 2: Detect the presence of signal via Matched Filter.....	39
1) Design a matched filter for detecting the presence of the signal and calculate the optimum signal to noise ratio.....	39
2) Will SNR improve if the longer signal is used?.....	41
3) Discuss whether the SNR will increase given a different ACRS?.....	41
Exam 2013, Problem 3, matched filter given ACRS for noise, SNR optimum vs non-optimum filter.....	43
1) Determine the matched filter for detecting the presence of absense of the signal.....	43
1a) Comment on the shape of the magnitude response.....	44
2) Calculate the optimum signal to noise ratio.....	45
3) Compute the SNR of a non-optimal filter.....	46

Matched Filters

Matched filters can be useful to determine whether a received signal is either the reflected signal with additive noise or just noise. This is useful in a radar system:

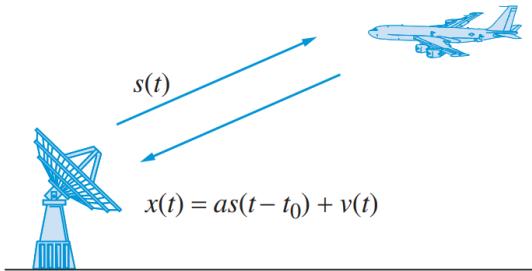


Figure 14.16 Principle of operation of a radar system.

where

- $s(t)$ is a deterministic signal of known form
- $x(t)$ is the signal measured by the radar if an object is present
- a is an attenuation factor
- t_0 is the round-trip delay
- $v(t)$ is random noise

The measured signal $x(n)$ by the radar can be two things:

- If an object happens to be in the way then part of the signal is reflected plus noise
- If there is no object in the way of the transmitted pulse, the received signal is just noise

To help us determine whether an object is present, we pass the received signal into a $p - 1$ -tap FIR filter:

$$y[n] = \sum_{k=0}^{p-1} h[k]x[n-k] \quad (14.89)$$

This figure shows what we are doing:

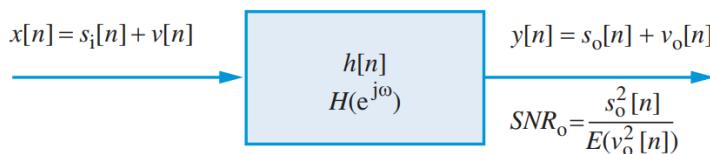


Figure 14.17 Input and output signals in a matched filter.

Our objective is to find the impulse $h[n]$ so that the output signal-to-noise ratio SNR_o is maximised:

$$\text{SNR}_o = \frac{\text{(Value of filtered signal at } n = n_0)^2}{\text{Power of filtered noise}} = \frac{s_o^2[n_0]}{\text{E}(v_o^2[n_0])}, \quad (14.90)$$

where $n_0 = p + D - 1$

at the decision time $n = n_0$. If we substitute the “signal present” case of (14.88) into (14.89) and set $n_0 = p + D - 1$, the output signal can be written as

$$y[n_0] = a\mathbf{h}^T s + \mathbf{h}^T v[n_0], \quad (14.91)$$

The solution is known as a *matched filter*. The impulse response of the optimum filter is

$$\mathbf{h} = \kappa \mathbf{R}_v^{-1} s. \quad (14.97)$$

where

\mathbf{R}_v is the autocorrelation matrix of the zero-mean wide-sense stationary noise $v[n]$

$$\mathbf{h} \triangleq \begin{bmatrix} h[0] \\ h[1] \\ \vdots \\ h[p-1] \end{bmatrix}, \quad s \triangleq \begin{bmatrix} s[p-1] \\ s[p-2] \\ \vdots \\ s[0] \end{bmatrix}$$

κ is a normalisation factor

Although the maximum SNR can be obtained by any choice of constant κ , we choose the constant by requiring that:

- (a) $\mathbf{h}^T s = 1$, which yields $\kappa = 1/s^T \mathbf{R}_v^{-1} s$
- (b) $\text{E}(v_o^2[n]) = \mathbf{h}^T \mathbf{R}_v \mathbf{h} = 1$, which yields $\kappa = 1/\sqrt{s^T \mathbf{R}_v^{-1} s}$.

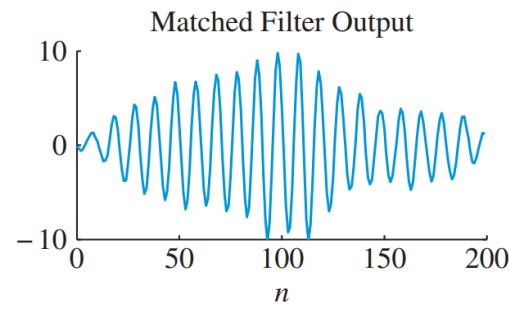
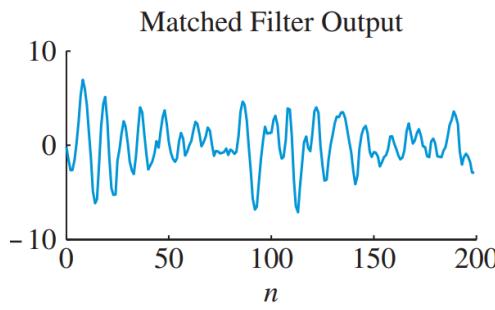
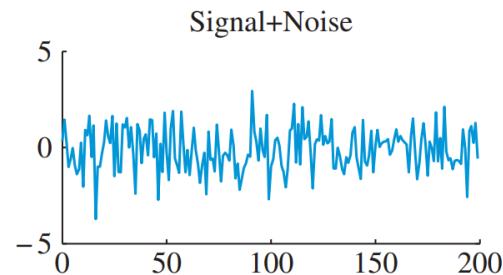
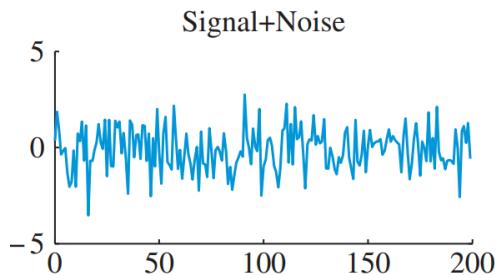
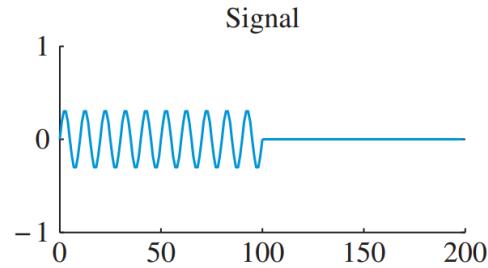
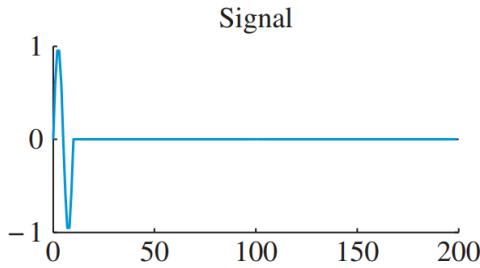
The maximum possible value of the output SNR is given by:

$$\text{SNR}_o = a^2 \tilde{s}^T \tilde{s} = a^2 s^T \mathbf{R}_v^{-1} s. \quad (14.98)$$

In summary, to design a matched filter we need two information:

- the autocorrelation sequence of the noise $r_{vv}[\ell]$
- the transmitted signal $s[n]$

Examples



White noise process

If the noise if white, then the autocorrelation matrix of white noise is $\mathbf{R}_v = \sigma_v^2 \mathbf{I}$. Then equations (14.97) and (14.98) are simplified to:

$$\mathbf{h}_w = \frac{\kappa}{\sigma_v^2} \mathbf{s}, \quad \text{SNR}_w = \frac{a^2}{\sigma_v^2} \sum_{k=0}^{p-1} s^2[k] \triangleq a^2 \frac{E_s}{\sigma_v^2}. \quad (14.99)$$

Problems

QUIZ: What is the assumption of the received signal?

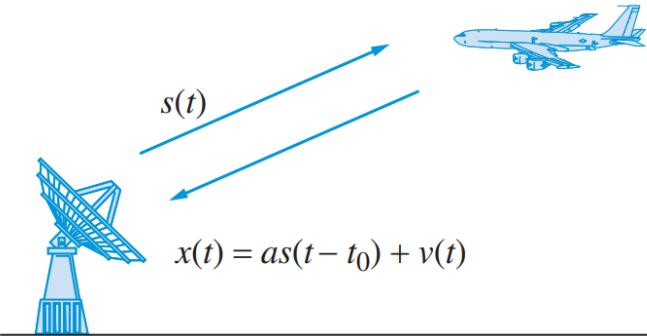


Figure 14.16 Principle of operation of a radar system.

The above model for $x(t)$ is sometimes a bit too simple. What assumptions have been made in the model?

1. Does not take the Doppler effect into account.
2. The signal could be reflected from different surfaces and not only by the plane
3. Water particles in the air would scatter the signal (radar clutter)

QUIZ: How long should the matched filter be?

In our quest for a optimum filter that will help us decide on whether the signal $s[n]$ is present or not the book decides on the filter

$$y[n] = \sum_{k=0}^{p-1} h[k]x[n-k] \quad (14.89)$$

Does the upper limit make sense?

or

What would change if we decrease or increase the number of taps in the filter?

We want to use a filter that is as long as the signal itself.

Quiz: Is it okay to choose a normalisation factor of 1

Is it okay to choose $\kappa = 1$? Yes, because κ is just a normalisation factor. We don't really care whether it is 1 or some other factor.

QUIZ: How can we interpret in the frequency domain?

We found the impulse response of the optimum filter

$$\mathbf{h}_{opt} = \kappa R^{-1} \mathbf{s}$$

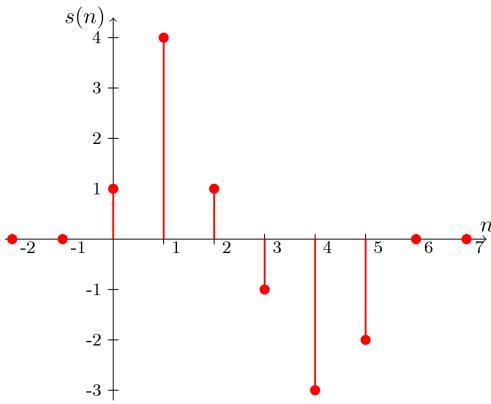
How do we interpret this in the frequency domain, i.e. what is $H_{opt}(e^{j\omega})$?

True/False: Matched filters are only applicable when the noise is white

The derivation of the matched filter makes no assumption of whether the noise is white or coloured. It should be noted though, that the interpretation of the filter is easier when the noise is white.

ADSI Problem 6.1: Matched filters

Consider a deterministic signal that is only non-zero in the vicinity of $n=0$ as shown below and zero for all other values of n .



The signal is disturbed by additive noise from an AR(3) process given by

$$v(n) = -0.5v(n-1) - 0.5v(n-2) - 0.25v(n-3) + w(n)$$

where $w(n) \sim WGN(0, 4)$.

The goal of this problem is to construct a matched filter that can help us distinguish between the presence or absence of the signal in the noise.

```
clear variables;
```

1) Plot a realisation of the signal with the added noise

Create a realization of the noise and add the signal somewhere in the noise. Plot the result and comment on whether the signal is detectable.

```
D = 50; % The place to embed signal into the noise

% Generate the deterministic signal s[n]
s = [1, 4, 1, -1, -3, -2]';

% Generate white noise w[n] with zero mean and variance 4
w_var = 4;
w = sqrt(w_var) * randn(10000, 1);

% Generate noise from an AR(3) process
b = 1;
a = [1, 0.5, 0.5, 0.25];
v = filter(b, a, w);

% Extract 100 samples to remove the transient effect
% at the beginning and at the end of the filtered signal
x = v(500:600);

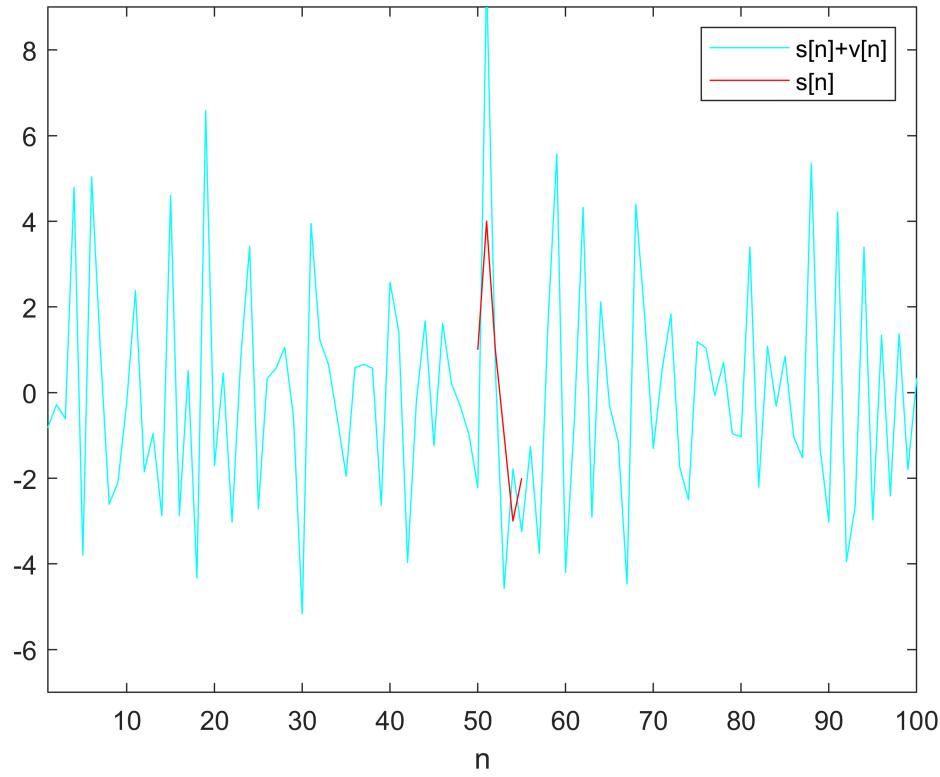
% Embed the signal s[n] in v[n] from D
s_in_x_start = D;
s_in_x_end = D + length(s) - 1;
x(s_in_x_start:s_in_x_end) = x(s_in_x_start:s_in_x_end) + s;

plot(1:length(x), x, 'c', ...
    s_in_x_start:s_in_x_end, s, 'r') % the signal is completely invisible
```

```

legend('s[n]+v[n]', 's[n]')
xlabel('n')
xlim([1,100])
ylim([-7, 9])

```



The signal is cannot be distinguished from the noise.

2) Plot the power density spectrum of the noise and spectrum of the signal

Plot the power density spectrum of the noise and spectrum of the signal, see Eq. (14.91).

The noise is generated by an AR(3) process plus some white Gaussian noise:

$$v(n) = -0.5v(n-1) - 0.5v(n-2) - 0.25v(n-3) + w(n)$$

where $w(n) \sim WGN(0, 4)$.

We can determine the Power Density Spectrum of an ARMA(p,q) process is given by

$$S_{yy}(\omega) = \sigma_x^2 |H(e^{j\omega})|^2 = \sigma_x^2 \left| \frac{\sum_{k=0}^q b_k e^{-j\omega k}}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2. \quad (13.133)$$

The power spectrum of an AR(p) process is given by:

$$S_{yy}(\omega) = \sigma_x^2 \left| \frac{1}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2$$

For this problem, we have:

$$S_{vv}(\omega) = 4 \left| \frac{1}{1 + 0.5e^{-j\omega} + 0.5e^{-j2\omega} + 0.25e^{-j3\omega}} \right|^2$$

The algorithm is as follows:

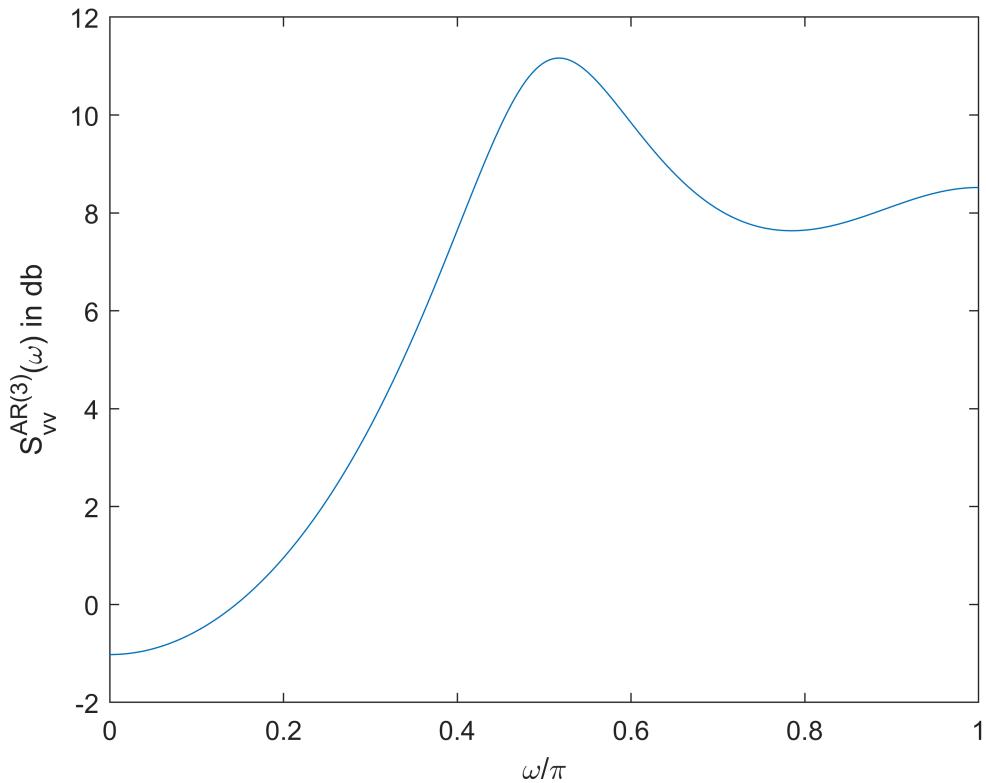
1. Use the coefficients $\{a_1, a_2, \dots, a_p\}$ for the AR(p) model,
2. Compute the transfer function for the AR(p) by computing the sum and finding its reciprocal
3. Compute the conjugate of the transfer function: $|H(e^{j\omega})|^2$
4. Multiply it with the variance σ_x^2

The algorithm is implemented in the functions ar2psd() function:

```
N = 256;

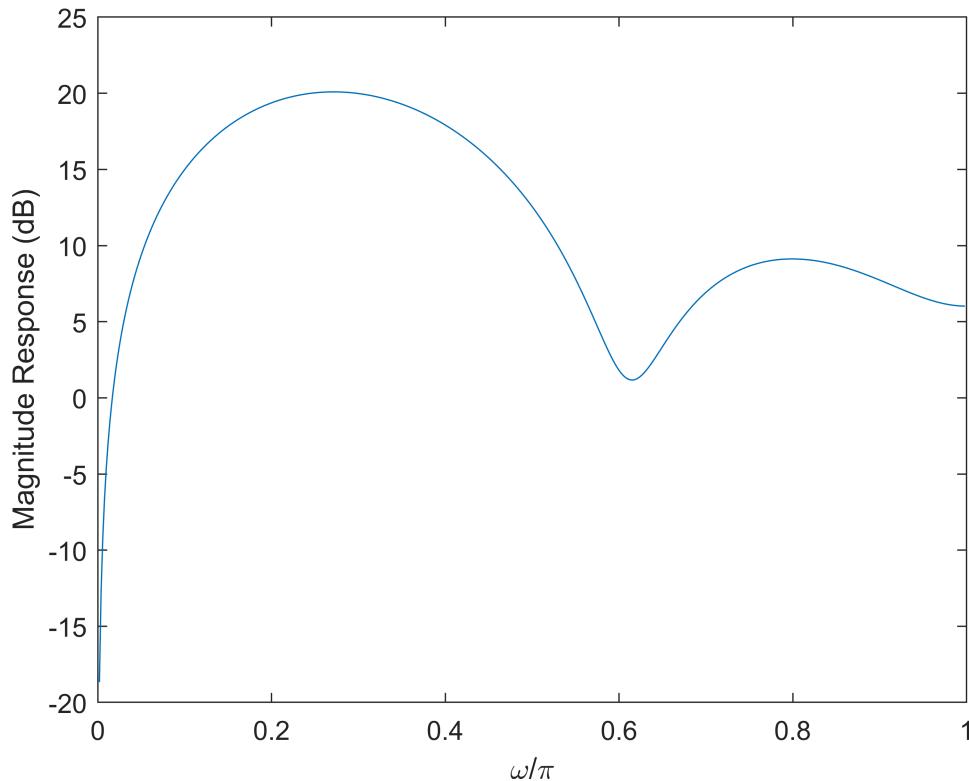
a = [0.5, 0.5, 0.25]; % The coefficients of the AR(3) model
w_var = 4; % The variance of white noise
[S_vv, w] = ar2psd(a, w_var, N); % Compute the PSD of AR(3) model

plot(w/pi, pow2db(S_vv))
xlabel('omega/pi')
ylabel('S_{vv}^{AR(3)}(omega) in db')
```



We can plot the spectrum of the signal $s(n)$ using the `freqz()` function:

```
[H,w2] = freqz(s, 1);
plot(w2/pi, 20*log10(abs(H)))
xlabel('omega/pi')
ylabel('Magnitude Response (dB)')
```



3) Calculate the matched filter

Calculate the matched filter, the frequency response of the matched filter and the optimum signal to noise ratio. You can use `xcorr` on the noise realization to find \mathbf{R}_v .

The impulse response of the matched filter is given by:

$$\mathbf{h} = \kappa \mathbf{R}_v^{-1} \mathbf{s}. \quad (14.97)$$

where κ is the normalisation factor. Although the maximum SNR can be obtained by any choice of constant κ , we choose the constant by requiring that:

- (a) $\mathbf{h}^T \mathbf{s} = 1$, which yields $\kappa = 1 / \mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}$
- (b) $E(v_o^2[n]) = \mathbf{h}^T \mathbf{R}_v \mathbf{h} = 1$, which yields $\kappa = 1 / \sqrt{\mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}}$.

```

p = numel(s); % Signal length

% The autocorrelation matrix must be p x p since
% its inverse is multiplied by a p-tap signal s(n)
[r_vv, ~] = xcorr(v, p-1, 'biased');
R_vv = toeplitz(r_vv(p:end));

% Compute normalisation factor

```

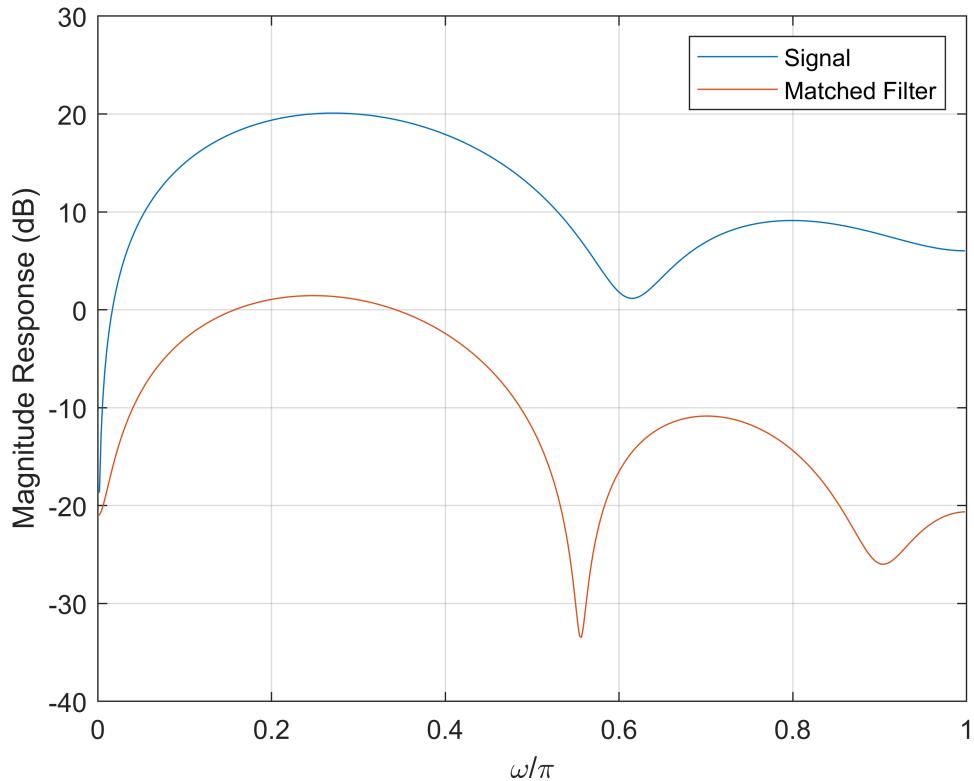
```

k = 1/sqrt(s'*(R_vv\s)); % Same as 1/sqrt(s'*inv(R_vv)*s)

% Compute the filter
h = k*(R_vv\s); % Same as k*inv(R_vv)*s

[H_s, w_s] = freqz(s, 1);
[H_h, w_h] = freqz(h, 1);
plot(w_s/pi, 20*log10(abs(H_s)), w_h/pi, 20*log10(abs(H_h)))
legend('Signal', 'Matched Filter')
xlabel('\omega/\pi')
ylabel('Magnitude Response (dB)')
grid on;

```



The maximum possible value of the output SNR is given by:

$$\text{SNR}_o = a^2 \tilde{s}^T \tilde{s} = a^2 s^T R_v^{-1} s. \quad (14.98)$$

Since the attenuation factor a is not given in this problem, we assume that it is 1:

```

a = 1;
SNR = a^2 * s' * (R_vv\s)

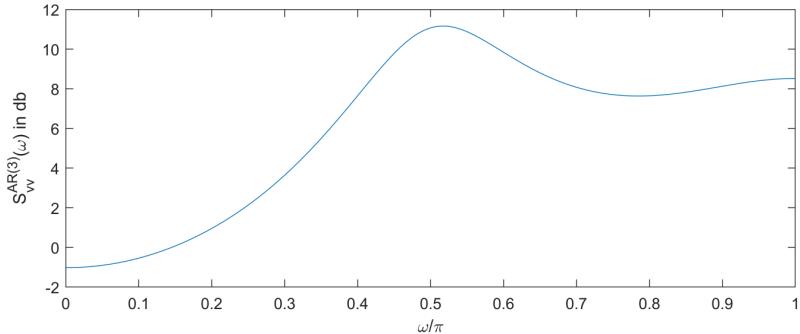
```

SNR = 11.3145

[?] 4) Does the frequency response of the matched filter make sense?

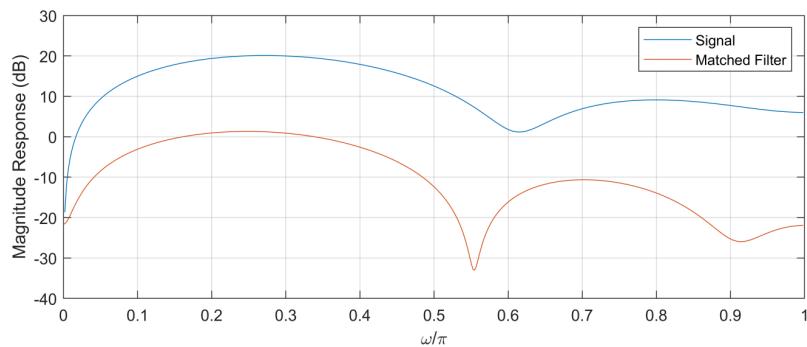
Does the frequency response of the matched filter make sense seen in relation with the spectra from question 2?

In question 2) we found that the AR(3) noise model has peak power at 0.5.



Looking at the frequency response of the signal, we observe that it peaks around 0.25 and dips around 20 dB at around 0.6.

The frequency response of the matched filter shows that it is minimising the noise in the region at around 0.55. The noise power peaks at around 0.5



[✓] 5) How efficient is the matched filter?

Plot the signal before and after the matched filter as well as the square of the output of the matched filter on the same graph and comment on the efficiency of the matched filter in our quest to detect the presence of the signal.

```
% x(n) is the 100 samples extracted from the noise signal v(n)
% y(n) is the result of feeding x(n) to the matched filter
y = filter(h, 1, x);
y_squared = y.^2;

n = 1:length(x);

n2 = s_in_x_start:s_in_x_end;
xs = s;

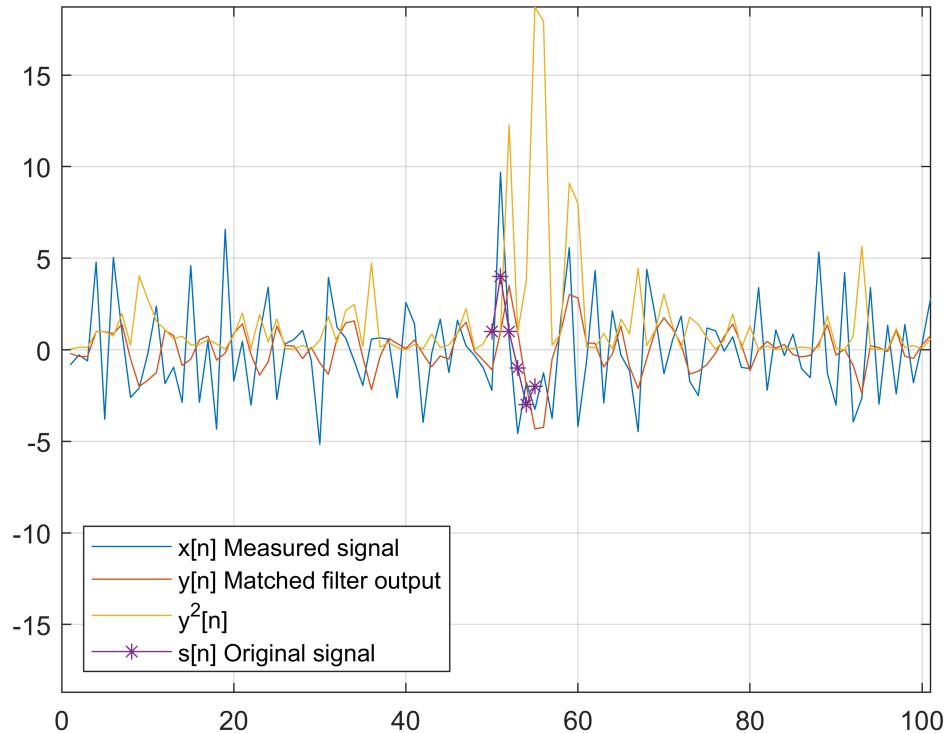
plot(n,x, n,y, n,y_squared, n2,xs, '*-')
xlim([0, numel(n)])
ylim([-max(y_squared), max(y_squared)])
```

```

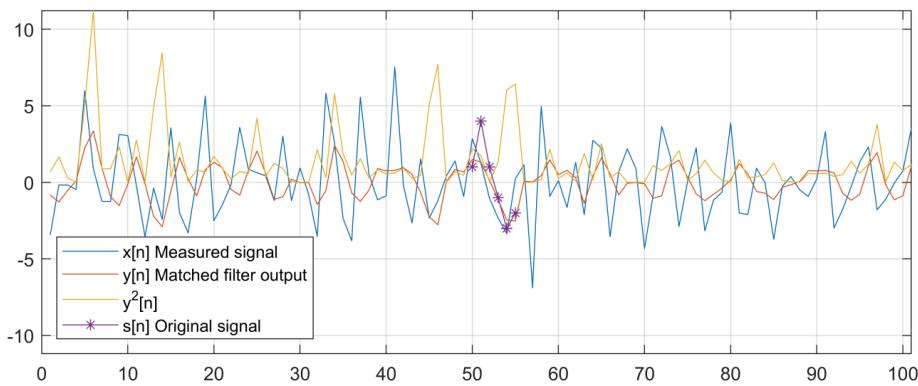
legend({'x[n] Measured signal', 'y[n] Matched filter output', 'y^2[n]', 's[n] Original signal'}, ...
    'Location','southwest')

grid on;

```



It's close to impossible to find the original signal $s[n]$ in the measured signal $x[n]$. It is also difficult to detect $s[n]$ after filtering $x[n]$ with the matched filter i.e., $y[n]$. It is easier to detect the original signal in the square of the output of the matched filter $y^2[n]$. Running the code multiple times i.e., creating different realisations of the noise, we may see many false positives. The figure below shows an example of this. The highest response is measured at $n = 6$ but we know that the signal is embedded in $n = 50$.



[✓] 6) What happens to the optimum SNR when the noise is twice as powerful?

Is the optimum signal to noise ratio halved if the AR(3) is instead driven by twice as powerful white noise, i.e. $w(n) \sim WGN(0,8)$? Why or why not?

We will compute the SNR again but this time with $\sigma_w^2 = 8$:

$$\text{SNR}_o = a^2 \tilde{s}^T \tilde{s} = a^2 s^T R_v^{-1} s. \quad (14.98)$$

```
% Generate white noise w[n] with zero mean and variance 8
w2_var = 8;
w2 = sqrt(w2_var) * randn(10000, 1);

% Put the white noise through an AR(3) process
b = 1;
a = [1, 0.5, 0.5, 0.25];
v2 = filter(b, a, w2);

% Extract 100 samples to remove the transient effect
% at the beginning and at the end of the filtered signal
x2 = v2(500:600);

p = numel(s); % Signal length

% The autocorrelation matrix must be MxM since
% its inverse is multiplied by a M-tap signal s(n)
[r_vv2, lags] = xcorr(x2, p-1, 'biased');
R_vv2 = toeplitz(r_vv2(p:end));

a = 1;
SNR2 = a^2 * s' * (R_vv2 \ s)
```

SNR2 = 6.0966

SNR2/SNR

ans = 0.5388

The optimal signal-to-noise-ratio is almost halved when the white noise is doubled. The reduction of SNR is not always decreased by a factor of though. Although the noise power is doubled, the shape of the optimum filter will change because it depends on the measured signal:

$$h = \kappa R_v^{-1} s. \quad (14.97)$$

[✓] 7) Compute SNR using a 6-tap moving average filter

Assume that instead of the matched filter a simple 6-tap moving average filter is used instead. That is

$$h = \left[\frac{1}{6} \ \frac{1}{6} \ \frac{1}{6} \ \frac{1}{6} \ \frac{1}{6} \ \frac{1}{6} \right]^T$$

Calculate the signal to noise ratio when the moving average filter is used.

We can compute the SNR using the following formula:

$$\text{SNR}_o = \frac{s_o^2[n_0]}{\text{E}(v_o^2[n_0])} = a^2 \frac{(\mathbf{h}^T \mathbf{s})^2}{\mathbf{h}^T \mathbf{R}_v \mathbf{h}}. \quad (14.94)$$

```
% h_avg = [1, 0, 1, 1, 1, 1]'./5;
h_avg = [1, 1, 1, 1, 1, 1]'./6;
a = 1;
SNR_avg = a^2 * (h_avg'*s)^2 / (h_avg'*R_vv*h_avg)

SNR_avg = 0
```

The SNR of the moving average filter is zero (or a number very close to zero). The moving average filter gives us the average of a signal. Since the original signal has zero mean, the average is zero. This is not surprising. If we change the filter, then the SNR is no longer zero.

```
h4 = [1, 0, 1, 1, 1, 1]'./5;
a = 1;
SNR4 = a^2 * (h4'*s)^2 / (h4'*R_vv*h4)
```

SNR4 = 1.4016

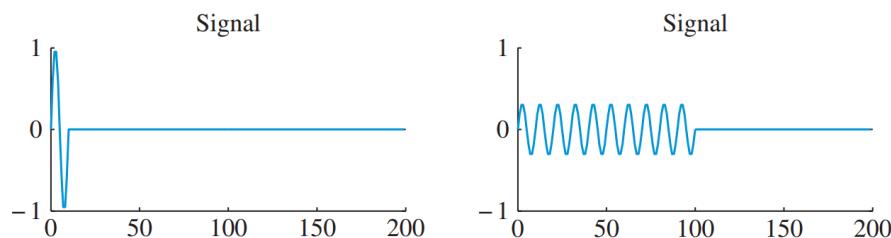
Problem 14.14: Matched filters, compare short vs long signals

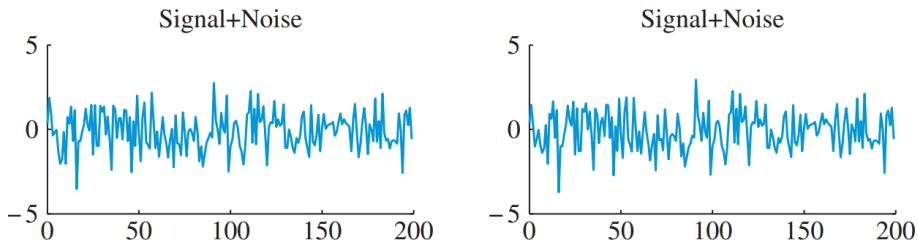
```
clear variables;
```

In this problem we discuss in detail the matched filtering problem illustrated in Figure 14.18.

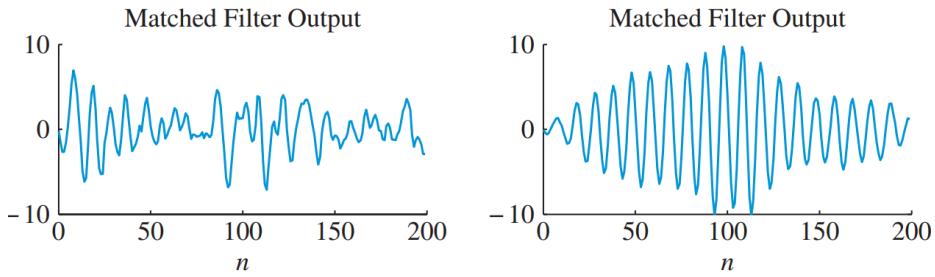
Figure 14.18 shows the operation of a matched filter in white noise.

The two signals have different lengths ($p = 10$ and $p = 100$, respectively) but the same energy.





Notice that the peak response of the matched filter occurs at $n_0 = p - 1$ because there is no time delay i.e. $D = 0$. Normally, the output SNR is maximised at time $n_0 = p + D - 1$. In practice, finding the time delay D depends on the particular application.



The input to the matched filter is given by

$$x[n] = s_I[n] + v[n] = a s[n - D] + v[n]$$

where

- a is the attenuation factor
- D is the round-trip delay
- $v[n]$ is the white Gaussian noise with zero mean and unit variance
- $s_I[n] = 0$ outside the interval $0 \leq n \leq p - 1$

1) Explain why, in the absence of noise, the output is the ACRS of the desired signal.

In case of white noise, the impulse response of the matched filter $h[n]$ is equal to the signal that we want to detect $s_I[n]$. As such the output of the matched filter is the same as autocorrelation:

2) Determine the matched filter of a short cosine signal

Suppose that $p = 10$ and $s_I[n] = \cos\left(2\pi \frac{n}{10}\right)$. Generate $N = 200$ samples of the noisy signal $s[n]$ and process it through the matched filter designed for $s_I[n]$. Plot the desired, input, and filtered signals and determine when the matched filter output is output.

The impulse response of the matched filter is given by:

$$\mathbf{h} = \kappa \mathbf{R}_v^{-1} \mathbf{s}. \quad (14.97)$$

where κ is the normalisation factor and \mathbf{R}_v is the autocorrelation matrix of the noise.

The output SNR can be computed using the formula:

$$\text{SNR}_o = a^2 \tilde{s}^T \tilde{s} = a^2 s^T R_v^{-1} s. \quad (14.98)$$

The computation is coded in a MATLAB function named `matched_filter` (see the end of the document)

```
N = 200;
n = (1:N)';
p = 10; % Signal length

% Generate the attenuated signal.
% Ensure that s[n] is zero when n is outside the interval [1, p]
s = zeros(N, 1);
inside_indices = (n >= 1 & n <= p);
s(inside_indices) = cos(2*pi*n(inside_indices)/10);

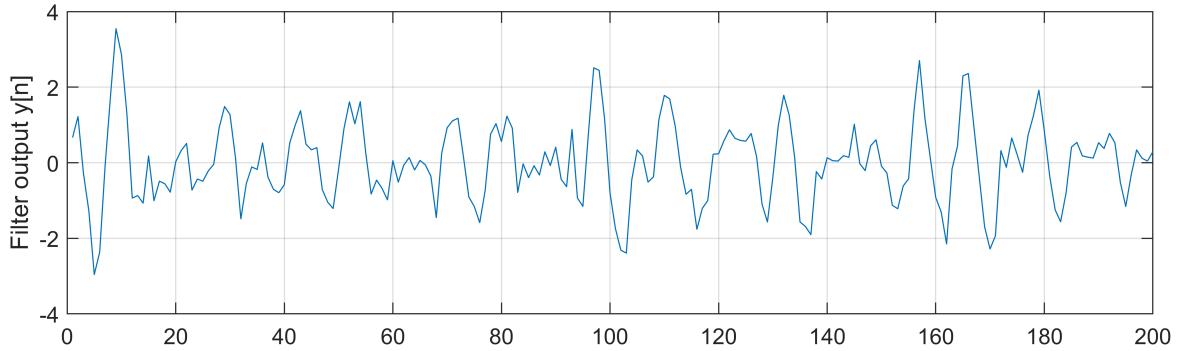
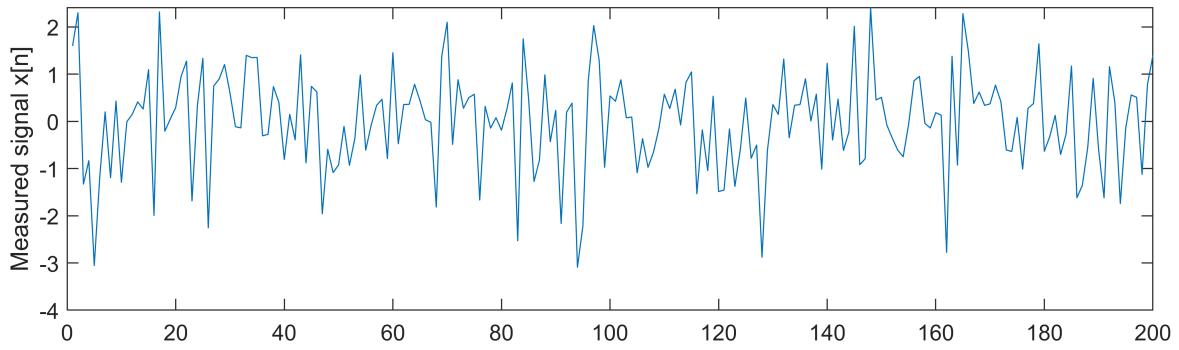
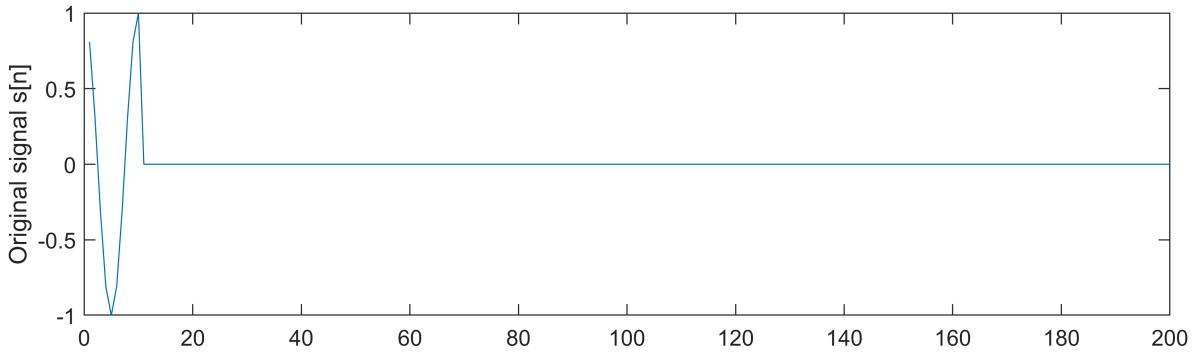
% Generate the zero-mean WGN with unit variance
v = randn(N, 1);

% Generate the measured signal x[n]
x = v + s;

% Compute the impulse response of the matched filter
[h, SNR] = matched_filter(s(1:p), v);

% Feed the measured signal to the matched filter
y = filter(h, 1, x);

figure('position', [0, 0, 800, 800])
subplot(3,1,1); plot(n,s); ylabel('Original signal s[n]')
subplot(3,1,2); plot(n,x); ylabel('Measured signal x[n]')
subplot(3,1,3); plot(n,y); ylabel('Filter output y[n]')
xlim([0 N])
grid on;
```



```
clear figure;
```

From the filtered output, it is difficult to determine where the original signal is placed.

3) Determine the matched filter of a long cosine signal

Repeat 2) for $p = 100$ and $s_i[n] = \frac{1}{\sqrt{10}} \cos\left(2\pi \frac{n}{10}\right)$.

```

p = 100; % Signal length

% Generate the attenuated signal.
% Ensure that s[n] is zero when n is outside the interval [1, p]
s = zeros(N, 1);
inside_indices = (n >= 1 & n <= p);
s(inside_indices) = (1/sqrt(10)) * cos(2*pi*n(inside_indices)/10);

% Generate the zero-mean WGN with unit variance
v = randn(N, 1);

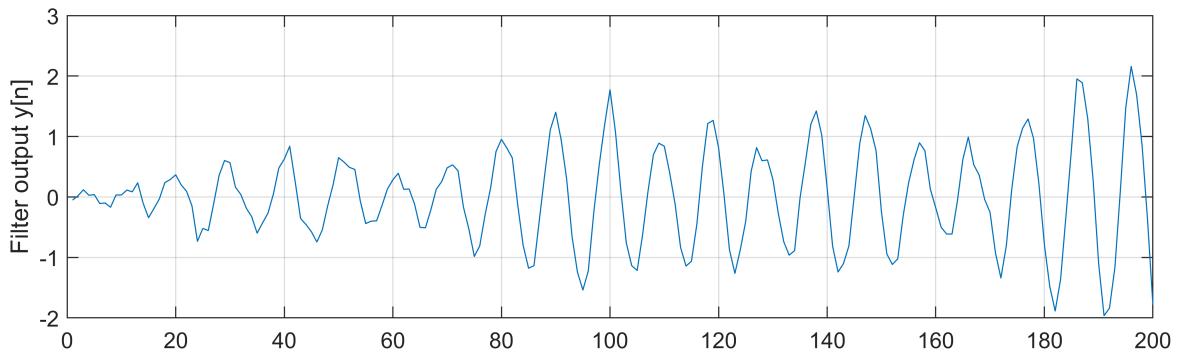
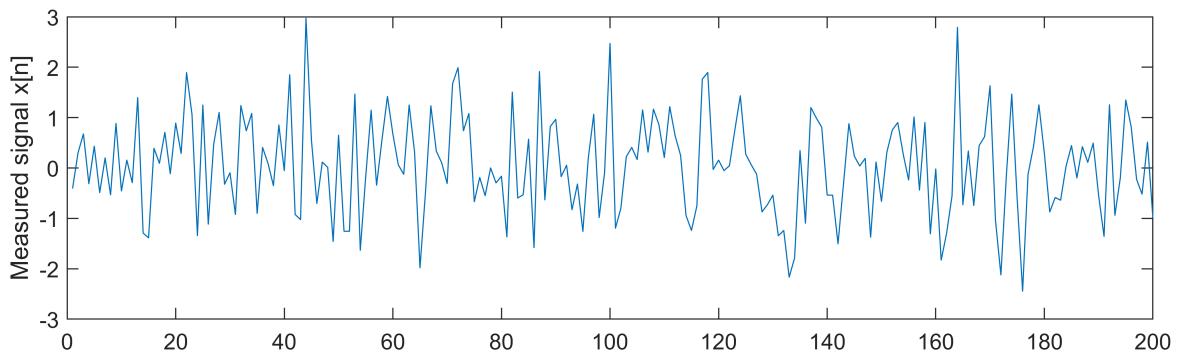
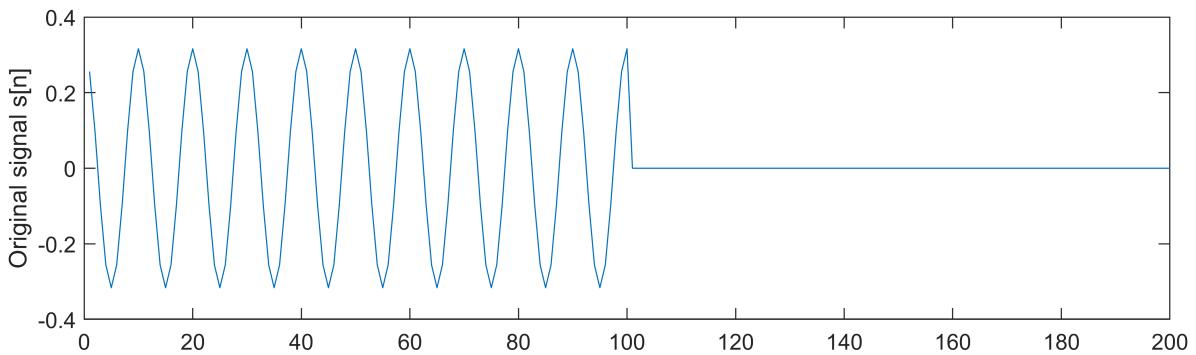
% Generate the measured signal x[n]
x = v + s;

% Compute the impulse response of the matched filter
[h_100, SNR_100] = matched_filter(s(1:p), v);

% Feed the measured signal to the matched filter
y = filter(h_100, 1, x);

figure('position', [0, 0, 800, 800])
subplot(3,1,1); plot(n,s); ylabel('Original signal s[n]')
subplot(3,1,2); plot(n,x); ylabel('Measured signal x[n]')
subplot(3,1,3); plot(n,y); ylabel('Filter output y[n]')
xlim([0 N])
grid on;

```



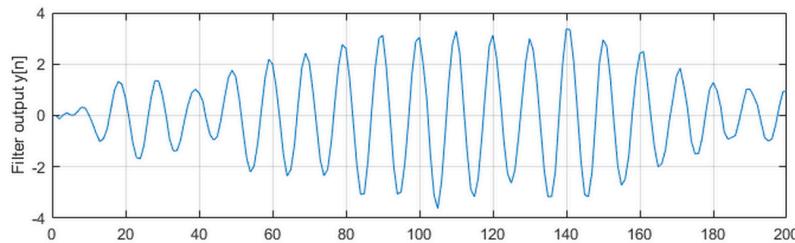
```
clear figure;
```

4) Which signal can be detected more easily by visual inspection?

Which signal can be detected more easily by visual inspection of the matched filter output? Is this justified by comparing the output SNR in each case?

For certain noise realisations, it is easier to detect the second (longer) signal by visual inspect than the first signal. However, running the code multiple times i.e., creating different realisations of the noise, we may see many false

positives. The figure below shows an example of this. The highest response is measured at $n = 140$ but the maximum filter response should occur at $n = p + D - 1 = 100 + 0 - 1 = 99$.



Comparing the output SNR of the two filters, we observe that the SNR of the longer filter is usually better but this is not always the case. Different realisation of the noise process, yields different results:

SNR

SNR = 5.8908

SNR_100

SNR_100 = 4.4832

Problem 14.35: Matched Filter and white noise

In the book on page 861, the authors claim: "*The term matched filter was introduced because, in the case of white noise, the impulse response is “matched” to the shape of the signal being sought. In fact, the output of the matched filter is proportional to the correlation between the signal segment stored in the filter memory and the signal of interest.*"

This problem investigates this claim.

Consider the two 10-point signals $x_0[n]$ and $x_1[n]$ given below:

$$x_0[n] = \{1, 1, 1, -1, -1, 0, 0, 0, 0, 0\},$$

$$x_1[n] = \{1, 1, 1, 1, -1, 0, 0, 0, 0, 0\}.$$

These signals are sent over a communication channel which adds white noise to the signals. Using a correlation-detector approach, we want to detect signals in white noise. Let $h_0[n]$ denote the matched filter for $x_0[n]$ and let $h_1[n]$ denote the matched filter for $x_1[n]$.

```
clear variables;
```

[◆] a) Compare the impulse responses with input signals

Determine and plot the responses of $h_0[n]$ to $x_0[n]$ and $x_1[n]$. Repeat the same for $h_1[n]$. Compare these outputs at $n = 10$.

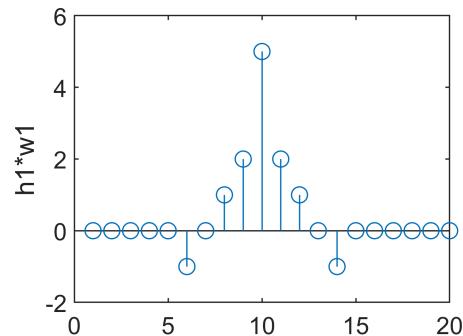
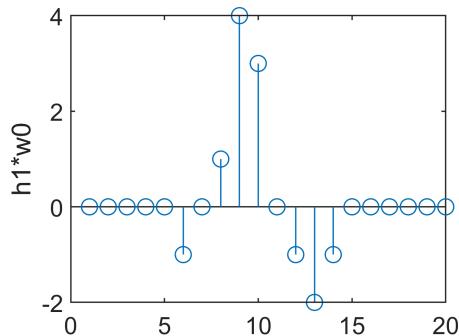
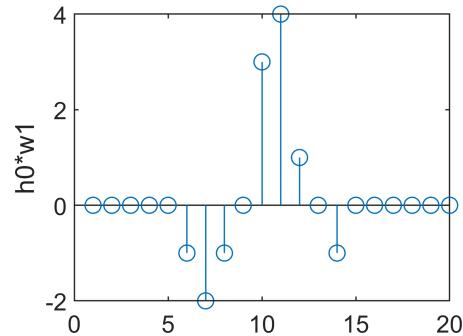
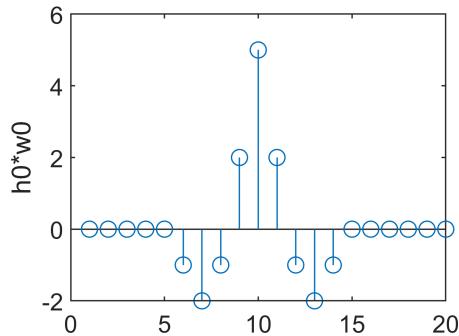
I don't understand what should be done here!

```

x0 = [1, 1, 1, -1, -1, 0, 0, 0, 0, 0]';
x1 = [1, 1, 1, 1, -1, 0, 0, 0, 0, 0]';

N = 20; n = 1:N;
v=zeros(N,1);%randn(N,1);
w0 = v + [x0;zeros(N-length(x0),1)];
w1 = v + [x1;zeros(N-length(x1),1)];
h0 = flip(x0); %wgn -> just use flipped
h1 = flip(x1);
figure
subplot(2,2,1); stem(filter(h0,1,w0)); ylabel('h0*w0')
subplot(2,2,2); stem(filter(h0,1,w1)); ylabel('h0*w1')
subplot(2,2,3); stem(filter(h1,1,w0)); ylabel('h1*w0')
subplot(2,2,4); stem(filter(h1,1,w1)); ylabel('h1*w1')

```



b) Compute the correlation of input and impulse response

You should notice that the output of the above matched filters at $n = 10$ can be computed as a correlation of the input and the impulse response. Implement such a structure and determine its output for each case in part (a) above.

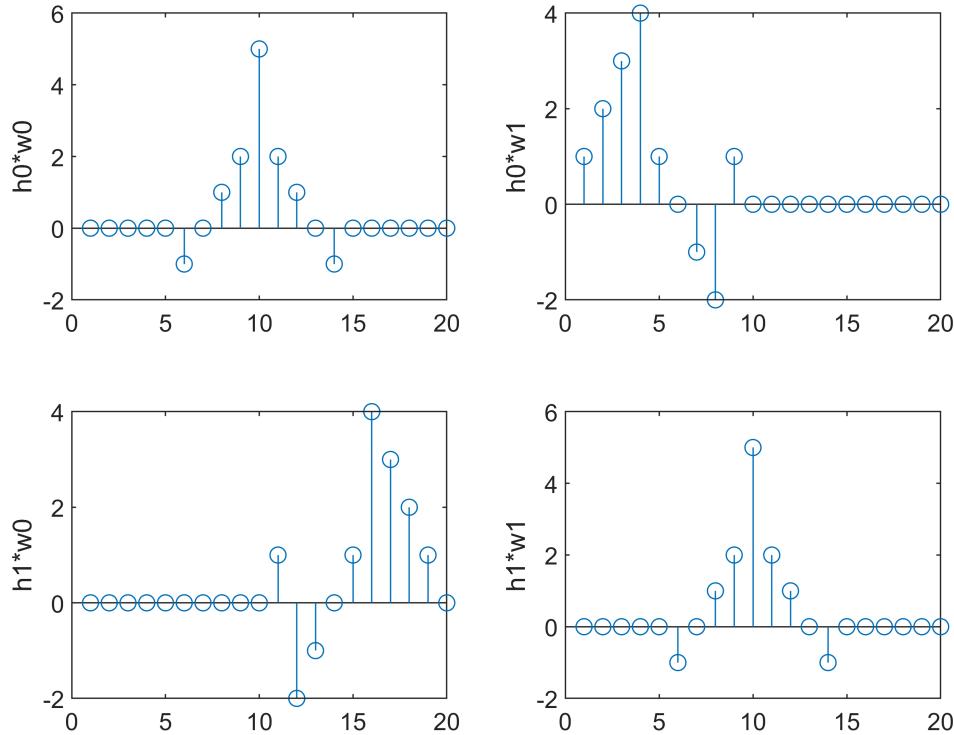
c) Modify the signals so the relations are zero

How would you modify the signal $x_0[n]$ so that the outputs of $h_0[n]$ to $x_1[n]$ and $h_1[n]$ to $x_0[n]$ are zero?

```
% Don't understand what is going on here
x0 = flip(x1); % the inverse, maybe?
corr(x0,x1)
```

```
ans = -0.2195
```

```
w0 = v + [x0;zeros(N-length(x0),1)];
h0 = flip(x0); %wgn -> just use flipped
figure
subplot(2,2,1); stem(filter(h0,1,w0)); ylabel('h0*w0')
subplot(2,2,2); stem(filter(h0,1,w1)); ylabel('h0*w1')
subplot(2,2,3); stem(filter(h1,1,w0)); ylabel('h1*w0')
subplot(2,2,4); stem(filter(h1,1,w1)); ylabel('h1*w1')
```



Problem 14.37: Matched Filter and Square Pulse Signals

Let $s_1[n]$ and $s_2[n]$ be short-length pulses of unit energy as given below:

$$s_1[n] = 1/3, \quad 0 \leq n \leq 8 \quad \text{and} \quad s_2[n] = 0.1, \quad 0 \leq n \leq 99$$

and zero everywhere. These pulses are observed in noise, i.e.

$$x_i[n] = s_i[n] + v[n], \quad i = 1, 2$$

where $v[n] \sim \text{WGN}(0, 1)$.

```
clear variables;
```

a) Process $x1[n]$ through a matched filter

Generate 200 samples of the noisy $x1[n]$ signal and process it through the matched filter designed for $s1[n]$. Plot the original, noisy, and filtered signals and determine when the matched filter output is maximum.

```

s1 = 1/3*ones(9, 1);

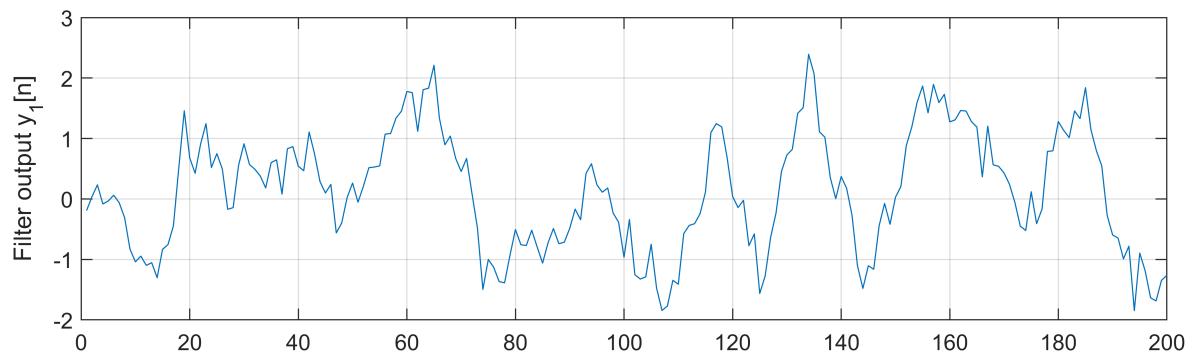
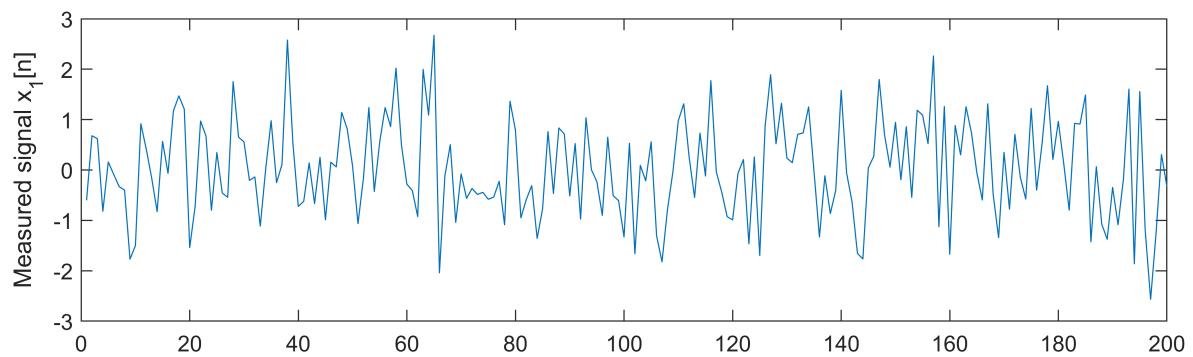
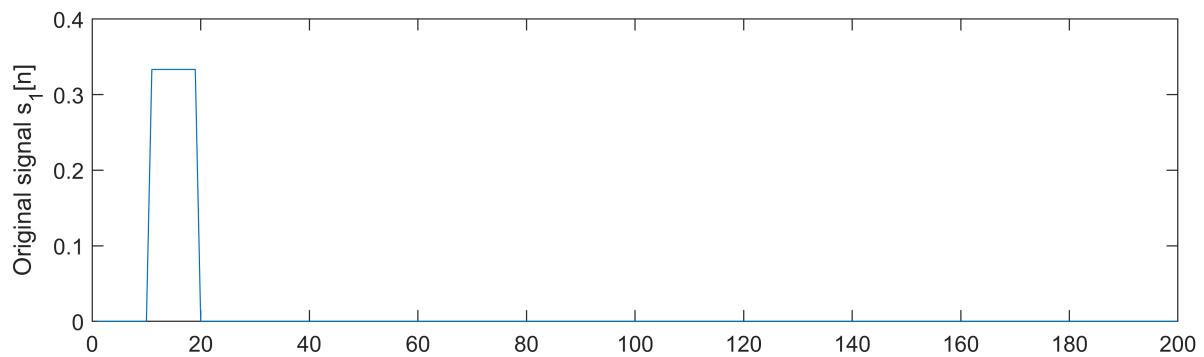
N = 200;
n = 1:N;
noise_var = 1;
D = 10;
[x1, v1, s1e] = gen_noisy_signal(s1, N, noise_var, D);

% Compute the impulse response of the matched filter
[h1, SNR1] = matched_filter(s1, v1);

% Feed the measured signal to the matched filter
y1 = filter(h1, 1, x1);

figure('position', [0, 0, 800, 800])
subplot(3,1,1); plot(n, s1e); ylabel('Original signal s_1[n]')
subplot(3,1,2); plot(n, x1); ylabel('Measured signal x_1[n]')
subplot(3,1,3); plot(n, y1); ylabel('Filter output y_1[n]')
xlim([0 N])
grid on;

```



b) Process $x_2[n]$ through a matched filter

Generate 200 samples of the noisy $x_2[n]$ signal and process it through the matched filter designed for $s_2[n]$. Plot the original, noisy, and filtered signals and determine when the matched filter output is maximum.

Let $s_1[n]$ and $s_2[n]$ be short-length pulses of unit energy as given below:

$$s_1[n] = 1/3, \quad 0 \leq n \leq 8 \quad \text{and} \quad s_2[n] = 0.1, \quad 0 \leq n \leq 99$$

and zero everywhere. These pulses are observed in noise, i.e.

$$x_i[n] = s_i[n] + v[n], \quad i = 1, 2$$

where $v[n] \sim \text{WGN}(0, 1)$.

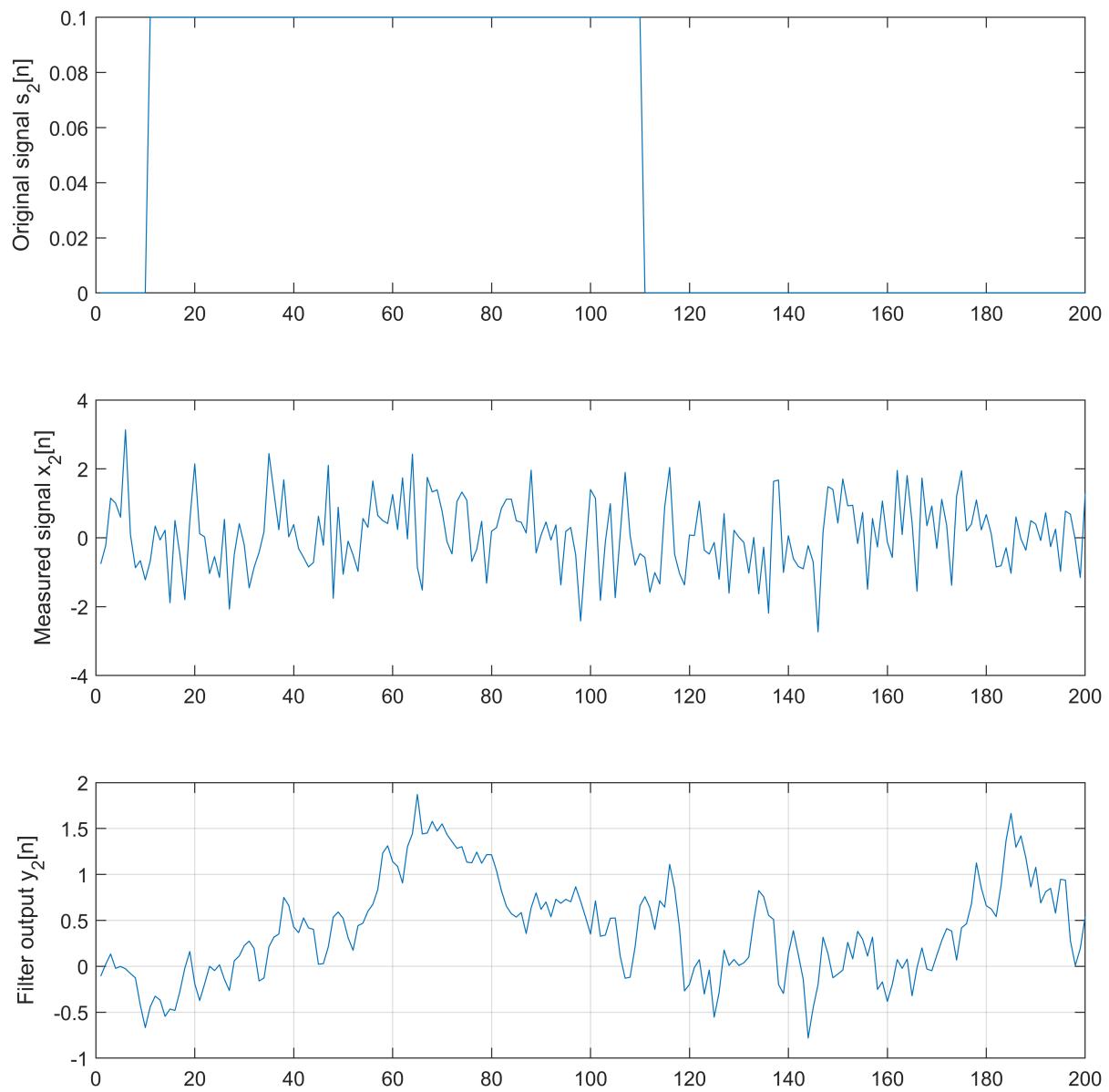
```
s2 = 0.1*ones(100, 1);

N = 200;
n = 1:N;
noise_var = 1;
D = 10;
[x2, v2, s2e] = gen_noisy_signal(s2, N, noise_var, D);

% Compute the impulse response of the matched filter
[h2, SNR2] = matched_filter(s2, v2);

% Feed the measured signal to the matched filter
y2 = filter(h2, 1, x1);

figure('position', [0, 0, 800, 800])
subplot(3,1,1); plot(n, s2e); ylabel('Original signal s_2[n]')
subplot(3,1,2); plot(n, x2); ylabel('Measured signal x_2[n]')
subplot(3,1,3); plot(n, y2); ylabel('Filter output y_2[n]')
xlim([0 N])
grid on;
```



c) Discuss the results in (a) and (b)

Discuss your results in (a) and (b) above in terms of filtered waveforms and maximization of the output SNR.

The peak response of the matched filter should occur at $n_0 = p + D - 1$

Since $s_1[n]$ is a 10-tap signal, this point should be $n_0 = 10 + 10 - 1 = 19$

Since $s_2[n]$ is a 100-tap signal, this point should be $n_0 = 100 + 10 - 1 = 119$

We rarely see these points for different realisations of the noise process.

```
[~, peak_n1] = max(y1)
```

```
peak_n1 = 134
```

```
[~, peak_n2] = max(y2)
```

```
peak_n2 = 65
```

What can we conclude?

```
SNR1
```

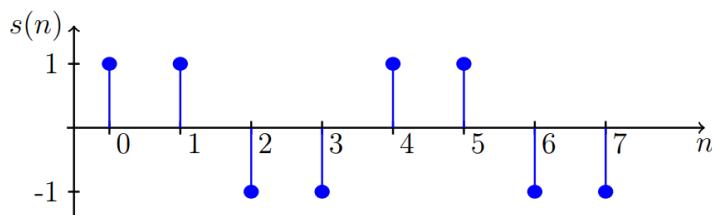
```
SNR1 = 0.9027
```

```
SNR2
```

```
SNR2 = 2.1919
```

Exam 2017 Problem 3: Detect presence of signal using matched filter

Consider the deterministic signal, $s(n)$ shown below in blue. The signal is zero for all other values of n .



The signal is distorted by additive low frequency noise with autocorrelation $r_v(\ell) = 0.4^{|\ell|}$.

```
clear variables;
```

1) Design a matched filter and determine the improvement in SNR

Design a matched filter for detecting the presence of the signal and determine the improvement in signal to noise ratio.

The impulse response of the matched filter is given by:

$$\mathbf{h} = \kappa \mathbf{R}_v^{-1} \mathbf{s}. \quad (14.97)$$

where \mathbf{R}_v is autocorrelation matrix of noise and κ is the normalisation factor.

Although the maximum SNR can be obtained by any choice of constant κ , we choose the constant by requiring that:

- (a) $\mathbf{h}^T \mathbf{s} = 1$, which yields $\kappa = 1/\mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}$
- (b) $E(v_o^2[n]) = \mathbf{h}^T \mathbf{R}_v \mathbf{h} = 1$, which yields $\kappa = 1/\sqrt{\mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}}$.

```
s = [1, 1, -1, -1, 1, 1, -1, -1]';

p = numel(s); % Signal length

% The autocorrelation matrix must be p x p since
% its inverse is multiplied by a p-tap signal s(n)
ell = 0:p-1;
r_vv = 0.4.^ell;
R_vv = toeplitz(r_vv);

% Compute normalisation factor (b)
k = 1/(s'*(R_vv\s)); % Using (a)
% k = 1/sqrt(s'*(R_vv\s)); % Using (b)

% Compute the filter
h = k*(R_vv\s); % Same as k*inv(R_vv)*s

% Print the matched filter coefficients
h
```

```
h = 8x1
0.0735
0.1422
-0.1422
-0.1422
0.1422
0.1422
-0.1422
-0.0735
```

The SNR at the input is given by:

$$\text{SNR}_i = \frac{\text{power of signal}}{\text{power of noise}} = \frac{s^2(n = n_0)}{r_v(0)}$$

```
SNR_i = s(1)^2/r_vv(1)
```

```
SNR_i = 1
```

The optimum SNR at the output is given by:

$$\text{SNR}_o = a^2 \tilde{s}^T \tilde{s} = a^2 s^T R_v^{-1} s. \quad (14.98)$$

Assuming the attenuation factor $a = 1$:

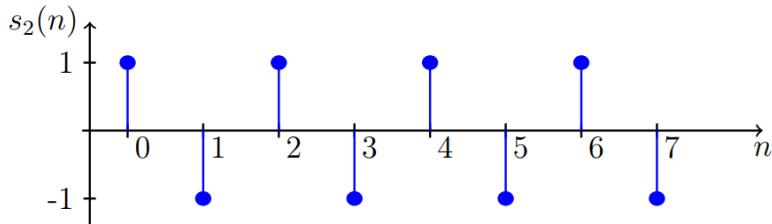
```
a = 1;
SNR_o = a^2 * s' * (R_vv\s)
```

```
SNR_o = 9.7143
```

The improvement in SNR is 9.7 (almost 10 times)

2) Discuss the improvement of SNR if a high-frequency signal is used

A second signal $s_2(n)$ is given by



2. Discuss whether the signal to noise ratio will be improved if the signal $s_2(n)$ is used instead of $s(n)$.

If we use $s_2(n)$ instead of $s(n)$, the output SNR becomes 17.3:

```
s = [1, -1, 1, -1, 1, -1, 1, -1]';
p = numel(s); % Signal length
ell = 0:p-1;
r_vv = 0.4.^ell;
R_vv = toeplitz(r_vv);
a = 1;
SNR_o = a^2 * s' * (R_vv\s)
```

```
SNR_o = 17.3333
```

The $s_2(n)$ is similar to $s(n)$ but it oscillates faster i.e., has higher frequency than the $s(n)$ signal. Since the noise is at low frequency, it is easier to separate the $s_2(n)$ signal from the noise. This means that a higher SNR can be expected from a signal with a higher frequency.

That is why the output SNR for $s_2(n)$ is better than that for $s(n)$.

Exam 2018 Problem 3: Improve SNR with a matched filter

A deterministic signal is given by

n	$s(n)$
0	1
1	-1
2	1
3	-1

The signal is distorted by additive low frequency noise with autocorrelation $r_v(l) = 0.8^{|l|}$.

```
clear variables;
```

1) Design a matched filter to improve the SNR

Design a matched filter to improve the signal to noise ratio and comment on the improvement.

The impulse response of the matched filter is given by:

$$\mathbf{h} = \kappa \mathbf{R}_v^{-1} \mathbf{s}. \quad (14.97)$$

where \mathbf{R}_v is autocorrelation matrix of noise and κ is the normalisation factor.

Although the maximum SNR can be obtained by any choice of constant κ , we choose the constant by requiring that:

- (a) $\mathbf{h}^T \mathbf{s} = 1$, which yields $\kappa = 1 / \mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}$
- (b) $E(v_o^2[n]) = \mathbf{h}^T \mathbf{R}_v \mathbf{h} = 1$, which yields $\kappa = 1 / \sqrt{\mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}}$.

```
s = flip([1, -1, 1, -1])';
p = numel(s); % Signal length

% The autocorrelation matrix must be p x p since
% its inverse is multiplied by a p-tap signal s(n)
ell = 0:p-1;
r_vv = 0.8.^ell;
R_vv = toeplitz(r_vv);

% Compute normalisation factor (b)
k = 1/ (s'*(R_vv\s));

% Compute the filter
h = k*(R_vv\s);

% Print the matched filter coefficients
h
```

```

h = 4x1
-0.1786
0.3214
-0.3214
0.1786

```

The instantaneous power of the signal (when present) is $s^2(n) = 1$, the average noise power is found from $r_v(0) = 1$ and the input SNR is therefore 1.

The SNR at the input is given by:

$$\text{SNR}_i = \frac{\text{power of signal}}{\text{power of noise}} = \frac{s^2(n = n_0)}{r_v(0)}$$

```
SNR_i = s(1)^2/r_vv(1)
```

```
SNR_i = 1
```

The optimum SNR is given by:

$$\text{SNR}_o = a^2 \tilde{s}^T \tilde{s} = a^2 s^T R_v^{-1} s. \quad (14.98)$$

Assuming the attenuation factor $a = 1$:

```
a = 1;
SNR = a^2 * s' * (R_vv\s)
```

```
SNR = 28.0000
```

The SNR is improved from 1 to 28.

The improvement in SNR from 1 to 28 might appear large but is caused by the fact that the noise is mainly at low frequency and the signal is high frequencies. Signal and noise is therefore rather easily separated.

2) Can SNR be improved by using a longer signal?

2. The $s(n)$ signal consists of two blocks each containing 1 and -1. Can the signal to noise ratio be improved by using more than two blocks?

It is indeed possible to improve SNR further by using more blocks in the signal. A simple extension of the above calculation to a 3-block signal gives an optimum SNR of 46.

Yes, the SNR can be improved. With three blocks, the SNR is increased to 46:

```

s = [1, -1, 1, -1, 1, -1]';
p = numel(s);
ell = 0:p-1;
r_vv = 0.8.^ell;
R_vv = toeplitz(r_vv);
k = 1/sqrt(s'*(R_vv\s));

```

```

h = k*(R_vv\s);
a = 1;
SNR = a^2 * s' * (R_vv\s)

```

SNR = 46.0000

Exam 2018 Problem 3: Matched Filters

A deterministic signal is given by

n	$s(n)$
0	1
1	-1
2	1
3	-1

The signal is distorted by additive low frequency noise with autocorrelation $r_v(l) = 0.8^{|l|}$.

```
clear variables;
```

[✓] 1) Design a matched filter to improve the signal to noise ratio and comment on the improvement.

The impulse response of the matched filter is given by:

$$\mathbf{h} = \kappa \mathbf{R}_v^{-1} \mathbf{s}. \quad (14.97)$$

where \mathbf{R}_v is autocorrelation matrix of noise and κ is the normalisation factor.

Although the maximum SNR can be obtained by any choice of constant κ , we choose the constant by requiring that:

(a) $\mathbf{h}^T \mathbf{s} = 1$, which yields $\kappa = 1 / \mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}$

(b) $E(v_0^2[n]) = \mathbf{h}^T \mathbf{R}_v \mathbf{h} = 1$, which yields $\kappa = 1 / \sqrt{\mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}}$.

```

s = [1, -1, 1, -1]';
p = numel(s); % Signal length

% The autocorrelation matrix must be p x p since
% its inverse is multiplied by a p-tap signal s(n)
ell = 0:p-1;
r_vv = 0.8.^ell;

```

```

R_vv = toeplitz(r_vv);

% Compute normalisation factor (b)
k = 1/sqrt(s'*(R_vv\s)); % Same as 1/sqrt(s'*inv(R_vv)*s)

% Compute the filter
h = k*(R_vv\s); % Same as k*inv(R_vv)*s

% Print the matched filter coefficients
h

```

$h = 4 \times 1$
 0.9449
 -1.7008
 1.7008
 -0.9449

The optimum SNR is given by:

$$\text{SNR}_o = a^2 \tilde{s}^T \tilde{s} = a^2 s^T R_v^{-1} s. \quad (14.98)$$

Assuming the attenuation factor $a = 1$:

```

a = 1;
SNR = a^2 * s' * (R_vv\s)

```

SNR = 28.0000

[✓] 2) Can the signal to noise ratio be improved by using more than two blocks?

2. The $s(n)$ signal consists of two blocks each containing 1 and -1. Can the signal to noise ratio be improved by using more than two blocks?

Yes, the SNR can be improved. With three blocks, the SNR is increased to 46:

```

s = [1, -1, 1, -1, 1, -1]';
p = numel(s);
ell = 0:p-1;
r_vv = 0.8.^ell;
R_vv = toeplitz(r_vv);
k = 1/sqrt(s'*(R_vv\s));
h = k*(R_vv\s);

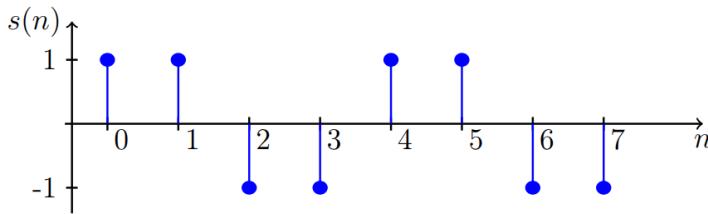
a = 1;
SNR = a^2 * s' * (R_vv\s)

```

SNR = 46.0000

Exam 2017 Problem 3: Detect presence of signal using matched filter

Consider the deterministic signal, $s(n)$ shown below in blue. The signal is zero for all other values of n .



The signal is distorted by additive low frequency noise with autocorrelation $r_v(\ell) = 0.4^{|\ell|}$.

1) Design a matched filter for detecting the presence of the signal and determine the improvement in signal to noise ratio.

The impulse response of the matched filter is given by:

$$\mathbf{h} = \kappa \mathbf{R}_v^{-1} \mathbf{s}. \quad (14.97)$$

where \mathbf{R}_v is autocorrelation matrix of noise and κ is the normalisation factor.

Although the maximum SNR can be obtained by any choice of constant κ , we choose the constant by requiring that:

- (a) $\mathbf{h}^T \mathbf{s} = 1$, which yields $\kappa = 1/\mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}$
- (b) $E(v_o^2[n]) = \mathbf{h}^T \mathbf{R}_v \mathbf{h} = 1$, which yields $\kappa = 1/\sqrt{\mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}}$.

```

s = [1, 1, -1, -1, 1, 1, -1, -1]';
p = numel(s); % Signal length

% The autocorrelation matrix must be p x p since
% its inverse is multiplied by a p-tap signal s(n)
ell = 0:p-1;
r_vv = 0.4.^ell;
R_vv = toeplitz(r_vv);

% Compute normalisation factor (b)
k = 1/(s'*(R_vv\s));      % Using (a)
k = 1/sqrt(s'*(R_vv\s)); % Using (b)

% Compute the filter
h = k*(R_vv\s); % Same as k*inv(R_vv)*s

% Print the matched filter coefficients
h

```

```

h = 8x1
0.2292
0.4431
-0.4431
-0.4431
0.4431
0.4431
-0.4431
-0.2292

```

The optimum SNR at the output is given by:

$$\text{SNR}_o = a^2 \tilde{s}^T \tilde{s} = a^2 s^T R_v^{-1} s. \quad (14.98)$$

Assuming the attenuation factor $a = 1$:

```

a = 1;
SNR_o = a^2 * s' * (R_vv\s)

```

```
SNR_o = 9.7143
```

The SNR at the input is given by:

$$\text{SNR}_i = \frac{\text{(Value of signal at } n = n_0)^2}{\text{power of noise}} = \frac{s^2(n = n_0)}{r_v(0)}$$

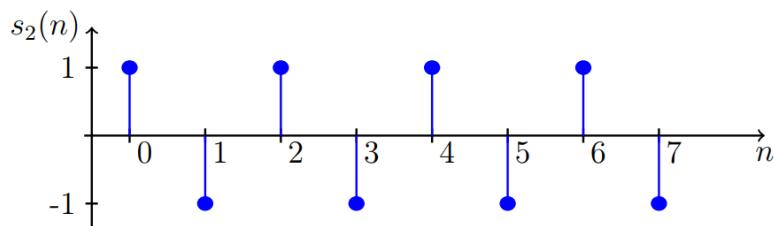
```
SNR_i = s(1)^2/r_vv(1)
```

```
SNR_i = 1
```

The improvement in SNR is 9.7 (almost 10 times)

2) Discuss the improvement of SNR if another signal is used

A second signal $s_2(n)$ is given by



2. Discuss whether the signal to noise ratio will be improved if the signal $s_2(n)$ is used instead of $s(n)$.

If we use $s_2(n)$ instead of $s(n)$, the output SNR becomes 17.3:

```

s = [1, -1, 1, -1, 1, -1, 1, -1]';
p = numel(s); % Signal length
ell = 0:p-1;
r_vv = 0.4.^ell;
R_vv = toeplitz(r_vv);
a = 1;
SNR_o = a^2 * s' * (R_vv\s)

SNR_o = 17.3333

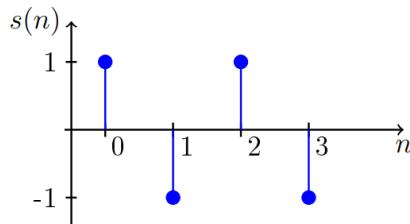
```

The $s_2(n)$ is similar to $s(n)$ but it oscillates faster i.e., has higher frequency than the $s(n)$ signal. Since the noise is at low frequency, it is easier to separate the $s_2(n)$ signal from the noise. This means that a higher SNR can be expected from a signal with a higher frequency.

That is why the output SNR for $s_2(n)$ is better than that for $s(n)$.

Exam 2016 Problem 2: Detect the presence of signal via Matched Filter

Consider the deterministic signal, $s(n)$ shown below in blue. The signal is zero for all other values of n .



Assume that the signal is distorted by additive low frequency noise with autocorrelation $r_v(l) = 0.3^{|l|}$.

```
clear variables;
```

1) Design a matched filter for detecting the presence of the signal and calculate the optimum signal to noise ratio.

The impulse response of the matched filter is given by:

$$\mathbf{h} = \kappa \mathbf{R}_v^{-1} \mathbf{s}. \quad (14.97)$$

where \mathbf{R}_v is autocorrelation matrix of noise and κ is the normalisation factor.

Although the maximum SNR can be obtained by any choice of constant κ , we choose the constant by requiring that:

- (a) $\mathbf{h}^T \mathbf{s} = 1$, which yields $\kappa = 1/\mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}$
(b) $E(v_o^2[n]) = \mathbf{h}^T \mathbf{R}_v \mathbf{h} = 1$, which yields $\kappa = 1/\sqrt{\mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}}$.

```
s = [1, -1, 1, -1]';
p = numel(s); % Signal length

% The autocorrelation matrix must be p x p since
% its inverse is multiplied by a p-tap signal s(n)
ell = 0:p-1;
r_vv = 0.3.^ell;
R_vv = toeplitz(r_vv);

% Compute normalisation factor (b)
k = 1/(s'*(R_vv\s));      % Using (a)
% k = 1/sqrt(s'*(R_vv\s)); % Using (b)

% Compute the filter
h = k*(R_vv\s); % Same as k*inv(R_vv)*s

% Print the matched filter coefficients
h
```

h = 4x1
0.2174
-0.2826
0.2826
-0.2174

The optimum SNR at the output is given by:

$$\text{SNR}_o = a^2 \tilde{\mathbf{s}}^T \tilde{\mathbf{s}} = a^2 \mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}. \quad (14.98)$$

Assuming the attenuation factor $a = 1$:

```
a = 1;
SNR_o = a^2 * s' * (R_vv\s)
```

SNR_o = 6.5714

The SNR at the input is given by:

$$\text{SNR}_i = \frac{(\text{Value of signal at } n = n_0)^2}{\text{power of noise}} = \frac{s^2(n = n_0)}{r_v(0)}$$

```
SNR_i = s(1)^2/r_vv(1)
```

SNR_i = 1

The improvement in SNR is slightly over 6.5.

2) Will SNR improve if the longer signal is used?

2. Discuss whether the signal to noise ratio will be improved if the longer signal $s_2(n) = \{1, -1, 1, -1, 1, -1\}$ is used instead of $s(n)$.

We can perform the calculations agains:

```
s = [1, -1, 1, -1, 1, -1]';  
  
p = numel(s); % Signal length  
ell = 0:p-1;  
r_vv = 0.3.^ell;  
  
R_vv = toeplitz(r_vv);  
a = 1;  
SNR_o = a^2 * s' * (R_vv\s)  
  
SNR_o = 10.2857
```

We see that the SNR is increased to 10.3.

So when the signal length is increased, a longer filter can be used. A longer filter implies more signal energy in the filter. Furthermore, the filter can be better tailored to the noise spectrum. Therefore, the signal energy is increased and the noise power is decreased which results in a better SNR.

3) Discuss whether the SNR will increase given a different ACRS?

First, we observe that the signal $s(n)$ us a high-frequency signal because of the alternative sign of the signal.

The noise process $r_v(\ell) = 0.3^{|\ell|}$ is low-frequecy noise. Since the signal is high frequency and the noise is low-frequency, it is easier to separate the signal and the noise.

The another noise process $r_v(\ell) = (-0.3)^{|\ell|}$ is a high-frequency noise. Separating noise from the signal is more difficult due to the overlapping spectra.

We confirm this by calculating the SNR:

```
s = [1, -1, 1, -1, 1, -1]';  
s = flip(s);  
p = numel(s); % Signal length  
ell = 0:p-1;  
r_vv2 = (-0.3).^ell;  
R_vv2 = toeplitz(r_vv2);  
a = 1;  
SNR_o = a^2 * s' * (R_vv2\s)  
  
SNR_o = 3.6923
```

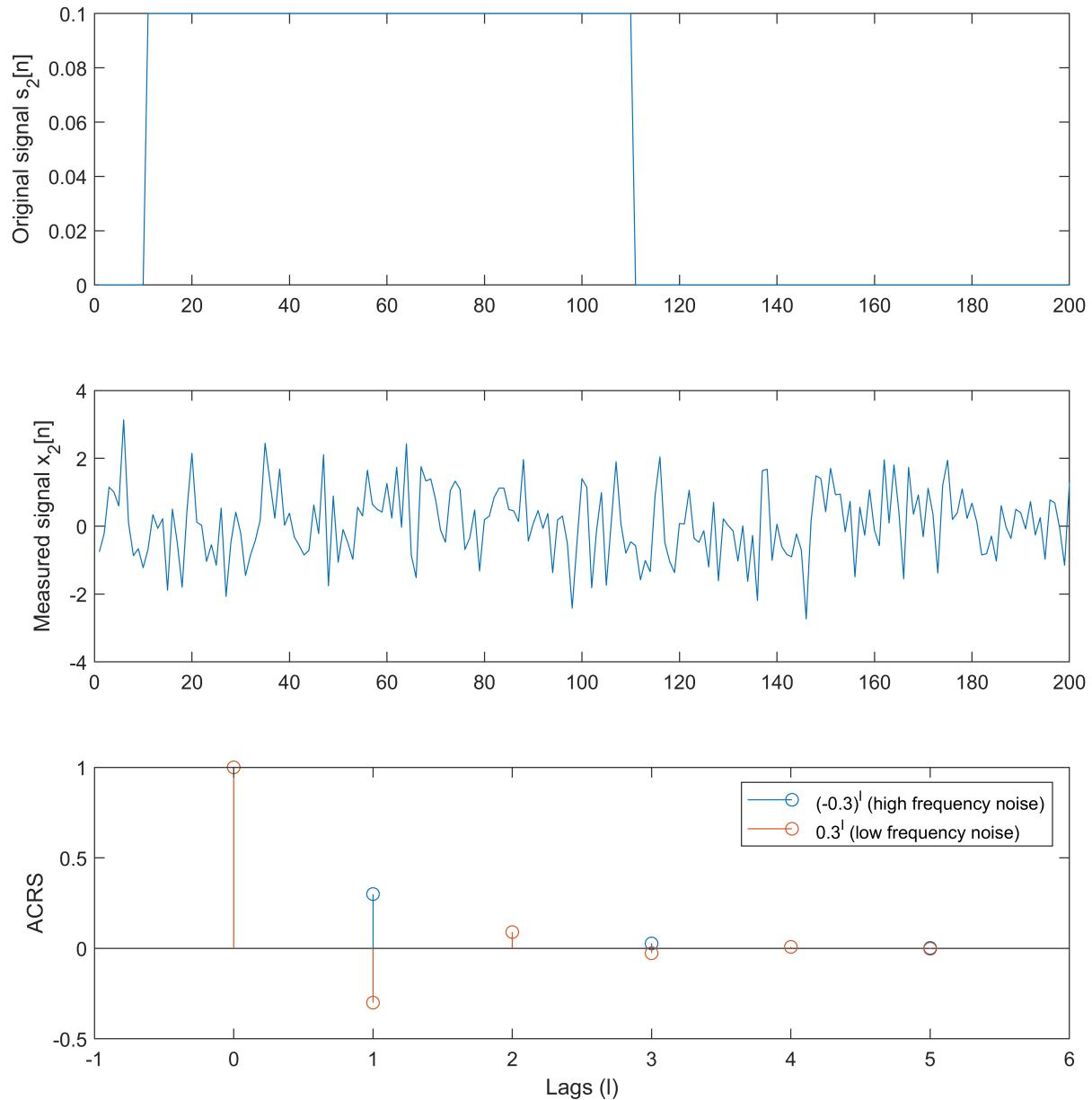
Characteristics of high-frequency noise ACRS vs low-frequency noise.

```
stem(ell, r_vv)
```

```

hold on;
stem(ell, r_vv2)
hold off;
legend('(-0.3)^l (high frequency noise)', '0.3^l (low frequency noise)')
xlim([min(ell)-1, max(ell)+1])
hold off;
xlabel('Lags (l)')
ylabel('ACRS')

```



Exam 2013, Problem 3, matched filter given ACRS for noise, SNR optimum vs non-optimum filter

Let a deterministic signal $s(n)$ be given by

$$s(n) = \begin{cases} 1 & n = 0 \\ 2 & n = 1 \\ 3 & n = 2 \\ 0 & \text{elsewhere} \end{cases}$$

Assume that the signal is corrupted by additive noise and that the autocorrelation function of the noise is $r_v(l) = 0.6^{|l|}$.

```
clear variables;
```

1) Determine the matched filter for detecting the presence or absence of the signal

Determine the matched filter for detecting the presence or absence of the signal in the noise at time n_0 and comment on the shape of its magnitude response.

The impulse response of the matched filter can be computed using Eq. 14.97:

$$\mathbf{h} = \kappa \mathbf{R}_v^{-1} \mathbf{s}. \quad (14.97)$$

where κ is the normalisation factor. Although the maximum SNR can be obtained by any choice of constant κ , we choose the constant by requiring that:

- (a) $\mathbf{h}^T \mathbf{s} = 1$, which yields $\kappa = 1/\mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}$
- (b) $E(v_o^2[n]) = \mathbf{h}^T \mathbf{R}_v \mathbf{h} = 1$, which yields $\kappa = 1/\sqrt{\mathbf{s}^T \mathbf{R}_v^{-1} \mathbf{s}}$.

```
s = flip([1, 2, 3])';
p = numel(s);
ell = 0:p-1;
r_vv = 0.6.^abs(ell);
R_vv = toeplitz(r_vv)
```

```
R_vv = 3x3
1.0000    0.6000    0.3600
0.6000    1.0000    0.6000
0.3600    0.6000    1.0000
```

```
% Compute normalisation factor
```

```
k = 1/sqrt(s'*(R_vv\s)) % Same as 1/sqrt(s'*inv(R_vv)*s)
```

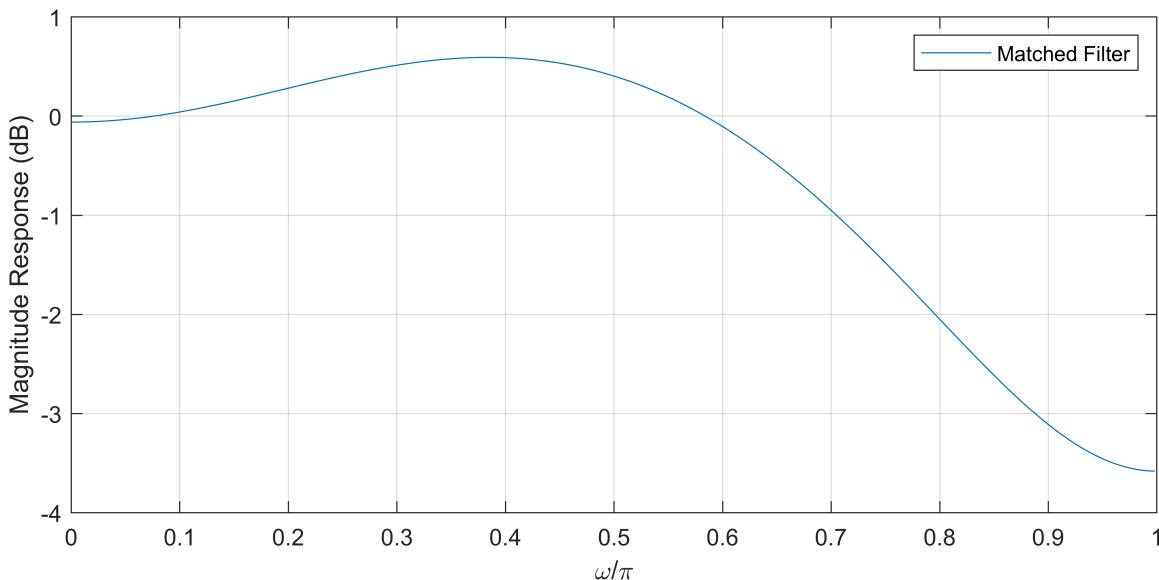
```
k = 0.3310
```

```
% Compute the filter
h = k*(R_vv\s)
```

```
h = 3x1
0.9311
0.1655
-0.1035
```

1a) Comment on the shape of the magnitude response

```
% Plot the magnitude response
figure('position', [0, 0, 800, 350])
[H_h, w_h] = freqz(h, 1);
plot(w_h/pi, 20*log10(abs(H_h)))
legend('Matched Filter')
xlabel('\omega/\pi')
ylabel('Magnitude Response (dB)')
grid on;
```



The matched filter passes DC components almost undistorted, has a small gain at midrange frequencies and attenuates high frequency components.

The reason for this shape is a trade-off between

- (A) the noise is centered at low frequencies, and
- (B) the signal that is also somewhat low-pass in nature as evident from the identical signs of all samples of the signal.

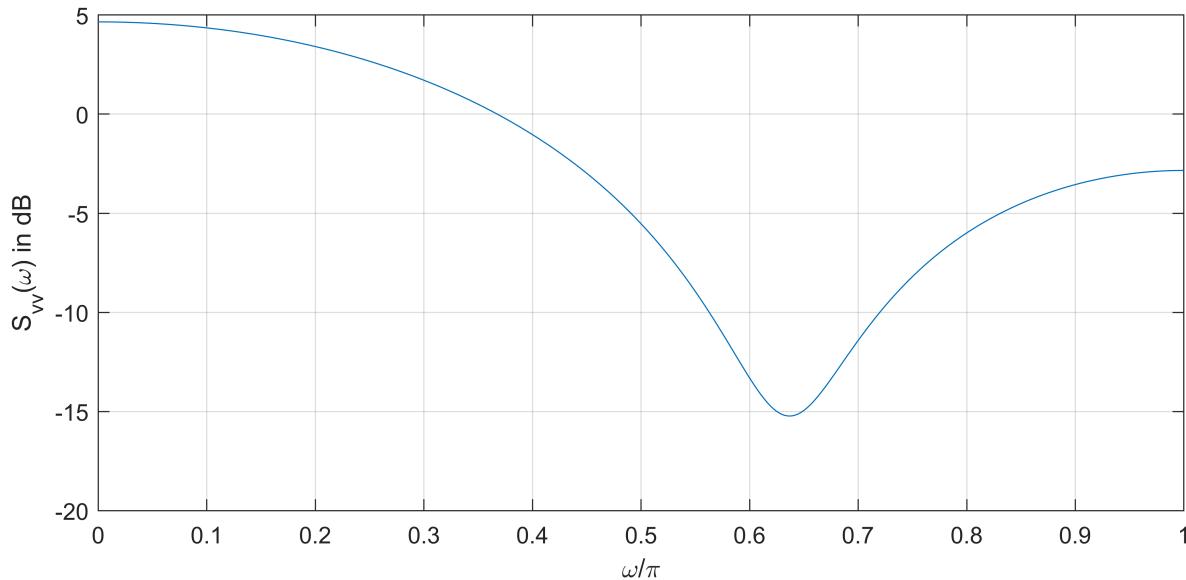
Assume the noise process is an MA(q) process, the PSD can be computed using Eq. (13.119):

$$S_{xx}(\omega) = r_{xx}[0] + 2 \sum_{\ell=1}^{\infty} r_{xx}[\ell] \cos \omega \ell, \quad (13.119)$$

By plotting the PSD of the noise, we observe that the noise is centred at low frequencies.

```
w = 0:0.001:pi;
S = r_vv(1);
for l = 2:numel(r_vv)
    S = S + 2 * r_vv(l)*cos(w*(l-1));
end

figure('position', [0, 0, 800, 350])
plot(w/pi, pow2db(S))
xlabel('\omega/\pi')
ylabel('S_{vv}(\omega) in dB')
clear figure;
grid on;
```



2) Calculate the optimum signal to noise ratio

The optimum SNR can be compute using Eq. 14.98:

$$\text{SNR}_o = a^2 \tilde{s}^T \tilde{s} = a^2 s^T R_v^{-1} s. \quad (14.98)$$

Since the attenuation factor a is not given in this problem, we assume $a = 1$:

```
SNR_opt = s'* (R_vv \ s)
```

```
SNR_opt = 9.1250
```

3) Compute the SNR of a non-optimal filter

Assume that a non-optimum filter $h^T = [1 \ 0 \ 1]$ is used rather than the optimum filter.

3. Determine the decrease in signal to noise ratio when this filter is used instead of the optimum filter.

We can compute the SNR using Eq. (19.94):

$$\text{SNR}_o = \frac{s_o^2[n_0]}{\text{E}(v_o^2[n_0])} = a^2 \frac{(h^T s)^2}{h^T R_v h} \quad (14.94)$$

```
h_nopt = [1, 0, 1]';  
SNR_nopt = (h_nopt'*s)^2 / (h_nopt'*R_vv*h_nopt)
```

```
SNR_nopt = 5.8824
```

Functions

```
function [S, w] = ar2psd(a, v, N)  
% AR2PSD Compute the Power Spectral Density from AR(p) coefficients  
% [S, w] = ar2psd(a, v, N)  
% a: AR(p) coefficients  
% v: the variance  
% N: number of points in the range [1, pi]  
% S: the estimated power spectrum  
% w: frequencies  
w = linspace(0, 1, N) * pi;  
  
% Compute the transfer function  
% Used Eq. (13.133) in the book  
H = ones(N, 1);  
for k=1:numel(a)  
    H = H + a(k)*exp(-1j * w' * k);  
end  
H = 1./H;
```

```

% Finally compute the PSD
S = v * H.*conj(H);
end

function [h, SNR] = matched_filter(s, v, a)
% MATCHED_FILTER: Compute the impulse response of a matched filter and
%                 and the corresponding output SNR.
% [h, SNR] = matched_filter(s, v, a)
% s: the signal
% v: a realisation of the additive noise
% a: the attenuation factor (default=1)
% h: the impulse response of the matched filter
% SNR: the output SNR
if nargin < 3
    a = 1;
end

p = numel(s); % Signal length

% The autocorrelation matrix must be p x p since
% its inverse is multiplied by a p-tap signal s(n)
[r_vv, ~] = xcorr(v, p-1, 'biased');
R_vv = toeplitz(r_vv(p:end));

% The expression `R_vv^{-1} * s` is used multiple times,
% so compute it once and reuse
R_vv_inv_s = R_vv\s;

% Compute normalisation factor
k = 1/sqrt(s'*R_vv_inv_s);

% Compute the filter
h = k*R_vv_inv_s;

% Compute the output SNR
SNR = a^2 * s' * R_vv_inv_s;
end

function [x, v, s] = gen_noisy_signal(s_original, N, noise_var, D)
% Embeds a signal into a realisation of a WGN process
% s_original: the original signal that needs to be embedded
% N:          the number of samples of the noisy signal
% noise_var:  the variance of WGN (default=1)
% D:          the position of embedding
if nargin < 4
    D = 0;
end
if nargin < 3
    noise_var = 1;
end

n = (1:N)';
p = numel(s_original); % Signal length

% Generate the signal.
% Ensure that s[n] is zero when n is outside the interval [1, p]
s = zeros(N, 1);
s(n >= D+1 & n <= p+D) = s_original;

```

```
% Generate the zero-mean WGN  
v = sqrt(noise_var) * randn(N, 1);  
  
% Generate the measured signal x[n]  
x = v + s;  
end
```

Multirate Signal Processing

Table of Contents

Multirate Signal Processing.....	1
Sampling rate conversion.....	1
Downsampling.....	2
[»] Problem 12.21: Resampling downsampled sequences.....	2
[»] Problem 12.22: Resample decimated sequences.....	2
[✓] Problem 12.26: Interpolation using MATLAB's interp function.....	3
a) Interpolate signal using MATLAB's interp function.....	3
b) Plot the frequency response of the lowpass filter.....	5
[»] Problem 12.19: Compare spectrum of a downsampled signal with the original.....	6
a) Compute the spectrum of the original signal and plot its magnitude.....	6
a) Compute the spectrum of the signal and plot its magnitude.....	8
b) Compute the spectrum of the downsampled signal and plot its magnitude.....	8
c) Compare the two spectra.....	8
ADSI Problem 8.1: Upsampling with linear interpolation.....	8
1. Upsample the signal using $I = 4$ and the linear interpolation kernel given by Eq. 12.48.....	9
2. Sketch the signal before and after the upsampling and interpolation.	10

Multirate Signal Processing

Discrete-time systems with different sampling rates at various parts of the system are called *multirate systems*.

The fundamental operations for changing the sampling rate are:

- decimation
- interpolation

Sampling rate conversion

Conceptually, the sampling rate conversion process can be regarded as a two-step operation:

1. the discrete-time signal is reconstructed into a continuous-time signal
2. the signal is resampled at a different sampling rate

In practice, the conversion is implemented using discrete-time signal processing without actual reconstruction of any continuous-time signal.

The term *resampling* is used to refer to the process of changing the *sampling rate* of a discrete-time signal without reconstructing the equivalent continuous-time signal.

There are three resampling operations:

- $T_0 = DT$, downsampling i.e., decreasing the sampling rate by an integer factor D
- $T_0 = \frac{T}{I}$, upsampling i.e., increasing the sampling rate by an integer factor I
- $T_0 = T\left(\frac{D}{I}\right)$, changing the sampling rate by non-integer factor

where T is the sampling rate of the original signal, T_0 is the new sampling rate, and D and I are integers.

Downsampling

[»] Problem 12.21: Resampling downsampled sequences

Using the `downsample` function, resample the following sequences using the given parameters D and the offset k . Using the `stem` function, plot the original and downsampled signals.

- (a) $x[n] = \sin(0.2\pi n)$, $0 \leq n \leq 50$, $D = 4$, $k = 0$, and $k = 2$.
- (b) $x[n] = \cos(0.3\pi n)$, $0 \leq n \leq 60$, $D = 3$, $k = 0$, and $k = 1$.

[»] Problem 12.22: Resample decimated sequences

Using the `decimate` function, resample the following sequences using the given parameters D . Using the `stem` function, plot the original and decimated signals. Obtain results using both the default IIR and FIR decimation filters and comment on the results.

- (a) $x[n] = \cos(0.4\pi n)$, $0 \leq n \leq 100$, $D = 2$.
- (b) $x[n] = \sin(0.15\pi n)$, $0 \leq n \leq 100$, $D = 3$.

[✓] Problem 12.26: Interpolation using MATLAB's interp function

```
clear variables;
```

Let $x[n] = 2 \cos(0.1\pi n) + \sin(0.05\pi n)$, $0 \leq n \leq 60$.

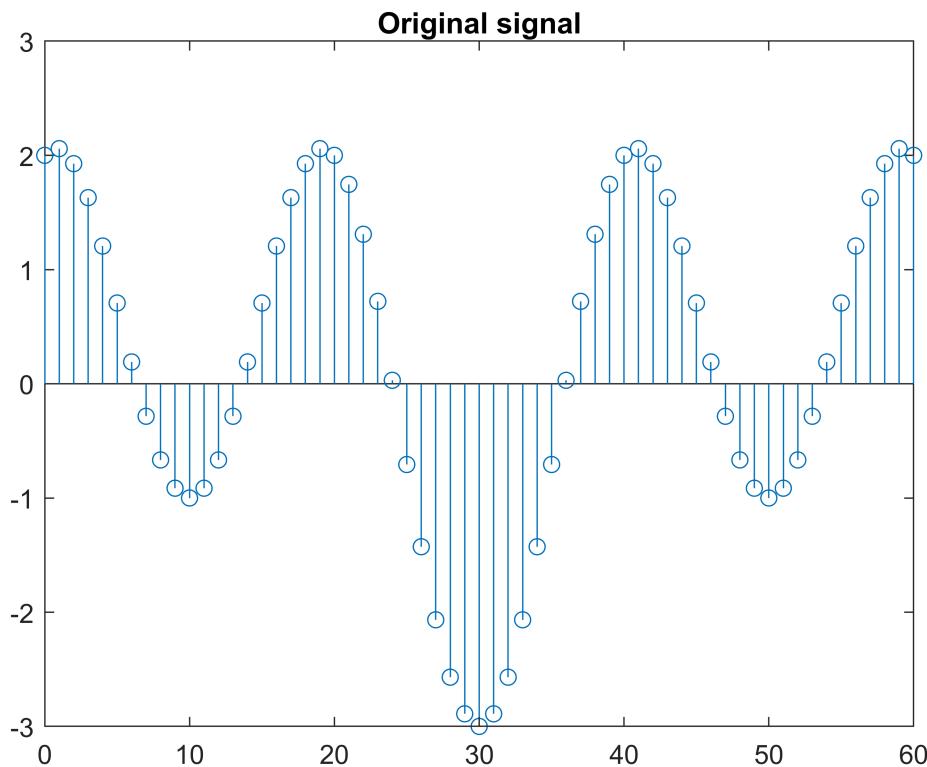
- (a) Using the `interp` function, interpolate using $I = 3$, $I = 6$, and $I = 9$. Stem plot the original and interpolated signals.
- (b) Using the second output argument of the `interp` function, plot the frequency response of the lowpass filter used in each of the above interpolations.

a) Interpolate signal using MATLAB's interp function

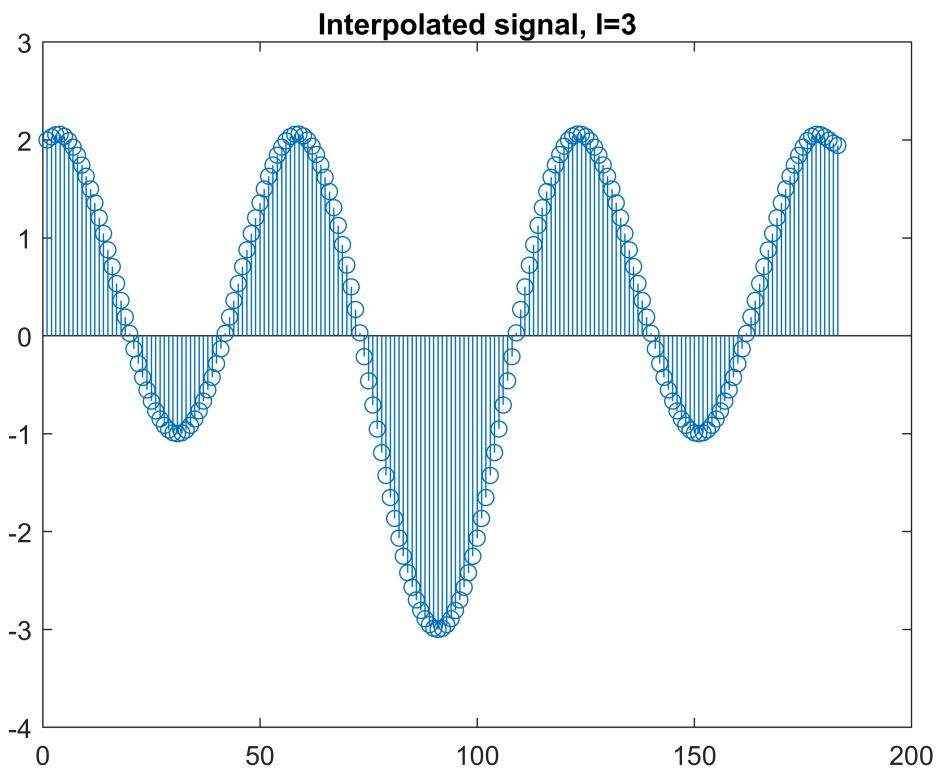
- (a) Using the `interp` function, interpolate using $I = 3$, $I = 6$, and $I = 9$. Stem plot the original and interpolated signals.

```
n = 0:60;
x = 2*cos(0.1*pi*n) + sin(0.05*pi*n);

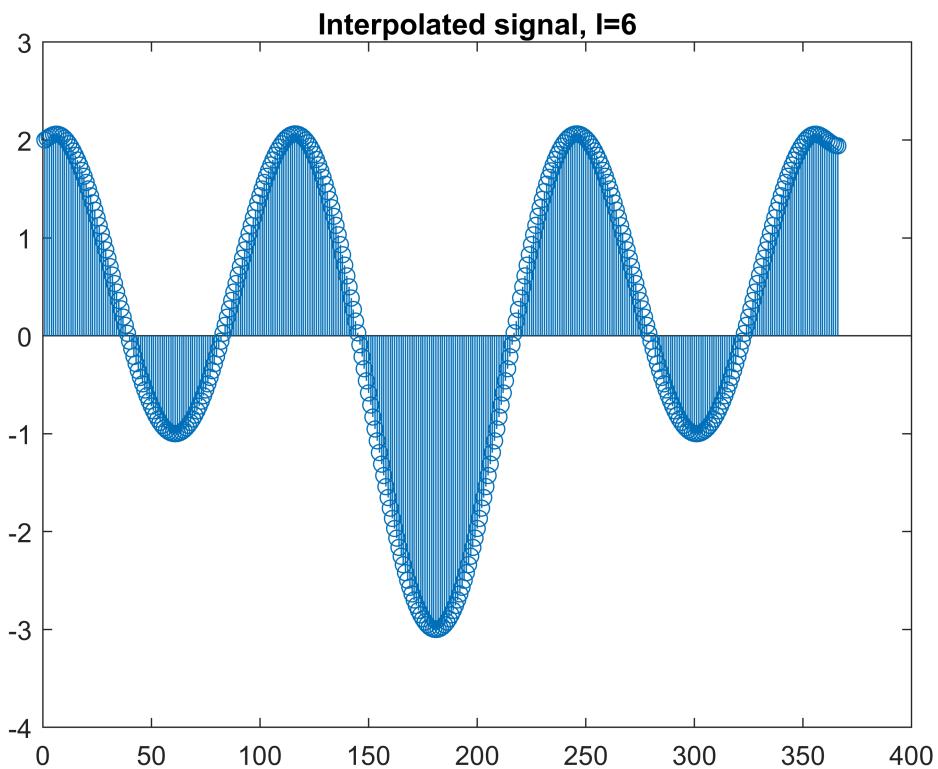
stem(n, x)
title('Original signal')
```



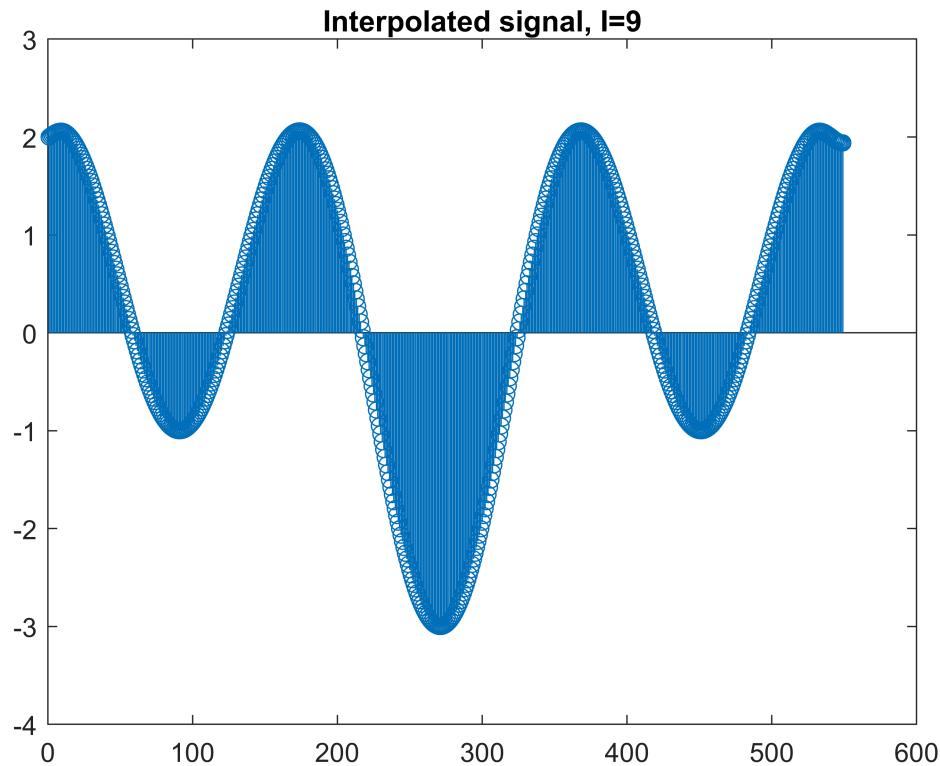
```
stem(interp(x, 3))
title('Interpolated signal, I=3')
```



```
stem(interp(x, 6))
title('Interpolated signal, I=6')
```



```
stem(interp(x, 9))
title('Interpolated signal, I=9')
```

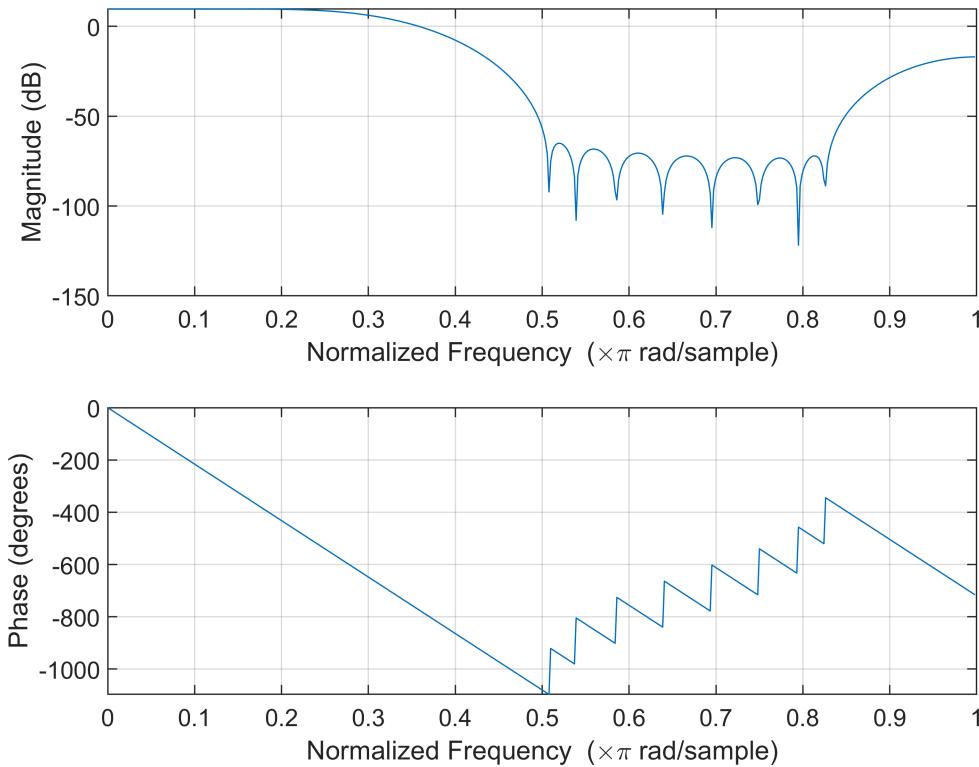


b) Plot the frequency response of the lowpass filter

- (b) Using the second output argument of the `interp` function, plot the frequency response of the lowpass filter used in each of the above interpolations.

`[y, b] = interp(x, r, n, cutoff)` returns a vector, `b`, with the filter coefficients used for the interpolation.

```
[y, b] = interp(x, 3);
freqz(b, 1)
```



»] Problem 12.19: Compare spectrum of a downsampled signal with the original

```
clear variables;
```

Consider the signal $x[n] = 0.9^n u[n]$. It is to be downsampled by a factor of $D = 3$ to obtain $x_D[n]$.

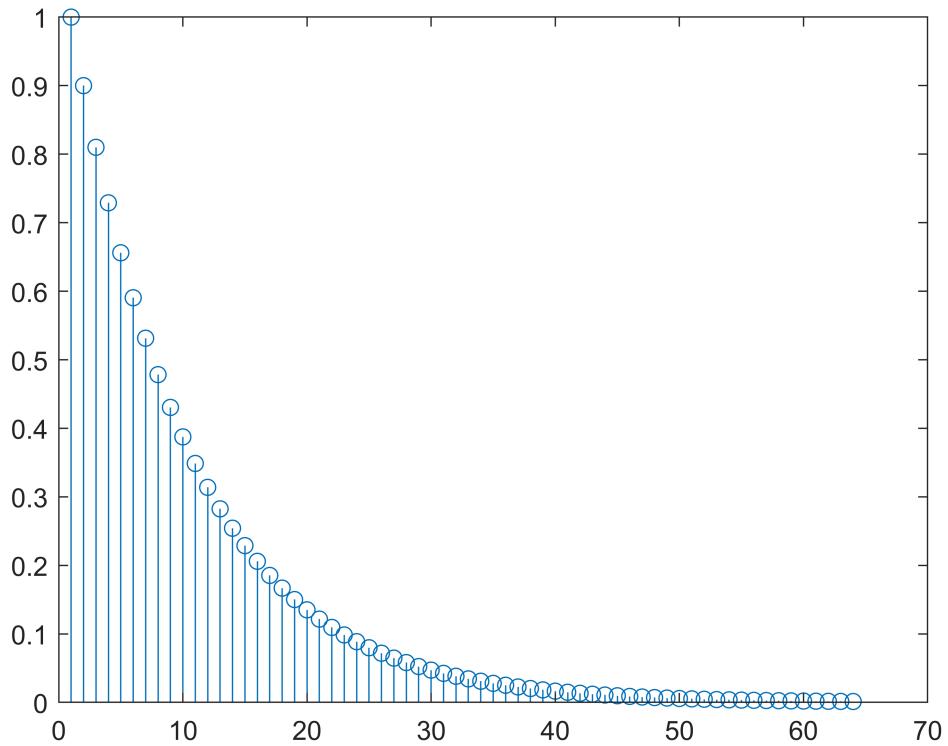
- (a) Compute the spectrum of $x[n]$ and plot its magnitude.
- (b) Compute the spectrum of $x_D[n]$ and plot its magnitude.
- (c) Compare the two spectra.

The *unit step* sequence is given by

$$u[n] \triangleq \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

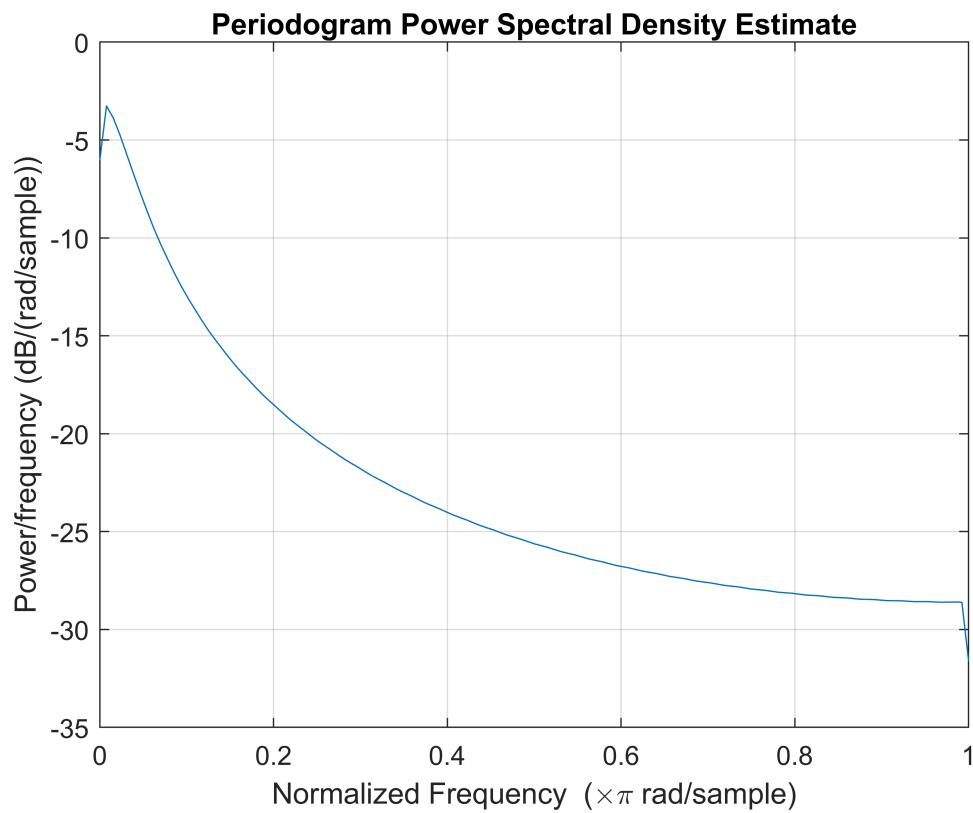
- a) Compute the spectrum of the original signal and plot its magnitude

```
N = 64;
n = 0:N-1;
x = 0.9.^n;
% x = exp(-n);
stem(x)
```



[?] How do you compute the spectrum of the signal?

```
periodogram(x)
```



- a) Compute the spectrum of the signal and plot its magnitude
- b) Compute the spectrum of the downsampled signal and plot its magnitude
- c) Compare the two spectra

ADSI Problem 8.1: Upsampling with linear interpolation

```
clear variables;
```

Consider a signal given by:

$$x[n] = \begin{cases} -1 & n = 1 \\ 3 & n = 2 \\ 0 & \text{elsewhere} \end{cases}$$

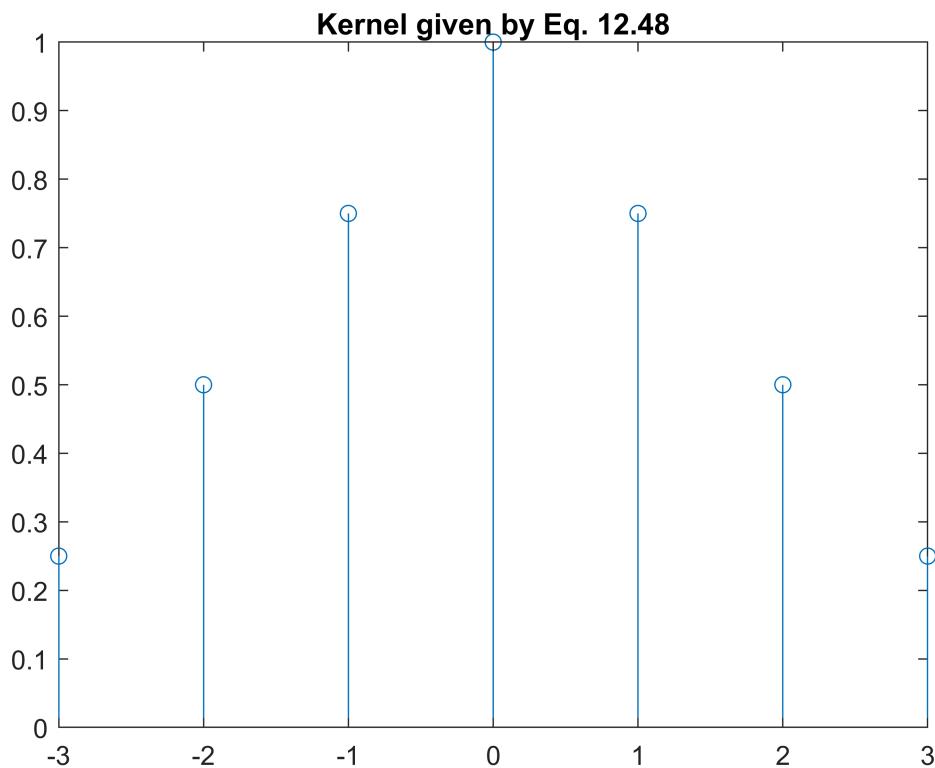
```
N = 10;
n = 1:N;
x = zeros(N,1);
x(1) = -1;
x(2) = 3;
```

1. Upsample the signal using $I = 4$ and the linear interpolation kernel given by Eq. 12.48.

$$g_{\text{lin}}[n] \triangleq \begin{cases} 1 - \frac{|n|}{I}, & -I < n < I \\ 0, & \text{otherwise} \end{cases} \quad (12.48)$$

```
I = 4;
gn = -I+1:1:I-1;
g = 1 - abs(gn)/I; % g = triang(I*2-1)

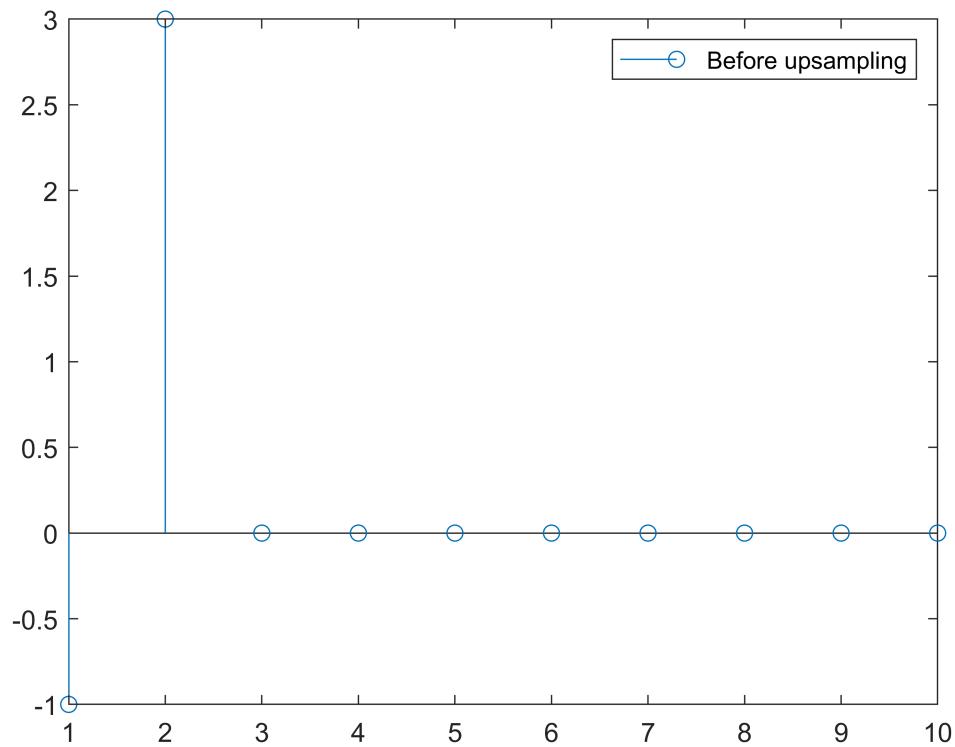
stem(gn, g);
title('Kernel given by Eq. 12.48')
```



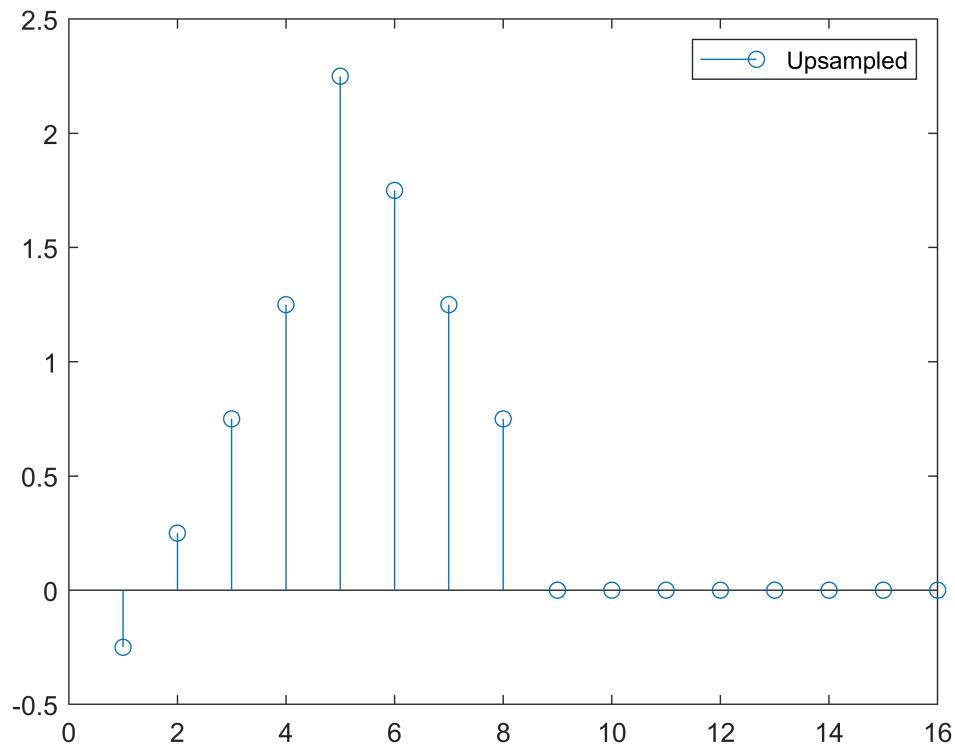
```
x_upsampled = conv(x, g);
```

2. Sketch the signal before and after the upsampling and interpolation.

```
stem(n, x)
legend('Before upsampling')
```



```
stem(x_upsampled)
legend('Upsampled')
```



Noise

Table of Contents

White Noise.....	1
Generating a realisation of the white Gaussian noise in MATLAB.....	1
Expected value of White Noise.....	2
Autocorrelation of White Noise.....	2
Variance of white noise.....	2
Cross-correlation of two uncorrelated noise processes.....	2
Power spectral density of white noise.....	3
Coloured Noise.....	3
How to generate coloured noise?.....	3
Problems.....	5
Quiz: is white noise cubed also a white noise?.....	5
Quiz: is a sequence drawn from a probability distribution white noise?.....	10

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.

Generating a realisation of the white Gaussian noise in MATLAB

To generate N random numbers for a Gaussian random variable with mean μ and variance σ^2 we use following MATLAB functions

```
clear variables;

N = 1000;
variance = 25;
std_dev = sqrt(variance);
mu = 500;
w = std_dev.*randn(N,1) + mu;

% Calculate the sample mean, standard deviation, and variance.
stats = [mean(w) std(w) var(w)]
```



```
stats = 1x3
499.9186    4.9628   24.6291
```

The mean and variance are not 500 and 25 exactly because they are calculated from a sampling of the distribution.

Suppose we have a random process that produces perfect random noise. Let $w(n)$ be a random signal from this process.

Expected value of White Noise

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

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

Autocorrelation of White Noise

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.

Variance of white noise

The variance of a random variable X is defined by:

$$\begin{aligned} \text{var}(X) &= E[(X - m_x)^2] = \int_{-\infty}^{\infty} (x - m_x)^2 f_X(x) dx \\ &= E(X^2) - 2m_x E(X) + m_x^2 = E(X^2) - m_x^2, \end{aligned} \quad (13.11)$$

The variance of white noise which can be computed as follow:

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

Cross-correlation of two uncorrelated noise processes

Let $v(n)$ and $w(n)$ be **two uncorrelated white noise** processes with variance $\sigma_v^2 = 0.49$ and $\sigma_w^2 = 1$. The cross-correlation between these processes is:

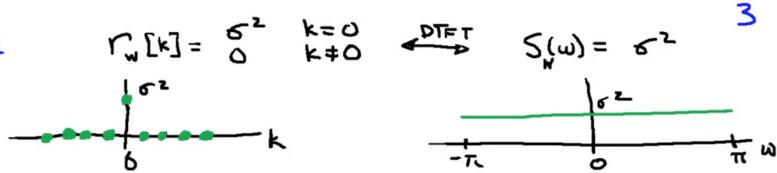
$$r_{vw}(\ell) = 0$$

Power spectral density of white noise

The power spectral density of white noise 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.

Examples

1) white noise



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 $\ell = 0$

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

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

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

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

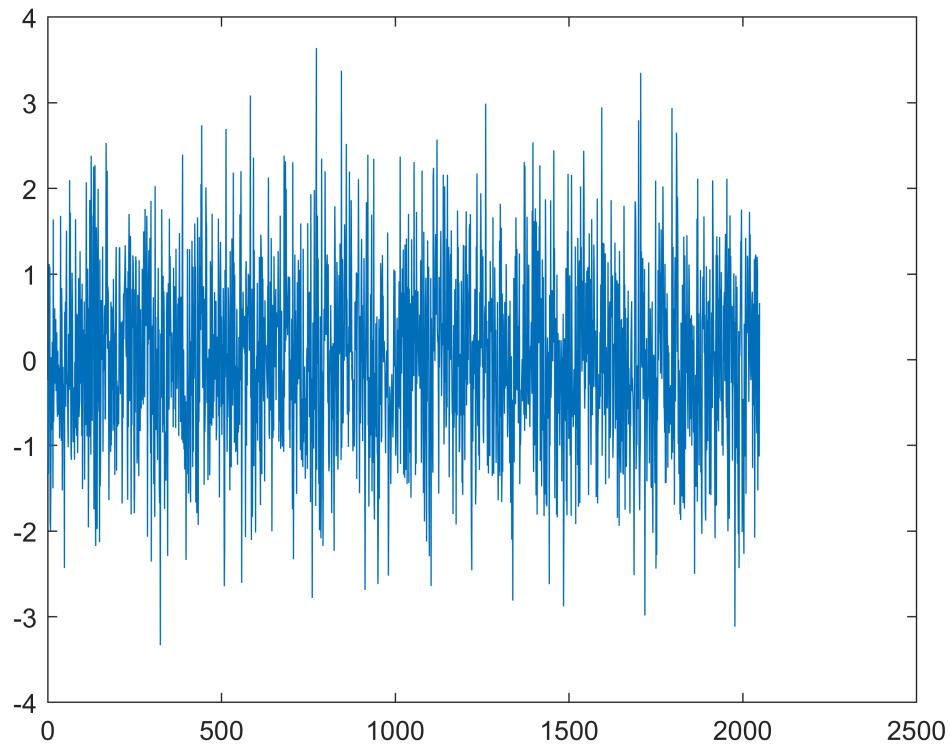
Coloured Noise

How to generate coloured noise?

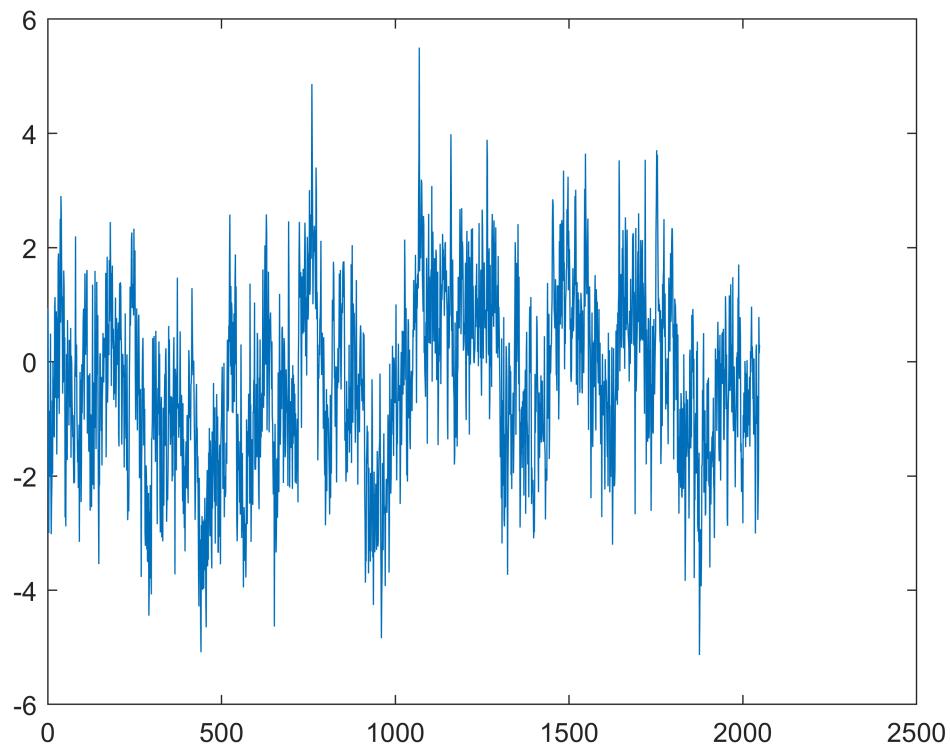
```
N = 2048;
```

```
% Slight coloured noise
```

```
x1 = step(dsp.ColoredNoise('InverseFrequencyPower', 0.1, 'SamplesPerFrame', N));  
plot(x1);
```



```
% Very coloured noise (Pink noise)  
x1 = step(dsp.ColoredNoise('InverseFrequencyPower', 1, 'SamplesPerFrame', N));  
plot(x1);
```



Problems

Quiz: is white noise cubed also a white noise?

```
clear variables;
```

We are used to white Gaussian noise

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

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.

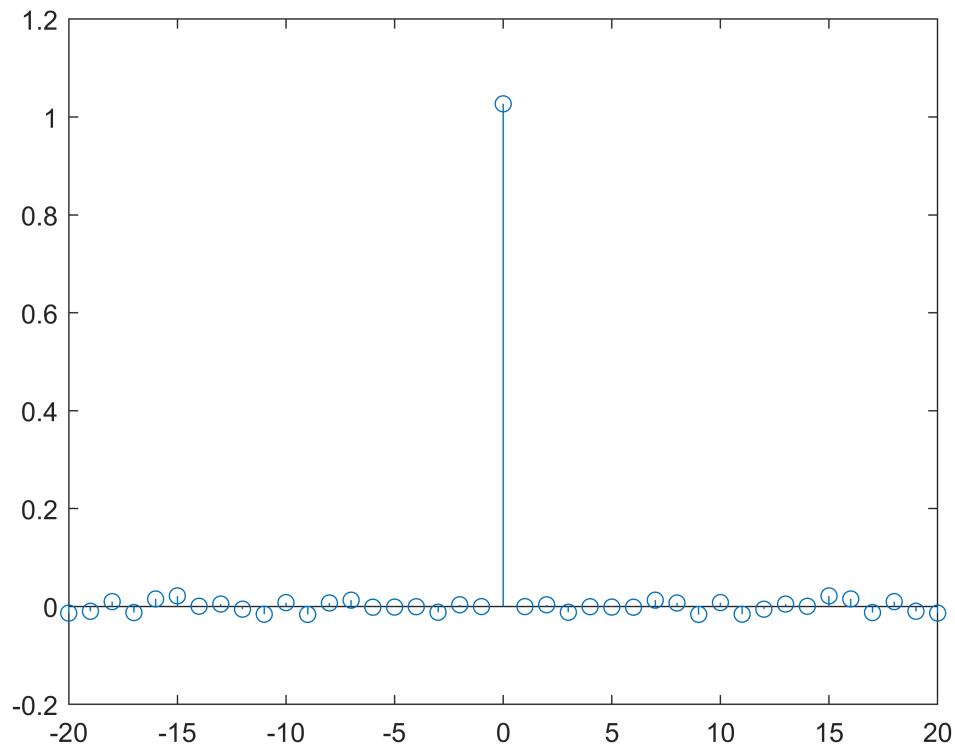
If we plot the autocorrelation of white noise, we should observe a signal similar to delta signal with amplitude of σ_x^2 because:

if $x(n) \sim WGN(0, \sigma_x^2)$ 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);

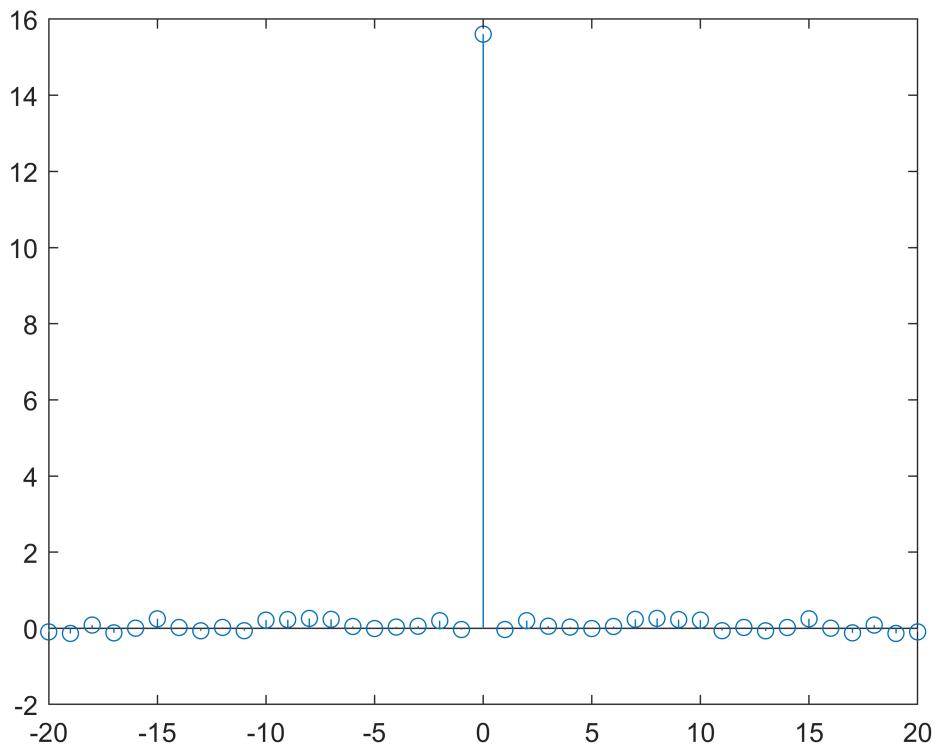
[r_xx, lags] = xcorr(x, 20, 'biased');
stem(lags, r_xx);
```



This looks like what we have expected.

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

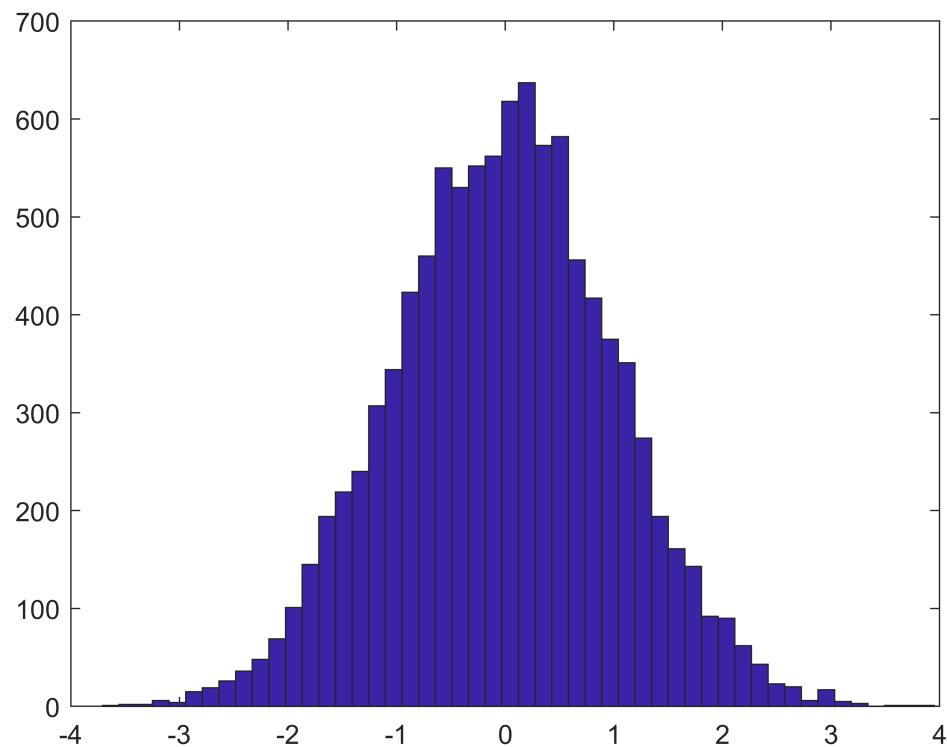
```
y = x.^3;
[r_yy, lags] = xcorr(y, 20, 'biased');
stem(lags, r_yy);
```



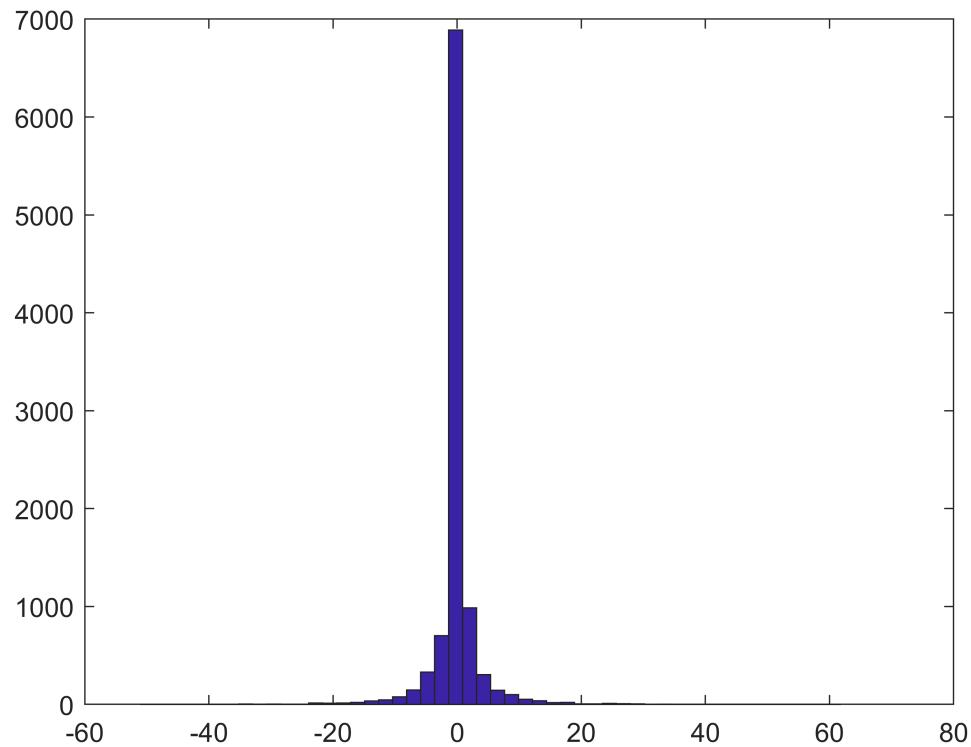
This proves that the answer to the quiz is Yes!

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)
```

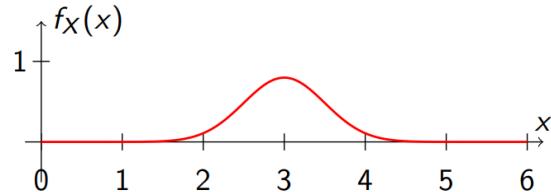


```
hist(y, 50)
```



Quiz: is a sequence drawn from a probability distribution white noise?

Assume we draw samples $\{x(n)\}$ from the following probability density function



Is the sequence $\{x(n)\}$ white noise?

- [A: Yes](#)
- [B: No](#)

Answer: Yes!

The samples are drawn from a Gaussian probability density function, but the mean value is non-zero, i.e. $x(n) = x_0 + x_1(n)$, where $E[x_1(n)] = 0$. Hence

$$\begin{aligned}r_{xx}(l) &= E[x(n)x(n-l)] \\&= E[(x_0 + x_1(n))(x_0 + x_1(n-l))] \\&= x_0^2 + \sigma_{x_1}^2 \delta(l)\end{aligned}$$

Pisarenko

Table of Contents

Difference Equations of Sinusoids.....	1
Pisarenko: harmonic decomposition method.....	2
Exam 2012, Problem 3: PSD Estimation assuming sinusoidal signal in white noise.....	3
1) Estimate PSD assuming sinusoidal white noise.....	4
Exam 2015 Problem 1: Estimate PSD of signal using Pisarenko.....	7
1) Estimate and sketch the PSD of sinusoidal signal in white noise.....	7
2) Calculate the signal to noise ratio.....	11
3) Does an additional value to the ACRS agree with the signal model?.....	11
Exam 2018 Problem 1: Compute PSD from data using Pisarenko (sinusoid with additive white noise).....	12
[✓] 1) Compute and plot the autocorrelation for lags 0 to 4.....	12
[✓] 2) Compute the power spectral density assuming a Pisarenko model.....	13
[✓] 3. Discuss whether the Pisarenko model can be considered appropriate for the given data.....	15
Problem 5.1: Frequency estimation using Pisarenko's method.....	17
Problem 5.2: Pisarenko of two sinusoids (sine) in white noise.....	20
0) Preliminary, compute the autocorrelation function of a sine signal:.....	20
1) Calculate the autocorrelation function.....	21
2) Choose appropriate values for the amplitudes and frequencies and for the noise power.....	22
3) Calculate the eigenvalues of the autocorrelation matrix as a function of its size and compare with your expectation.....	22
4) Use the Pisarenko method to calculate the spectrum and compare with the expected results.....	23
Problem 5.3: Frequency resolution of the Pisarenko method.....	23
1) The stability of PSD estimates using the Pisarenko method.....	24
Problem 5.4: Pisarenko and coloured noise.....	27
1) Create a MATLAB model of a sinusoidal signal in white, slightly coloured and very coloured noise.....	27
2) Compare the eigenvalues of the autocorrelation matrix for the three different scenarios.....	27
3) Calculate the Pisarenko spectra and discuss whether Pisarenko is useful when the noise is coloured.....	28
Problem 5.5: Pisarenko, wrong choice of eigenvector.....	29
1) How does the use of a wrong eigenvalue influence the solution?.....	29
Functions.....	32

Difference Equations of Sinusoids

A signal consisting of p sinusoidal components has the difference equation:

$$x(n) = - \sum_{m=1}^{2p} a_m x(n-m) \quad (14.5.2)$$

This corresponds to a system with the system function:

$$H(z) = \frac{1}{1 + \sum_{m=1}^{2p} a_m z^{-m}} \quad (14.5.3)$$

From the polynomial $A(z)$, we observe that the system has $2p$ poles on the unit circle which correspond to the frequencies of the sinusoids

$$A(z) = 1 + \sum_{m=1}^{2p} a_m z^{-m} \quad (14.5.4)$$

Now, suppose that the sinusoids are corrupted by a white noise sequence $w(n)$ with zero mean and variance σ_w^2 :

$$y(n) = x(n) + w(n) \quad (14.5.5)$$

The difference equation for (14.5.5) is an ARMA($2p, 2p$) process that can be expressed in matrix form:

$$\mathbf{Y}^t \mathbf{a} = \mathbf{W}^t \mathbf{a} \quad (14.5.7)$$

where:

- $\mathbf{Y}^t = [y(n) \ y(n-1) \ \dots \ y(n-2p)]$ is the observed data vector of size $2p+1$
- $\mathbf{W}^t = [w(n) \ w(n-1) \ \dots \ w(n-2p)]$ is the noise vector of size $2p+1$
- $\mathbf{a} = [1 \ a_1 \ \dots \ a_{2p}]$ is the coefficients vector

If we multiply (14.5.7) by \mathbf{Y} and take the expected value, we obtain the following:

$$(\mathbf{\Gamma}_{yy} - \sigma_w^2 \mathbf{I}) \mathbf{a} = \mathbf{0} \quad (14.5.9)$$

where:

- $\mathbf{\Gamma}_{yy}$ is the autocorrelation matrix
- σ_w^2 is an eigenvalue of the autocorrelation matrix
- \mathbf{a} is the eigenvector associated with the eigenvalue σ_w^2

Pisarenko: harmonic decomposition method

The Pisarenko method can be used to recover the sinusoidal frequencies of a corrupted signal $x(n)$ given two assumptions:

- The signal $x(n)$ consists of p sinusoids that has been corrupted by white noise.
- The autocorrelation matrix of size $(p+1) \times (p+1)$ is known or can be estimated from data

The Pisarenko method consists of the following steps:

Step 1: Compute the autocorrelation matrix \mathbf{R}_{xx}

Step 2: Find the eigenvector corresponding to the smallest minimum eigenvalue. The elements of this eigenvector is the parameters of the ARMA($2p, 2p$) model

Step 3: Find the frequencies $\{f_i\}$ of the sinusoids. This can be done by computing the roots of the polynomial $A(z)$ in Eq. (14.5.4) in the book. This polynomial has $2p$ poles on the unit circle which correspond to the frequencies of the system.

$$A(z) = 1 + \sum_{m=1}^{2p} a_m z^{-m} \quad (14.5.4)$$

Step 4: Solve Eq. (14.5.11) for the signal powers $\{P_i\}$

$$\begin{bmatrix} \cos 2\pi f_1 & \cos 2\pi f_2 & \cdots & \cos 2\pi f_p \\ \cos 4\pi f_1 & \cos 4\pi f_2 & \cdots & \cos 4\pi f_p \\ \vdots & \vdots & & \vdots \\ \cos 2\pi p f_1 & \cos 2\pi p f_2 & \cdots & \cos 2\pi p f_p \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_p \end{bmatrix} = \begin{bmatrix} \gamma_{yy}(1) \\ \gamma_{yy}(2) \\ \vdots \\ \gamma_{yy}(p) \end{bmatrix} \quad (14.5.11)$$

where

- $\gamma_{yy}(1), \gamma_{yy}(2), \dots, \gamma_{yy}(p)$ are the estimated autocorrelation values
- $P_i = \frac{A_i^2}{2}$ is the average power of the i th sinusoid and A_i is the corresponding amplitude

Step 5: Estimate the amplitude $A_i = \sqrt{2P_i}$

These steps are coded up in the `pisarenko()` function (see at end of this document).

Exam 2012, Problem 3: PSD Estimation assuming sinusoidal signal in white noise

For a given random process $\{x(n)\}$ the autocorrelation has been estimated and is given by

$ m $	$r_x(m)$
0	4
1	2
2	-1

```
clear variables;
```

1) Estimate PSD assuming sinusoidal white noise

Estimate the power density spectrum under the assumption that $\{x(n)\}$ consist of a single sinusoidal signal in additive white noise.

The Pisarenko method is used to estimate the power spectrum density of a random process.

The method makes two assumptions:

- The signal $x(n)$ consists of p sinusoids that has been corrupted by white noise.
- The autocorrelation matrix of size $(p + 1) \times (p + 1)$ is known or can be estimated

Given these assumptions, the Pisarenko method can recover the sinusoidal frequencies of the corrupted signal using the following steps:

Step 1: Compute the autocorrelation matrix \mathbf{R}_{yy}

Step 2: Find the eigenvector corresponding to the smallest minimum eigenvalue. The elements of this eigenvector is the parameters of the ARMA($2p, 2p$) model

Step 3: Find the frequencies $\{f_i\}$ of the sinusoids. This can be done by computing the roots of the polynomial $A(z)$ in (14.5.4). This polynomial has $2p$ poles on the unit circle which correspond to the frequencies of the system.

$$A(z) = 1 + \sum_{m=1}^{2p} a_m z^{-m} \quad (14.5.4)$$

Step 4: Solve Eq. (14.5.11) for the signal powers $\{P_i\}$

$$\begin{bmatrix} \cos 2\pi f_1 & \cos 2\pi f_2 & \cdots & \cos 2\pi f_p \\ \cos 4\pi f_1 & \cos 4\pi f_2 & \cdots & \cos 4\pi f_p \\ \vdots & \vdots & & \vdots \\ \cos 2\pi p f_1 & \cos 2\pi p f_2 & \cdots & \cos 2\pi p f_p \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_p \end{bmatrix} = \begin{bmatrix} \gamma_{yy}(1) \\ \gamma_{yy}(2) \\ \vdots \\ \gamma_{yy}(p) \end{bmatrix} \quad (14.5.11)$$

where

- $\gamma_{yy}(1), \gamma_{yy}(2), \dots, \gamma_{yy}(p)$ are the estimated autocorrelation values
- $P_i = \frac{A_i^2}{2}$ is the average power of the i th sinusoid and A_i is the corresponding amplitude

Step 5: Estimate the amplitude $A_i = \sqrt{2P_i}$

These steps are coded up in the `pisarenko()` function (see at end of this document):

```
r_xx = [4, 2, -1];
[F, A, P, lambda_min] = pisarenko(r_xx, 1)
```

```
F = 0.1490
A = 2.5970
P = 3.3723
```

```
lambda_min = 0.6277
```

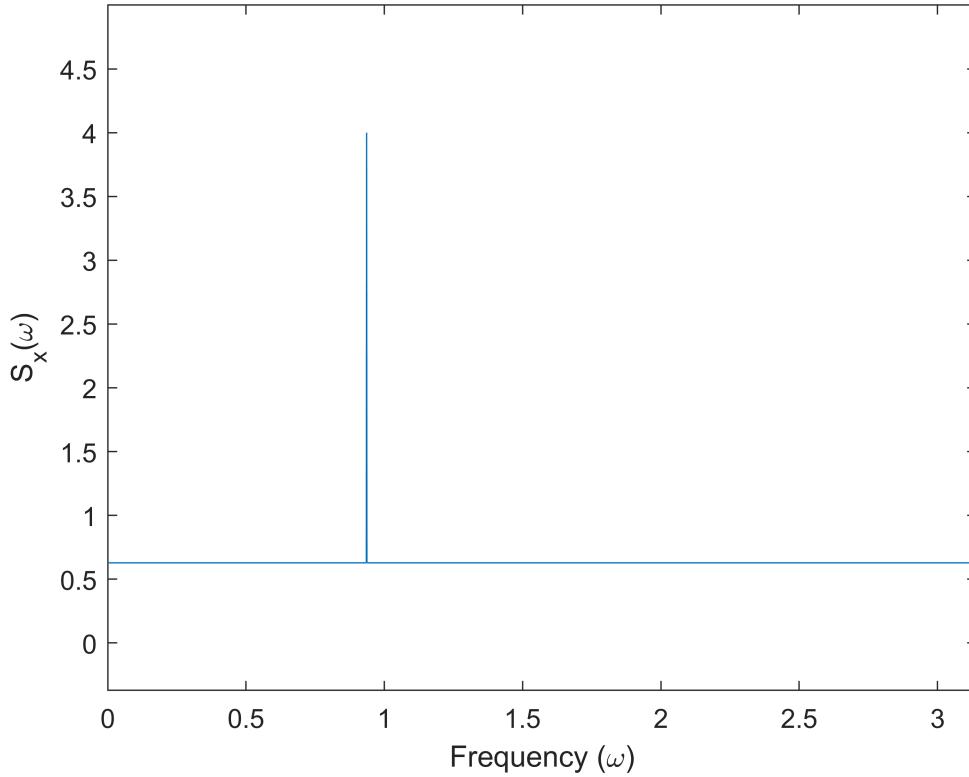
We can describe the signal as follows:

$$x(n) = 2.597 \cdot \cos(2\pi \cdot 0.149n + \phi) + w(n)$$

where $w(n)$ is white noise with variance $\sigma_w^2 = 0.6277$.

A sketch of the PSD is given as:

```
D = 3;
w = 0:10^-D:pi;
delta = @(n) round(n, D) == 0;
S = lambda_min*ones(1, numel(w)) + P*delta(w - 2*pi*F);
plot(w, S)
xlim([0, pi])
ylim([lambda_min-1, lambda_min+P+1])
xlabel('Frequency (\omega)')
ylabel('S_x(\omega)')
```



In ADSI Problem 4.4, we found that the autocorrelation of a real sinusoid given by $y(n) = A \cos(\omega n + \phi)$ where A and ω are real constants and ϕ is a random variable with $\phi \sim U(0, 2\pi)$ is:

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

The autocorrelation function of white noise with variance σ_w^2 is given by:

$$r_{ww}(\ell) = \sigma_w^2 \delta(\ell)$$

Using these results, we have the autocorrelation function of the signal:

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

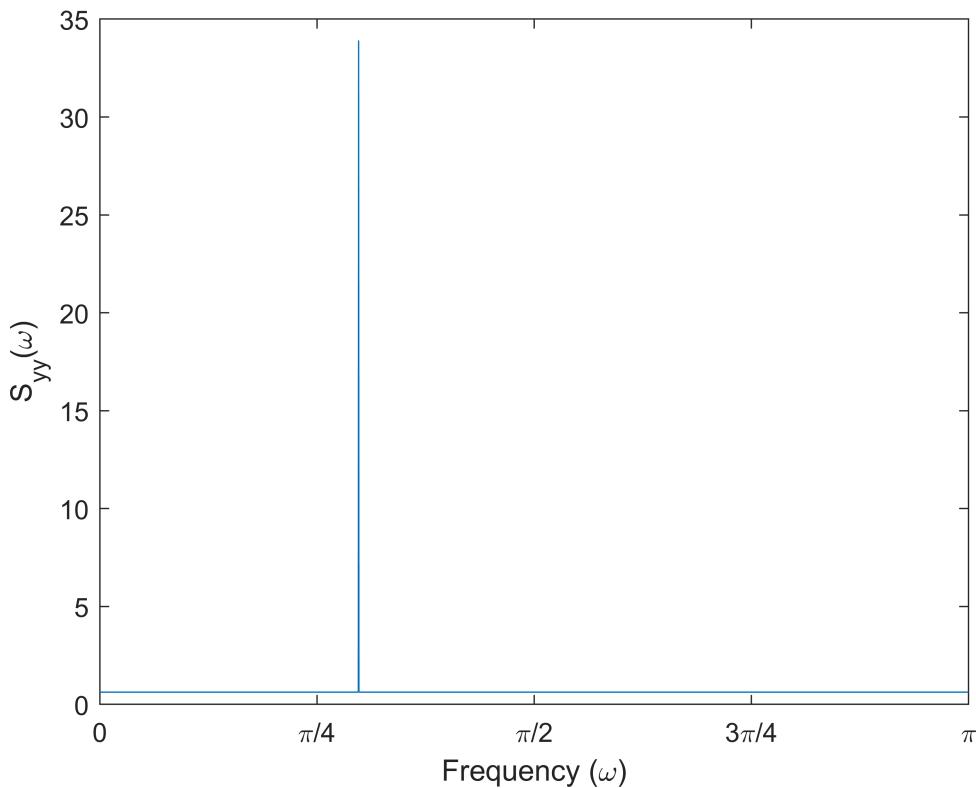
```
ell = 0:2;
(A^2/2) * cos(2*pi*F*ell) + lambda_min * (ell == 0)
```

```
ans = 1x3
4.0000    2.0000   -1.0000
```

The power spectral density is the Fourier transform the autocorrelation function:

$$S(\omega) = \frac{A^2}{2} \pi [\delta(\omega - \omega_0) + \delta(\omega + \omega_0)] + \sigma_w^2$$

```
D = 3;
w=0:10^-(D):pi;
delta = @(n) round(n, D) == 0;
S = (A^2*pi)/2 * pi * (delta(w-2*pi*F) + delta(w+2*pi*F)) + lambda_min;
plot(w, S);
xlabel('Frequency (\omega)');
ylabel('S_{yy}(\omega)');
set(gca, 'XTick', 0:pi/4:pi)
set(gca, 'XTickLabel', {'0', '\pi/4', '\pi/2', '3\pi/4', '\pi'})
xlim([0, pi])
```



Exam 2015 Problem 1: Estimate PSD of signal using Pisarenko

The autocorrelation function of an unknown signal is estimated as

$ l $	$r_x(l)$
0	17
1	4
2	5

```
clear variables;
```

1) Estimate and sketch the PSD of sinusoidal signal in white noise

1. Estimate and sketch the power spectral density of the signal under the assumption that the signal is a single sinusoidal signal in white noise.

The Pisarenko method can be used to recover the sinusoidal frequencies of a corrupted signal $x(n)$ given two assumptions:

- The signal $x(n)$ consists of p sinusoids that has been corrupted by white noise.

- The autocorrelation matrix of size $(p+1) \times (p+1)$ is known or can be estimated from data

The Pisarenko method consists of the following steps:

Step 1: Compute the autocorrelation matrix \mathbf{R}_{xx}

Step 2: Find the eigenvector corresponding to the smallest minimum eigenvalue. The elements of this eigenvector is the parameters of the ARMA($2p, 2p$) model

Step 3: Find the frequencies $\{f_i\}$ of the sinusoids. This can be done by computing the roots of the polynomial $A(z)$ in Eq. (14.5.4) in the book. This polynomial has $2p$ poles on the unit circle which correspond to the frequencies of the system.

$$A(z) = 1 + \sum_{m=1}^{2p} a_m z^{-m} \quad (14.5.4)$$

Step 4: Solve Eq. (14.5.11) for the signal powers $\{P_i\}$

$$\begin{bmatrix} \cos 2\pi f_1 & \cos 2\pi f_2 & \cdots & \cos 2\pi f_p \\ \cos 4\pi f_1 & \cos 4\pi f_2 & \cdots & \cos 4\pi f_p \\ \vdots & \vdots & & \vdots \\ \cos 2\pi p f_1 & \cos 2\pi p f_2 & \cdots & \cos 2\pi p f_p \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_p \end{bmatrix} = \begin{bmatrix} \gamma_{yy}(1) \\ \gamma_{yy}(2) \\ \vdots \\ \gamma_{yy}(p) \end{bmatrix} \quad (14.5.11)$$

where

- $\gamma_{yy}(1), \gamma_{yy}(2), \dots, \gamma_{yy}(p)$ are the estimated autocorrelation values
- $P_i = \frac{A_i^2}{2}$ is the average power of the i th sinusoid and A_i is the corresponding amplitude

Step 5: Estimate the amplitude $A_i = \sqrt{2P_i}$

These steps are coded up in the `pisarenko()` function (see at end of this document).

```
r_xx = [17, 4, 3]; % In the solution r_x(2)=3
[F, A, P, lambda_min] = pisarenko(r_xx, 1)
```

```
F = 0.0645
A = 2.9504
P = 4.3523
lambda_min = 12.6477
```

We can describe the signal as:

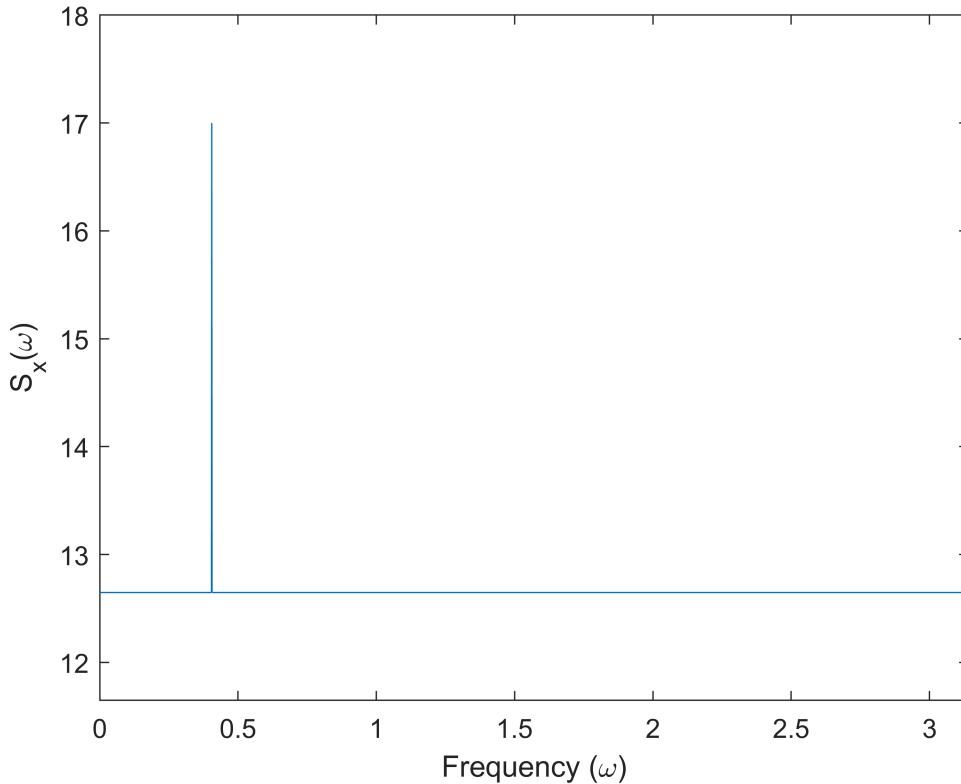
$$x(n) = 2.9504 \cos(2\pi \cdot 0.0645 + \phi) + w(n)$$

where $w(n)$ is white noise with variance $\sigma_w^2 = 12.6477$

A sketch of the PSD is given as:

```
D = 3;
```

```
w = 0:10^-D:pi;
delta = @(n) round(n, D) == 0;
S = lambda_min*ones(1, numel(w)) + P*delta(w - 2*pi*F);
plot(w, S)
xlim([0, pi])
ylim([lambda_min-1, lambda_min+P+1])
xlabel('Frequency (\omega)')
ylabel('S_x(\omega)')
```



To compute the power spectral density, we need to find the autocorrelation function of the sinusoid.

In ADSI Problem 4.4, we found that the autocorrelation of a real sinusoid given by $y(n) = A \cos(\omega n + \phi)$ where A and ω are real constants and ϕ is a random variable with $\phi \sim U(0, 2\pi)$ is:

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

The autocorrelation function of white noise with variance σ_w^2 is given by:

$$r_{ww}(\ell) = \sigma_w^2 \delta(\ell)$$

Combining these results, the general autocorrelation function of our signal is:

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

The power spectral density is the Fourier transform the autocorrelation function which is given by:

$$S(\omega) = \frac{A^2}{2} \pi [\delta(\omega - \omega_0) + \delta(\omega + \omega_0)] + \sigma_w^2$$

A, F, lambda_min

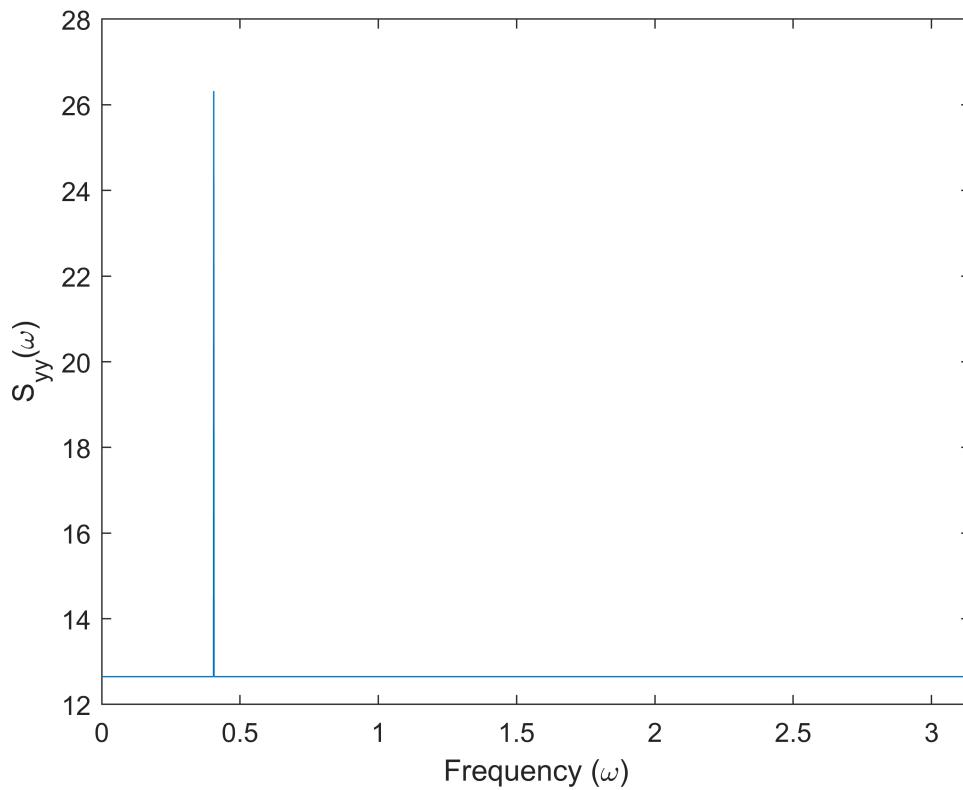
```
A = 2.9504
F = 0.0645
lambda_min = 12.6477
```

Using the values found via the Pisarenko, the PSD is:

$$S(\omega) = \frac{(2.9504)^2}{2} \pi [\delta(\omega - 0.0645 \cdot 2\pi) + \delta(\omega + 0.0645 \cdot 2\pi)] + 12.6477$$

We can plot the PSD:

```
D = 3;
w = 0:10^(-D):pi;
delta = @(n) round(n, D) == 0;
S = (A^2/2)*pi * (delta(w-(F*2*pi)) + delta(w+(F*2*pi))) + lambda_min;
%plot(w/pi, real(pow2db(S)))
plot(w, S);
xlabel('Frequency (\omega)');
ylabel('S_{yy}(\omega)');
%set(gca,'XTick',0:pi/4:pi)
%set(gca,'XTickLabel',{'0', '\pi/4', '\pi/2', '3\pi/4', '\pi'})
xlim([0, pi])
```



2) Calculate the signal to noise ratio

2. Calculate the signal to noise ratio.

The signal to noise ratio is the power of the signal divided by the power of the noise:

$$\text{SNR} = \frac{\text{power of signal}}{\text{power of noise}} = \frac{P}{\sigma_w^2} = 0.3441$$

```
SNR = P/lambda_min
```

```
SNR = 0.3441
```

3) Does an additional value to the ACRS agree with the signal model?

Assume that an additional value of the autocorrelation function has been estimated and $r_x(3) = 20$.

3. Is this additional value in agreement with the signal model determined in the above questions?

The additional autocorrelation value is not in agreement with the signal model.

A fundamental property of the autocorrelation function is that $r_x(0) \geq r_x(\ell)$ for all ℓ .

This property is violated with $r_x(0) = 17$ and $r_x(3) = 20$.

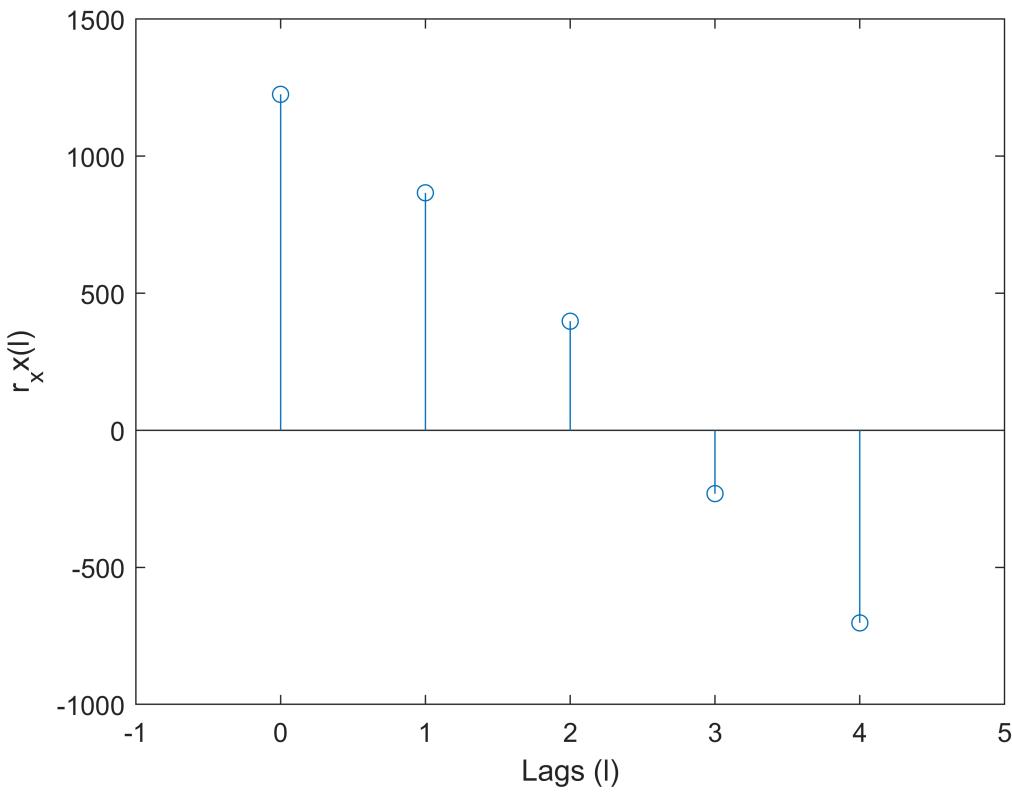
Exam 2018 Problem 1: Compute PSD from data using Pisarenko (sinusoid with additive white noise)

Here's a sequence of data from a measurement.

$$\{-11, 21, 18, 62, 34, -5, -14, -64, -49, -35, -1, 29, 37, 49, 32, 12\}$$

[✓] 1) Compute and plot the autocorrelation for lags 0 to 4

```
x = [-11, 21, 18, 62, 34, -5, -14, -64, -49, -35, -1, 29, 37, 49, 32, 12];  
  
% Estimate autocorrelation using data  
[r_xx, ell] = xcorr(x, 'biased');  
  
% Print out the lags 0 to 4  
mid = floor(numel(ell)/2)+1;  
r_xx(mid:mid+4)  
  
ans = 1x5  
103 ×  
1.2256 0.8664 0.3979 -0.2310 -0.7027  
  
% Plot the results  
stem(0:4, r_xx(mid:mid+4))  
xlim([-1, 5])  
xlabel('Lags (1)')  
ylabel('r_xx(1)')
```



[✓] 2) Compute the power spectral density assuming a Pisarenko model.

Assume that the signal is a sinusoid in additive white Gaussian noise.

The Pisarenko method can be used to recover the sinusoidal frequencies of a corrupted signal $x(n)$ given two assumptions:

- The signal $x(n)$ consists of p sinusoids that has been corrupted by white noise.
- The autocorrelation matrix of size $(p + 1) \times (p + 1)$ is known or can be estimated from data

The Pisarenko method consists of the following steps:

Step 1: Compute the autocorrelation matrix \mathbf{R}_{xx}

Step 2: Find the eigenvector corresponding to the smallest minimum eigenvalue. The elements of this eigenvector is the parameters of the ARMA($2p, 2p$) model

Step 3: Find the frequencies $\{f_i\}$ of the sinusoids. This can be done by computing the roots of the polynomial $A(z)$ in Eq. (14.5.4) in the book. This polynomial has $2p$ poles on the unit circle which correspond to the frequencies of the system.

$$A(z) = 1 + \sum_{m=1}^{2p} a_m z^{-m} \quad (14.5.4)$$

Step 4: Solve Eq. (14.5.11) for the signal powers $\{P_i\}$

$$\begin{bmatrix} \cos 2\pi f_1 & \cos 2\pi f_2 & \cdots & \cos 2\pi f_p \\ \cos 4\pi f_1 & \cos 4\pi f_2 & \cdots & \cos 4\pi f_p \\ \vdots & \vdots & & \vdots \\ \cos 2\pi p f_1 & \cos 2\pi p f_2 & \cdots & \cos 2\pi p f_p \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_p \end{bmatrix} = \begin{bmatrix} \gamma_{yy}(1) \\ \gamma_{yy}(2) \\ \vdots \\ \gamma_{yy}(p) \end{bmatrix} \quad (14.5.11)$$

where

- $\gamma_{yy}(1), \gamma_{yy}(2), \dots, \gamma_{yy}(p)$ are the estimated autocorrelation values
- $P_i = \frac{A_i^2}{2}$ is the average power of the i th sinusoid and A_i is the corresponding amplitude

Step 5: Estimate the amplitude $A_i = \sqrt{2P_i}$

These steps are coded up in the `pisarenko()` function (see at end of this document):

```
[F, A, P, lambda_min] = pisarenko(r_xx(mid:mid+4), 1)
```

```
F = 0.0938
A = 45.6585
P = 1.0423e+03
lambda_min = 183.2154
```

We can describe the signal as:

$$x(n) = 45.6585 \cos(2\pi \cdot 0.0938n) + w(n)$$

where $w(n)$ is white noise with variance $\sigma_w^2 = 183.2154$

To compute the power spectral density, we need to find the autocorrelation function of the sinusoid.

In ADSI Problem 4.4, we found that the autocorrelation of a real sinusoid given by $y(n) = A \cos(\omega n + \phi)$ where A and ω are real constants and ϕ is a random variable with $\phi \sim U(0, 2\pi)$ is:

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

The autocorrelation function of white noise with variance σ_w^2 is given by:

$$r_{ww}(\ell) = \sigma_w^2 \delta(\ell)$$

Combining these results, the general autocorrelation function of our signal is:

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

The power spectral density is the Fourier transform the autocorrelation function which is given by:

$$S(\omega) = \frac{A^2}{2} \pi [\delta(\omega - \omega_0) + \delta(\omega + \omega_0)] + \sigma_w^2$$

A, F, lambda_min

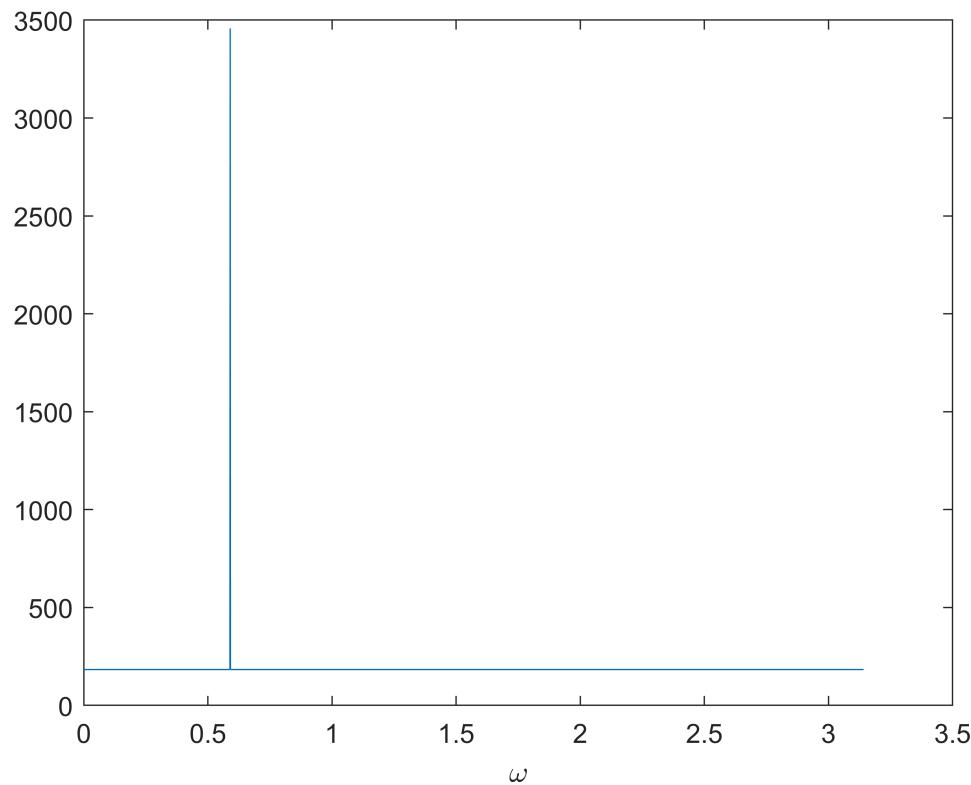
```
A = 45.6585
F = 0.0938
lambda_min = 183.2154
```

Using the values found via the Pisarenko, the PSD is:

$$S(\omega) = \frac{(45.6585)^2}{2} \pi [\delta(\omega - 2\pi \cdot 0.0938) + \delta(\omega + 2\pi \cdot 0.0938)] + 183.2154$$

Plot the PSD:

```
D = 3;
w = 0:10^(-D):pi;
delta = @(n) round(n, D) == 0;
S = (A^2/2)*pi * (delta(w-(F*2*pi)) + delta(w+(F*2*pi))) + lambda_min;
%plot(w/pi, real(pow2db(S)))
plot(w, S);
xlabel('\omega')
```



[✓] 3. Discuss whether the Pisarenko model can be considered appropriate for the given data.

We can describe the signal as:

$$x(n) = 45.6585 \cos(2\pi \cdot 0.0938n) + w(n)$$

where $w(n)$ is white noise with variance $\sigma_w^2 = 183.2154$

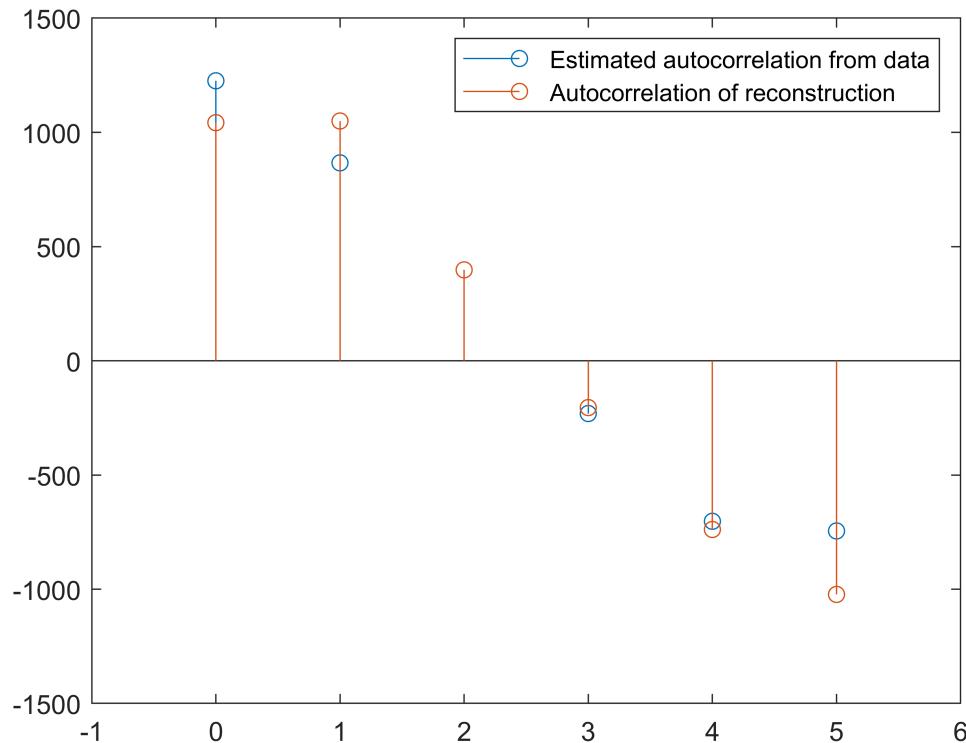
The general autocorrelation function of that signal is:

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

As only 16 samples are available, the estimate of the autocorrelation is quite uncertain and this uncertainty will enter into the Pisarenko model.

The plot below of the autocorrelation does resemble a slow oscillation and as such the Pisarenko model can't be ruled out on the basis of the available data.

```
% Plot the autocorrelation function of the sinusoid
ell = 0:5;
r_ww = lambda_min * (ell == 1);
r_xx_p = (A^2/2) * cos(2*pi*F*ell) + r_ww;
stem(ell, r_xx(mid:mid+max(ell)))
hold on;
stem(ell, r_xx_p)
legend('Estimated autocorrelation from data', 'Autocorrelation of reconstruction')
xlim([min(ell)-1, max(ell)+1])
hold off;
```



In addition, plotting the limited amount of samples, we see a sinusoidal shape. Since Pisarenko model assumes sinusoidal signal that has been corrupted by white noise then it can be appropriate model.

Problem 5.1: Frequency estimation using Pisarenko's method

Find the frequency and amplitude of a single real sinusoidal signal in white noise, $y(n) = A \cos(2\pi f n + \phi) + w(n)$ when the first few values of the autocorrelation function are given by

$$r_y(l) = \begin{cases} 3, & l = 0 \\ 0, & l = \pm 1 \\ -2, & l = \pm 2 \end{cases}$$

```
clear variables;
```

Step 1: Compute the autocorrelation matrix \mathbf{R}_{yy}

```
p = 1;
r_yy = [3, 0, -2];
R_yy = toeplitz(r_yy)
```

```
R_yy = 3x3
      3     0    -2
      0     3     0
     -2     0     3
```

Step 2: Find the minimum eigenvalue and the corresponding eigenvector. The elements of this eigenvector is the parameters of the ARMA($2p, 2p$) model.

```
[eigvecs, eigvals] = eigs(R_yy, size(R_yy, 1), 'smallestreal');
min_eig_vec = eigvecs(:, 1)
```

```
min_eig_vec = 3x1
-0.7071
      0
-0.7071
```

Step 3: Find the frequencies $\{f_i\}$ of the sinusoids. This can be done by computing the roots of the polynomial $A(z)$ in (14.5.4). This polynomial has $2p$ poles on the unit circle which correspond to the frequencies of the system.

$$A(z) = 1 + \sum_{m=1}^{2p} a_m z^{-m} \quad (14.5.4)$$

Since the given estimating the frequency of a signal real sinusoid, we have:

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2}$$

The parameters $\{a_0, a_1, a_2\}$ corresponds to the eigenvector of the smallest eigenvalue:

```
% By definition a_0 = 1 because we want A(z)=1+....  
a = min_eig_vec / min_eig_vec(1)
```

```
a = 3x1  
1  
0  
1
```

This means we have following IIR filter:

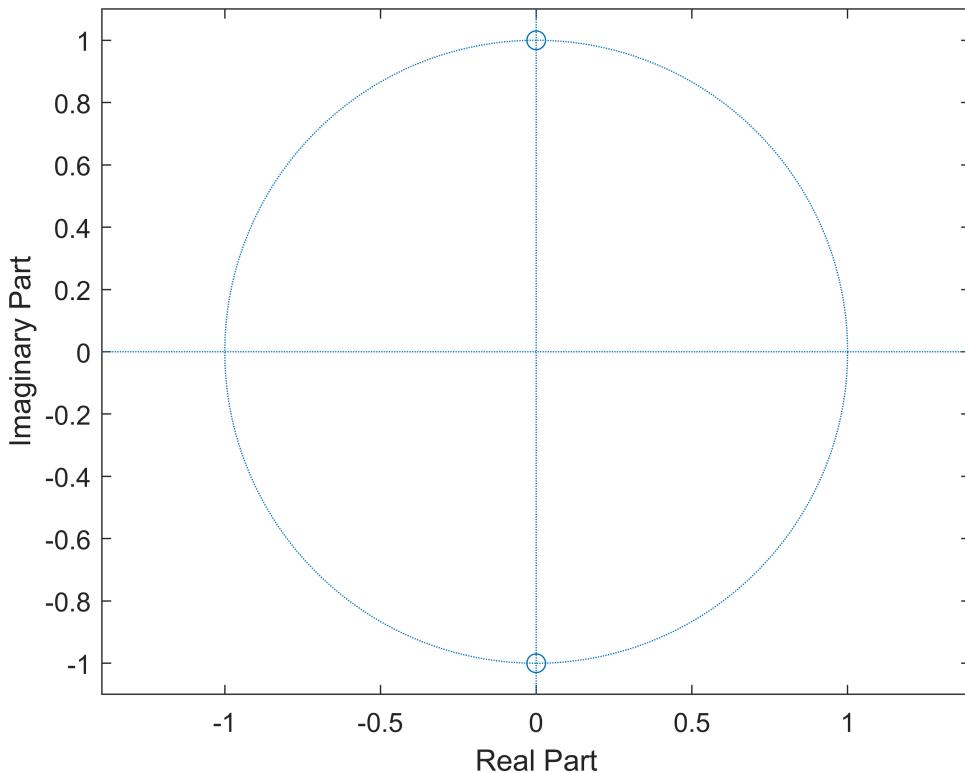
$$A(z) = 1 + z^{-2}$$

We know that the polynomial $A(z)$ in (14.5.4) has $2p = 2 \cdot 1 = 2$ poles on the unit circle. We can find the poles by finding the roots of the system $A(z) = 1 + z^{-2}$:

```
z = roots(a)
```

```
z = 2x1 complex  
0.0000 + 1.0000i  
0.0000 - 1.0000i
```

```
zplane(z)
```



The poles on the unit circle correspond to the frequencies of the system.

The two poles are complex conjugate of each other in order to create a real sinusoid. If we only had one pole then it would be complex.

Since $z = A e^{j2\pi f}$, we can find the frequency by the angle:

$$\angle z = 2\pi f \text{ and } f = \frac{\angle z}{2\pi}$$

```
% The poles come in pairs. Each pair is complex conjugate
% of one another. Only use one of them and find the absolute value.
f1 = abs(angle(z(2))) / (2*pi)
```

f1 = 0.2500

```
f2 = angle(z(1)) / (2*pi)
```

f2 = 0.2500

Step 4: Solve Eq. (14.5.11) to find the signal powers $\{P_i\}$

$$\begin{bmatrix} \cos 2\pi f_1 & \cos 2\pi f_2 & \cdots & \cos 2\pi f_p \\ \cos 4\pi f_1 & \cos 4\pi f_2 & \cdots & \cos 4\pi f_p \\ \vdots & \vdots & & \vdots \\ \cos 2\pi p f_1 & \cos 2\pi p f_2 & \cdots & \cos 2\pi p f_p \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_p \end{bmatrix} = \begin{bmatrix} \gamma_{yy}(1) \\ \gamma_{yy}(2) \\ \vdots \\ \gamma_{yy}(p) \end{bmatrix} \quad (14.5.11)$$

Since we only have one frequency, the equation becomes:

$$\cos(2\pi f_1)P_1 = \gamma_{yy}(1) \text{ where } \gamma_{yy}(1) \text{ is the second autocorrelation value.}$$

We want to find P_1 :

$$P_1 = \frac{\gamma_{yy}(1)}{\cos(2\pi f_1)}$$

```
P1 = r_yy(2) / cos(2*pi*f1)
```

P1 = 0

This does not work because the autocorrelation value $\gamma_{yy}(1) = 0$. Instead we need to go one lag higher.

$$\cos(2\pi f_1)P_2 = \gamma_{yy}(2) \text{ where } \gamma_{yy}(2) \text{ is the third autocorrelation value.}$$

$$P_2 = \frac{\gamma_{yy}(2)}{\cos(4\pi f_1)}$$

```
P2 = r_yy(3) / cos(4*pi*f1)
```

P2 = 2

From section 14.5.1, we are given:

$$P_i = \frac{A_i^2}{2}$$

This allows us to estimate the amplitude A_i given P_i :

$$A_i = \sqrt{2P_i}$$

```
A1 = sqrt(2*P2)
```

```
A1 = 2
```

So frequency is $f = 0.25$ and the amplitude is $A = 2$.

Problem 5.2: Pisarenko of two sinusoids (sine) in white noise

Let a signal be given by

$$x(n) = A_1 \sin(\omega_1 n + \phi_1) + A_2 \sin(\omega_2 n + \phi_2) + w(n)$$

Where the phases ϕ_1 and ϕ_2 are uncorrelated and uniformly distributed from 0 to 2π , $w(n)$ is zero mean gaussian white noise.

```
clear variables;
```

0) Preliminary, compute the autocorrelation function of a sine signal:

In ADSI Problem 4.4, we found that the autocorrelation of a complex sinusoid givey

by $y(n) = A e^{j(\omega n + \phi)}$ where A and ω are real constants and ϕ is a random variable with $\phi \sim U(0, 2\pi)$ is:

$$r_{yy}(\ell) = A^2 e^{j\omega\ell}$$

To use this result, we need to convert the given signal in this problem to complex exponential signal.

A complex exponential signal is always formed by the sum of two real signals:

$$Ae^{j\omega n} = A \cos(\omega n) + j A \sin(\omega n)$$

Therefore, we know that:

$$\sin(\omega) = \frac{1}{2j} e^{j\omega} - \frac{1}{2j} e^{-j\omega}$$

Using this relation, we can rewrite a real signal $A \sin(\omega n + \phi)$ as:

$$z(n) = A \sin(\omega n + \phi)$$

$$z(n) = \frac{A}{2j} e^{j(\omega n + \phi)} - \frac{A}{2j} e^{-j(\omega n + \phi)}$$

To compute the autocorrelation function, we square the magnitude, remove the phase and replace n with ℓ :

$$r_{zz}(\ell) = \left(\frac{A}{2j} \right)^2 e^{j\omega\ell} - \left(\frac{A}{2j} \right)^2 e^{-j\omega\ell}$$

We know that $(2j)^2 = 2^2 \cdot j^2 = -4$ because $j = \sqrt{-1}$ so $j^2 = -1$

$$r_{zz}(\ell) = \frac{A^2}{-4} e^{j\omega\ell} - \frac{A^2}{-4} e^{-j\omega\ell}$$

$$r_{zz}(\ell) = -\frac{A^2}{4} e^{j\omega\ell} + \frac{A^2}{4} e^{-j\omega\ell}$$

We want to make the autocorrelation function in terms of $\cos(\cdot)$, we rewrite the expression as follows:

$$r_{zz}(\ell) = \left(-\frac{A^2}{2} \right) \cdot \frac{1}{2} (e^{j\omega\ell} + e^{-j\omega\ell})$$

Since $\cos(\theta) = \frac{1}{2}(e^{j\theta} + e^{-j\theta})$, we can rewrite the expression as:

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

Thus, the autocorrelation function of a real signal $z(n) = A \sin(\omega n + \phi)$ is

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

1) Calculate the autocorrelation function

The autocorrelation function of a real signal $z(n) = A \sin(\omega n + \phi)$ where A and ω are real constants and ϕ is a random variable with $\phi \sim U(0, 2\pi)$ is:

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

The autocorrelation function of white noise Gaussian noise with zero mean and variance σ_w^2 is:

$$r_{ww}(\ell) = \sigma_w^2 \delta(\ell)$$

In this problem, we are given:

Let a signal be given by

$$x(n) = A_1 \sin(\omega_1 n + \phi_1) + A_2 \sin(\omega_2 n + \phi_2) + w(n)$$

Where the phases ϕ_1 and ϕ_2 are uncorrelated and uniformly distributed from 0 to 2π , $w(n)$ is zero mean gaussian white noise.

The autocorrelation function of the signal $x(n)$ given in this problem is:

$$r_{xx}(\ell) = -\frac{A_1^2}{2} \cos(\omega_1 \ell) - \frac{A_2^2}{2} \cos(\omega_2 \ell) + \sigma_w^2 \delta(\ell)$$

or

$$r_{xx}(\ell) = -\frac{A_1^2}{2} \cos(2\pi f_1 \ell) - \frac{A_2^2}{2} \cos(2\pi f_2 \ell) + \sigma_w^2 \delta(\ell)$$

2) Choose appropriate values for the amplitudes and frequencies and for the noise power

We choose following values:

$$A_1 = 4, A_2 = 8, f_1 = \frac{1}{4\pi}, f_2 = \frac{1}{\pi}, \sigma_w^2 = 1$$

3) Calculate the eigenvalues of the autocorrelation matrix as a function of its size and compare with your expectation

```
% The values that we chose in problem 5.2
A1=4; A2=8; f1=1/(4*pi); f2=1/pi; wvar=1;

% The size of the autocorrelation sequence
% L = 64;
% l = 0:L-1;
% r_xx = -((A1^2/2) * cos(2*pi*f1*l)) - ((A2^2/2) * cos(2*pi*f2*l)) + wvar
ell_seq = 0:pi/4:2*pi;

L = numel(ell_seq);

% Initialise the autocorrelation sequence with zeros.
% This becomes very important especially when
% the size of the ACRZ (the ell variable) is changed
r_xx = zeros(L, 1);

% Compute the autocorrelation sequence based on the
% autocorrelation function derived in problem 1)
for i = 1:L
    ell = ell_seq(i);
    r_xx(i) = -((A1^2)/2 * cos(2*pi*f1*ell)) - ((A2^2)/2 * cos(2*pi*f2*ell)) + wvar;
end
r_xx
```

```
r_xx = 9x1
-39.0000
-6.3910
27.3431
-2.0615
-31.0000
4.0615
38.6569
8.3910
-23.0000
```

```
% Compute the autocorrelation matrix
R_xx = toeplitz(r_xx)
```

```
R_xx = 9x9
-39.0000 -6.3910 27.3431 -2.0615 -31.0000 4.0615 38.6569 8.3910 ...
-6.3910 -39.0000 -6.3910 27.3431 -2.0615 -31.0000 4.0615 38.6569
27.3431 -6.3910 -39.0000 -6.3910 27.3431 -2.0615 -31.0000 4.0615
-2.0615 27.3431 -6.3910 -39.0000 -6.3910 27.3431 -2.0615 -31.0000
-31.0000 -2.0615 27.3431 -6.3910 -39.0000 -6.3910 27.3431 -2.0615
4.0615 -31.0000 -2.0615 27.3431 -6.3910 -39.0000 -6.3910 27.3431
38.6569 4.0615 -31.0000 -2.0615 27.3431 -6.3910 -39.0000 -6.3910
8.3910 38.6569 4.0615 -31.0000 -2.0615 27.3431 -6.3910 -39.0000
-23.0000 8.3910 38.6569 4.0615 -31.0000 -2.0615 27.3431 -6.3910
```

4) Use the Pisarenko method to calculate the spectrum and compare with the expected results

```
[F, A] = pisarenko(r_xx, 2)
```

```
F = 2x1
0.0878
0.4166
A = 2x1
7.0875
8.0111
```

Problem 5.3: Frequency resolution of the Pisarenko method

In this problem the goal is to investigate the frequency resolution of the Pisarenko method. We will use your two-sinusoids-in-white-noise model from the previous problem:

Let a signal be given by

$$x(n) = A_1 \sin(\omega_1 n + \phi_1) + A_2 \sin(\omega_2 n + \phi_2) + w(n)$$

Where the phases ϕ_1 and ϕ_2 are uncorrelated and uniformly distributed from 0 to 2π , $w(n)$ is zero mean gaussian white noise.

```
clear variables;
```

1) The stability of PSD estimates using the Pisarenko method

a) Create a number of short realizations, i.e. $N=64$ or $N=128$ with different initial phases.

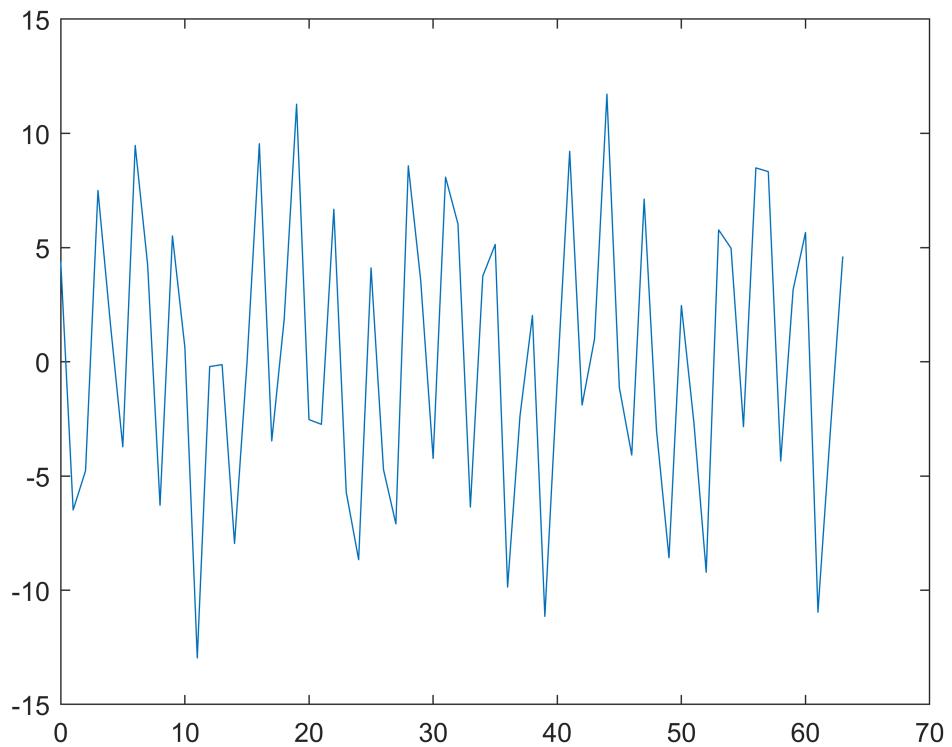
```
% The values that we chose in problem 5.2
A1=4; A2=8; f1=1/(4*pi); f2=1/pi; wvar=1;

N = 64;
n = 0:N-1;

% Generate phi(1) and phi(2) from U(0, 2*pi)
phi = 2*pi*rand(1, 2);

% Generate noise signal with chosen variance
wn = sqrt(wvar) * randn(1, N);

% Generate the signal
x1 = A1 * sin(2*pi*f1*n + phi(1)) + A2 * sin(2*pi*f2*n + phi(2)) + wn;
plot(n, x1)
```



```

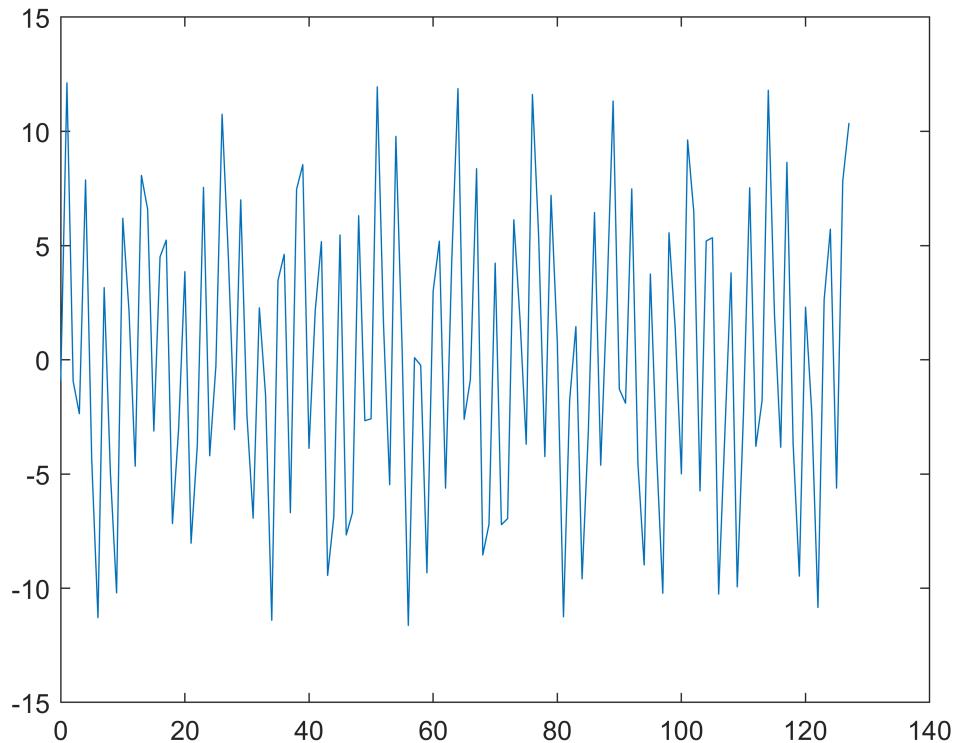
N = 128;
n = 0:N-1;

% Generate phi(1) and phi(2) from U(0, 2*pi)
phi = 2*pi*rand(1, 2);

% Generate noise signal with chosen variance
wn = sqrt(wvar) * randn(1, N);

% Generate the signal
x2 = A1 * sin(2*pi*f1*n + phi(1)) + A2 * sin(2*pi*f2*n + phi(2)) + wn;
plot(n, x2)

```



b) Use the Pisarenko method to compute the spectrum of the two realizations.

```
p = 2; % Number of sinusoids in the signals

% Compute the autocorrelation sequence
[r_x1, lags] = xcorr(x1', 'biased');

% Compute the autocorrelation matrix from rx(0)
mid_index = ceil(numel(lags)/2);
[F, A] = pisarenko(r_x1(mid_index:end), p)
```

```
F = 2x1
0.3168
0.0842
A = 2x1
7.6815
3.6995
```

```
% Compute the autocorrelation sequence
[r_x2, lags] = xcorr(x2', 'biased');

% Compute the autocorrelation matrix from rx(0)
mid_index = ceil(numel(lags)/2);
[F, A] = pisarenko(r_x2(mid_index:end), p)
```

```
F = 2x1
0.3182
0.0800
```

```
A = 2x1
    7.8739
    3.9904
```

```
f1, f2, A1, A2
```

```
f1 = 0.0796
f2 = 0.3183
A1 = 4
A2 = 8
```

c) Comment on the stability of the Pisarenko method

The Pisarenko method seem to be stable.

Problem 5.4: Pisarenko and coloured noise

In the derivation of the harmonic methods it was assumed that the signal consisted of a sinusoids or complex exponentials in white noise. In real life, the assumption of white noise is likely to be wrong in a lot of situations.

```
clear variables;
```

1) Create a MATLAB model of a sinusoidal signal in white, slightly coloured and very coloured noise.

When the power spectral density of the noise is not uniform across the entire frequency spectrum, it is called colored noise.

```
% The values that we chose in problem 5.2
A1=4;      f1=1/8;    phi1 = 2*pi*rand(); % Phase ~ U(0, 2pi)

N = 128;    n = 0:N-1;

% Generate a signal with white noise
w1 = randn(1, N);
x1 = A1 * cos(2*pi*f1*n + phi1) + w1;

% Generate a signal with slightly coloured noise
w2 = step(dsp.ColoredNoise('InverseFrequencyPower', 0.4, 'SamplesPerFrame', numel(n)));
x2 = A1 * cos(2*pi*f1*n + phi1) + w2;

% Generate a signal with highly coloured noise
w3 = step(dsp.ColoredNoise('InverseFrequencyPower', 2, 'SamplesPerFrame', numel(n)));
x3 = A1 * cos(2*pi*f1*n + phi1) + w3;
```

2) Compare the eigenvalues of the autocorrelation matrix for the three different scenarios.

```
L = 3; % The length of ACRS
[r_x1, ~] = xcorr(x1, L, 'biased');
R_x1 = toeplitz(r_x1(L+1:end));
eigs(R_x1, size(R_x1, 1), 'smallestreal')
```

```
ans = 4x1
0.7408
0.8408
16.5341
16.7258
```

```
[r_x2, ~] = xcorr(x2, L, 'biased');
R_x2 = toeplitz(r_x2(L+1:end));
eigs(R_x2, size(R_x2, 1), 'smallestreal')
```

```
ans = 4x1
0.8768
1.1592
16.4413
17.1596
```

```
[r_x3, ~] = xcorr(x3, L, 'biased');
R_x3 = toeplitz(r_x3(L+1:end));
eigs(R_x3, size(R_x3, 1), 'smallestreal')
```

```
ans = 4x1
0.2948
2.6214
17.4658
243.7402
```

3) Calculate the Pisarenko spectra and discuss whether Pisarenko is useful when the noise is coloured.

```
[F1, A1] = pisarenko(r_x1(L+1:end), 1)
```

```
F1 = 0.1247
A1 = 3.9887
```

```
[F2, A2] = pisarenko(r_x2(L+1:end), 1)
```

```
F2 = 0.1235
A2 = 3.9976
```

```
[F3, A3] = pisarenko(r_x3(L+1:end), 1)
```

```
F3 = 0.0401
A3 = 11.4277
```

It seems that Pisarenko yields relatively good results when the noise is slightly coloured. However, the results degrade for the noise highly coloured.

Problem 5.5: Pisarenko, wrong choice of eigenvector

In example 14.5.1 from the note the autocorrelation matrix for a process consisting of a single sinusoid in additive white noise is given as

$$\mathbf{R}_y = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix}$$

The example proceeds to find $f_1 = \frac{1}{8}$

1) How does the use of a wrong eigenvalue influence the solution?

Repeat the example, but use the two other eigenvalues as starting points. How does the use of a wrong eigenvalue influence the solution?

Step 1: Compute the autocorrelation matrix \mathbf{R}_{yy}

```
r_yy = [3, 1, 0];
R_yy = toeplitz(r_yy);
[eigvecs, eigvals] = eigs(R_yy, size(R_yy, 1), 'smallestreal');
```

```
% Choose which eigenvalues is used for the calculations.
% The eigenvalues are sorted in increasing order, so
% the first eigenvalue is the smallest.
chosen_eig_idx = 1;
```

Step 2: Find the minimum eigenvalue and the corresponding eigenvector. The elements of this eigenvector is the parameters of the ARMA(2p, 2p) model

```
vals = diag(eigvals);
eig_val_min = vals(chosen_eig_idx)
```

```
eig_val_min = 1.5858
```

Using the smallest eigenvalue, we should get:

$$\sigma_w^2 = \lambda_{\min} = 3 - \sqrt{2} = 1.5858$$

```
% Get the eigenvector corresponding to the minimum eigenvalue
eig_vec_min = eigvecs(:,chosen_eig_idx);

% Ensure that a_0 = 1 (this is by definition)
eig_vec_min = eig_vec_min / eig_vec_min(1)
```

```
eig_vec_min = 3x1
1.0000
-1.4142
1.0000
```

If the smallest eigenvalue is used we would get $a_0 = 1$, $a_1 = -1.4142$, $a_2 = 1$

Step 3: Find the frequencies $\{f_i\}$ of the sinusoids. This can be done by computing the roots of the polynomial $A(z)$ in (14.5.4). This polynomial has $2p$ poles on the unit circle which correspond to the frequencies of the system.

$$A(z) = 1 + \sum_{m=1}^{2p} a_m z^{-m} \quad (14.5.4)$$

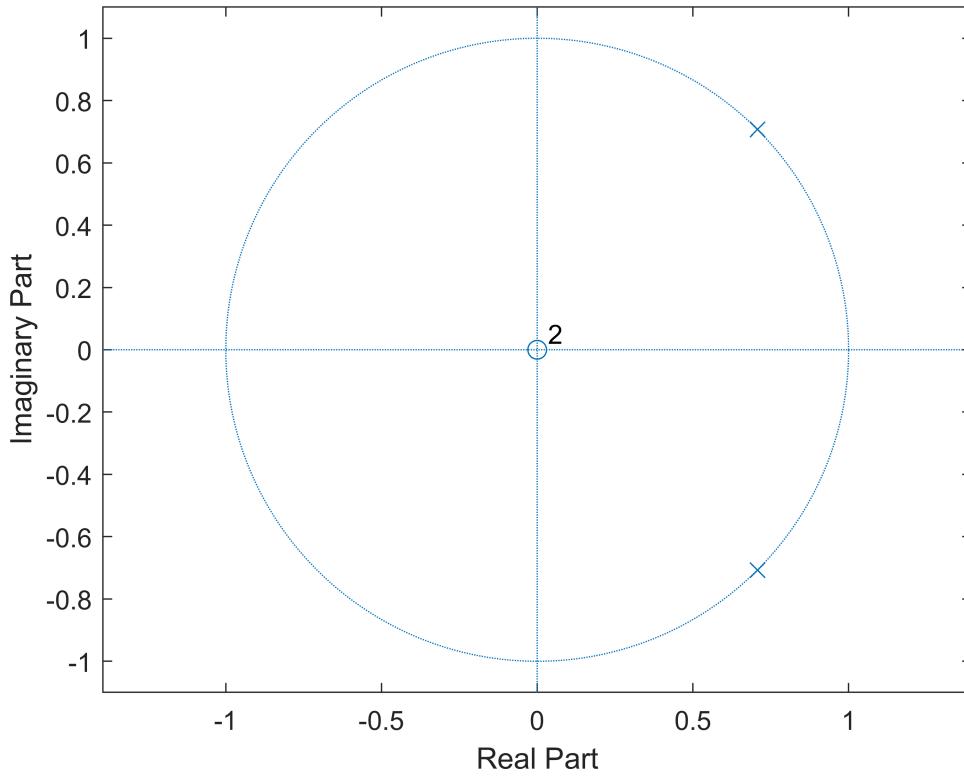
```
z = roots(eig_vec_min)
```

```
z = 2x1 complex
0.7071 + 0.7071i
0.7071 - 0.7071i
```

So we have $z_1 = 0.7071 + 0.7071j$ and $z_2 = 0.7071 - 0.7071j$

Note that $|z_1| = |z_2| = 1$ so poles are on the unit circle:

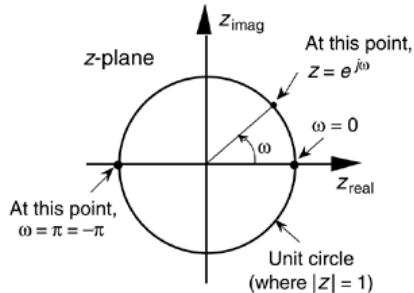
```
% norm(z(1)), norm(z(2))
% Verify that the poles are on the unit circle
zplane(1, eig_vec_min')
```



In general, the z is defined as follows:

$$z = A e^{j2\pi f} = A[\cos(2\pi f) + j \sin(2\pi f)]$$

Recall that $\angle z = 2\pi f$ and $f = \frac{\angle z}{2\pi}$ because of the definition of the complex number z :



Since the two z quantities that we found are on the unit circle, their magnitude $A = 1$.

We also found that two pair of z s are complex conjugates of one another. So we only use one of them to compute the frequency:

```
f1 = abs(angle(z(2))) / (2*pi)
```

```
f1 = 0.1250
```

Step 4: Solve Eq. (14.5.11) for the signal powers $\{P_i\}$

$$\begin{bmatrix} \cos 2\pi f_1 & \cos 2\pi f_2 & \cdots & \cos 2\pi f_p \\ \cos 4\pi f_1 & \cos 4\pi f_2 & \cdots & \cos 4\pi f_p \\ \vdots & \vdots & & \vdots \\ \cos 2\pi p f_1 & \cos 2\pi p f_2 & \cdots & \cos 2\pi p f_p \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_p \end{bmatrix} = \begin{bmatrix} \gamma_{yy}(1) \\ \gamma_{yy}(2) \\ \vdots \\ \gamma_{yy}(p) \end{bmatrix} \quad (14.5.11)$$

Since we only have one frequency, the equation becomes:

$\cos(2\pi f_1)P_1 = \gamma_{yy}(1)$ where $\gamma_{yy}(1)$ is the second autocorrelation value:

We want to find P_1 :

$$P_1 = \frac{\gamma_{yy}(1)}{\cos(2\pi f_1)}$$

```
P1 = r_yy(2) / cos(2*pi*f1)
```

P1 = 1.4142

From section 14.5.1, we are given:

$$P_i = \frac{A_i^2}{2}$$

This allows us to estimate the amplitude A_i given P_i :

$$A_i = \sqrt{2P_i}$$

```
A1 = sqrt(2*P1)
```

A1 = 1.6818

Functions

```
function [A]=test()
A=1
end
```

```

function [F, A, P, lambda_min]=pisarenko(r_xx, p)
    % Estimates the frequencies and amplitudes using the Pisarenko method
    % r_xx: autocorrelation sequence starting from zero.
    % The length of ACRS must be at least 2p+1.
    % p: assumed number of sinusoids in the signal
    % F: normalised frequencies of the sinuoids
    % A: amplitudes of the sinuoids
    if numel(r_xx) < 2*p+1
        error(strcat('The length of ACRS must be at least ', int2str(2*p+1)));
    end

    % Compute the autocorrelation matrix
    R_xx = toeplitz(r_xx(1:2*p+1));

    % Perform the eigendecomposition
    [eigvecs, eigvals] = eigs(R_xx, size(R_xx, 1), 'smallestreal');

    eigvals = diag(eigvals);
    lambda_min = eigvals(1);

    % Find the eigenvector 'a' corresponding to the smallest eigenvalue.
    % The function eigs() sorts eigenvectors, so just pick the first column.
    a = eigvecs(:, 1);

    % Ensure that a_0 = 1 (this is by definition)
    a = a / a(1);

    % The elements of this eigenvector corresponds to the parameters
    % of an ARMA(2p, 2p) model: a_0, a_1, ..., a_2p where p: number of sinusoids
    % The polynomial A(z) in (14.5.4) has 2p poles on the unit circle.
    % Obtain the poles by finding the roots of the system.
    z = roots(a);

    % Estimate frequencies
    F = zeros(p, 1);
    for i = 1:p
        % The poles come in pairs. Each pair is complex conjugate
        % of one another. Only use one of them and find the absolute value.
        z_i = z(2*i);
        F(i) = abs(angle(z_i)) / (2*pi);
    end

    % Build the matrix of cosines
    C = zeros(p);
    for i = 1:p
        for j = 1:p
            C(i, j) = cos(i * 2*pi * F(j));
        end
    end

    % Start the ACRS from the second element according to equation (14.5.11)
    gamma = r_xx(2:p+1);

    % Solve equation (14.5.11) for P
    P = C\gamma; % Same as `inv(C)*gamma` but faster and more accurate

    % Since P=A^2/2, we can compute the amplitude A=sqrt(2*P)
    A = sqrt(2 * P);

```

end

Probability

Table of Contents

Statistics.....	2
Parameter vs. Statistic.....	2
Estimators.....	2
Estimator Evaluation.....	2
Bias.....	3
Variance.....	3
Mean Square Error.....	3
Bias-variance tradeoff.....	3
Consistency.....	4
Probability.....	4
Random variables.....	4
Statistical averages.....	5
Expectation.....	5
Expectation rules.....	5
Variance.....	5
Variance Rules.....	6
Standard Deviation.....	6
Probability Distributions (Probability Density Functions).....	6
Useful Distribution: Gaussian or Normal Distribution.....	7
Generating a realisation in MATLAB.....	8
Properties of normal random variables.....	8
Useful Distribution: Uniform Distribution.....	9
Generate realisation in MATLAB.....	9
Jointly distributed random variables.....	10
Covariance.....	10
Correlation.....	11
[✓] Quiz: sketch the probability density function of a sine wave.....	11
[✓] Quiz: sketch a realisation of a discrete-time random process.....	13
[✓] Quiz: compute the expectation of a random variable given probability mass function.....	13
[✓] Quiz: find the probability density function from a realisation signal.....	14
Problem 13.13: Decompose expectation (MSE objective function).....	16
a) Express the objective function in terms of its parameters.....	16
b) Using partial derivatives to determine the values of parameters.....	17
Problem 13.22: Marginal and conditional distributions.....	18
a) Determine marginal distributions and conditional probabilities.....	18
b) Determine if two random variables are independent.....	20
ADSI Problem 4.1: Discrete distribution (Poisson).....	21
1) Sketch the density function.....	21
2) Calculate the probability of measuring 4 or 5 events in an experiment.....	22
3) Calculate the probability of measuring 3 or more events in an experiment.....	22
ADSI Problem 4.3: Realisation a discrete random process.....	23
1) Draw a realisation of a random process.....	23
2) Is the random process deterministic or non-deterministic?.....	24
3) Plot realisations of the random process in MATLAB.....	25
Exam 2018 Problem 5: Given PDS function, marginal, correlations.....	26
[✓] 1) Show that the probability density function is valid.....	26
[✓] 2) Calculate the marginal probability density function.....	27
[✓] 3) Without calculations, argue whether $\text{var}(X)$ or $\text{var}(Y)$ is larger.....	28
[✓] 4) Show that X and Y are uncorrelated.....	28
Exam 2018 Problem 5: Continuous Probability Density Function.....	30
[✓] 1) Find a valid probability density function.....	30
[✓] 2) Compute the probability given a probability density function?.....	31

[✓] 3) Compute the expected value a function:	32
Exam 2016 Problem 4: Discrete Random Process	32
1) Sketch a realization of a signal with this density function.	33
2) Compute marginal density function.	34
3) Determine whether signal is deterministic or random by calculating the correlation.	36
Functions	36

Statistics

Parameter vs. Statistic

In statistics, a **parameter** denoted θ is a numerical value that states something about the entire population. Since a parameter is difficult to obtain, we use a statistic. A **statistic** is a numerical value that states something about a small sample of the entire population. The value of a parameter is a fixed number whereas a statistic can vary because it depends on the given sample.

Estimators

An **estimator** denoted $\hat{\theta}$ is a rule or a formula that can approximate some parameter θ given a sample of N observations.

For example, to estimate the mean value m of a random variable X , we can use the arithmetic average as an estimate:

$$\hat{m} = \frac{1}{N} \sum_{k=1}^N x_k.$$

This is called the **sample mean** estimator.

Sample variance

$$\hat{\sigma}^2 \triangleq \frac{1}{N} \sum_{k=1}^N (x_k - \hat{m})^2$$

Sample covariance

$$\hat{\sigma}_{xy} \triangleq \frac{1}{N} \sum_{k=1}^N (x_k - \hat{m}_x)(y_k - \hat{m}_y)$$

Estimator Evaluation

The quality of an estimator $\hat{\theta}$ of a parameter θ is evaluated based on three properties:

1. Bias
2. Variance
3. Mean square error

Bias

$$B(\hat{\theta}) \triangleq E(\hat{\theta}) - \theta. \quad (14.3)$$

When $B(\hat{\theta}) = 0$ then the estimator yields the correct value on average when used a large number of time. An estimator with $B = 0$ is called **unbiased** estimator.

Variance

$$\text{var}(\hat{\theta}) = E\{[\hat{\theta} - E(\hat{\theta})]^2\} \quad (14.4)$$

The variance describes the spread of the estimates about its expected value.

Mean Square Error

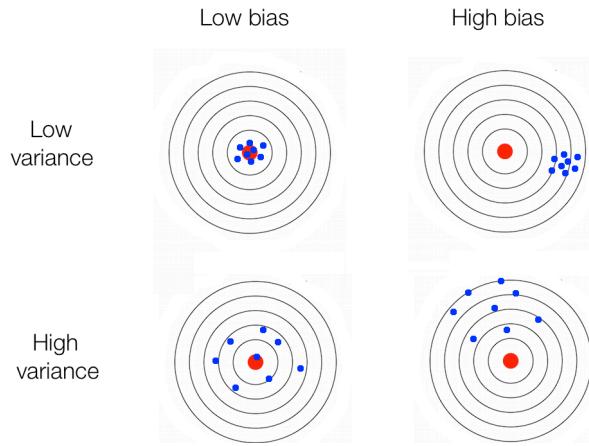
measures the average deviation of the estimator $\hat{\theta}$ from the true value θ :

$$\text{mse}(\hat{\theta}) = E[(\hat{\theta} - \theta)^2]. \quad (14.5)$$

By rewrite Eq. (14.5) we see that the mean square error takes into consideration both ***the variance and the bias*** of the estimator:

$$\text{mse}(\hat{\theta}) = \text{var}(\hat{\theta}) + B^2(\hat{\theta}). \quad (14.6)$$

Bias-variance tradeoff



A good estimator should have zero bias and the smallest possible variance. However, typically these two quantities conflict with each other. This is known as the bias-variance tradeoff.

Consistency

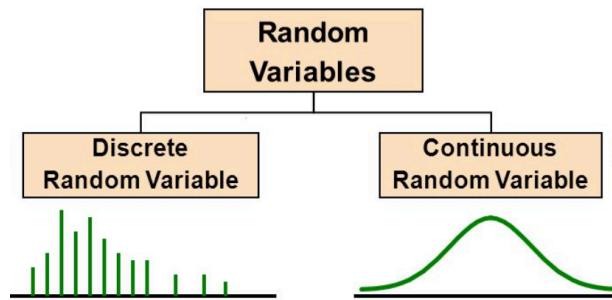
An estimator is said to be **consistent** if its estimate converges to the true value θ as the number of observations N approaches infinity.

From Eq. (14.6), we see that if variance and the bias of the estimator is zero then the estimator converges to the true value.

$$\text{mse}(\hat{\theta}) = \text{var}(\hat{\theta}) + B^2(\hat{\theta}). \quad (14.6)$$

Therefore, a sufficient condition for a consistent estimator is that both its variance and bias converge to zero as the number of observations becomes very large.

Probability



Random variables

A random variable is a variable that can take on a numerical value determined by the outcome of a random experiment.

A random variable is defined on a sample space S .

A random variable $X(\zeta)$ is a function that assigns a real number to any outcome ζ i.e. $X : S \rightarrow \mathbb{R}$. For simplicity, we often drop the dependence on the outcome and write X .

We use capital letters to denote random variables and lower-case letters for values they may take. For example, $X = x$ means that X takes on the value x .

A **discrete** random variable can take values from a finite or countably infinite set of numbers.

A **continuous** random variable can take values from any real number in a specified range.

Statistical averages

Expectation

The **mean value** or **expectation** of a random variable X is defined as:

$$(13.7) \quad m_x \triangleq E(X) \triangleq \int_{-\infty}^{\infty} xf_X(x)dx.$$

The mean value is an average of the values that a random variable takes on, where each value is weighted by the probability that the random variable is equal to that value.

The mean value is a measure of where the values of the random variable X are centered.

Expectation rules

Let X and Y be two random variables and a and b be two constants. Following expectation properties apply:

1. $E[a] = a$ e.g. $E(42) = 42$
2. $E[aX] = aE[X]$ e.g. if you multiply every value by 2, the expectation doubles
3. $E[a \pm X] = a \pm E[X]$ e.g. if you add 42 to every case, the expectation increases by 42
4. $E[X + Y] = E[X] + E[Y]$
5. If X and Y are independent, then $E[XY] = E[X]E[Y]$
6. $E[a \pm bX] = a \pm bE[X] = a \pm bE[X]$
7. $E[b(a \pm X)] = bE[a \pm X] = b(a \pm E[X])$
8. $E[aX + bY] = aE[X] + bE[Y]$ (using rule 2 and 4)

Variance

The **variance** of a random variable X is defined as:

$$(13.8) \quad \sigma_x^2 \triangleq \text{var}(X) \triangleq E[(x - m_x)^2] = \int_{-\infty}^{\infty} (x - m_x)^2 f_X(x)dx,$$

Another way to compute the variance is:

$$(13.11) \quad \text{var}(X) = E[X^2] - (E[X])^2 = E[X^2] - m_x^2$$

The variance measures the spread of the distribution about its mean value.

A small variance indicates that X is more likely to assume values close to its mean.

A large variance indicates that the values of X are spread over a wider interval about the mean.

The variance is used as a measure of variability of a random variable.

Variance Rules

Let X and Y be two random variables and a and b be two constants. Following variance identities apply:

1. $\text{var}(a) = 0$
2. $\text{var}(a \pm X) = \text{var}(X)$
3. $\text{var}(bX) = b^2 \cdot \text{var}(X)$
4. $\text{var}(a \pm bX) = b^2 \cdot \text{var}(X)$
5. If X and Y are independent i.e. $\text{cov}(X, Y) = 0$ then $\text{var}(X \pm Y) = \text{var}(X) + \text{var}(Y)$
6. If X and Y are independent $\text{var}(aX + bY) = a^2 \text{var}(X) + b^2 \text{var}(Y)$

Standard Deviation

The **standard deviation** is a measure of spread which has the same units as the original observations:

$$(13.9) \quad \sigma_x \triangleq \sqrt{\text{var}(X)}.$$

Probability Distributions (Probability Density Functions)

We can study the nature of a random variable by its probability distribution.

A probability distribution can be constructed by creating a *histogram* of a large set of observations.

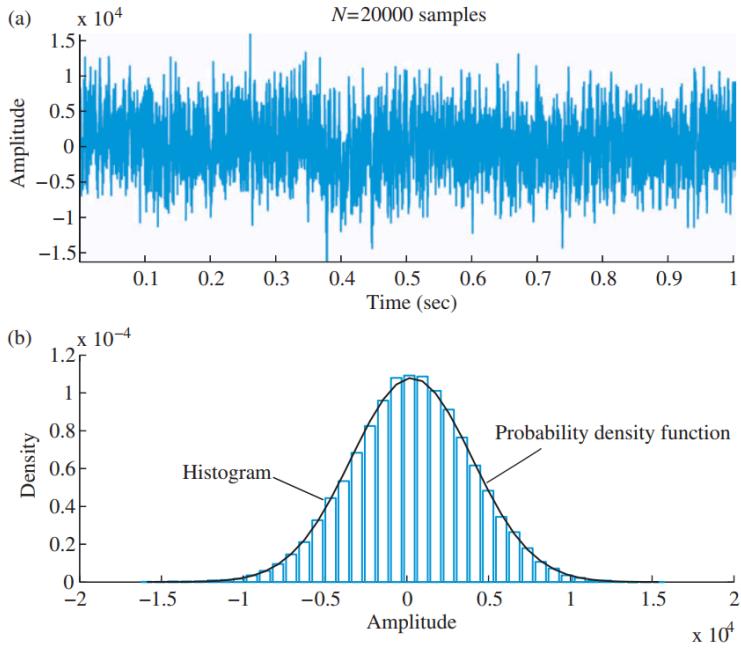


Figure 13.4 (a) Waveform of F-16 noise recorded at the co-pilot's seat with a sampling rate of 19.98 kHz using a 16 bit ADC. (b) Histogram and theoretical probability density function.

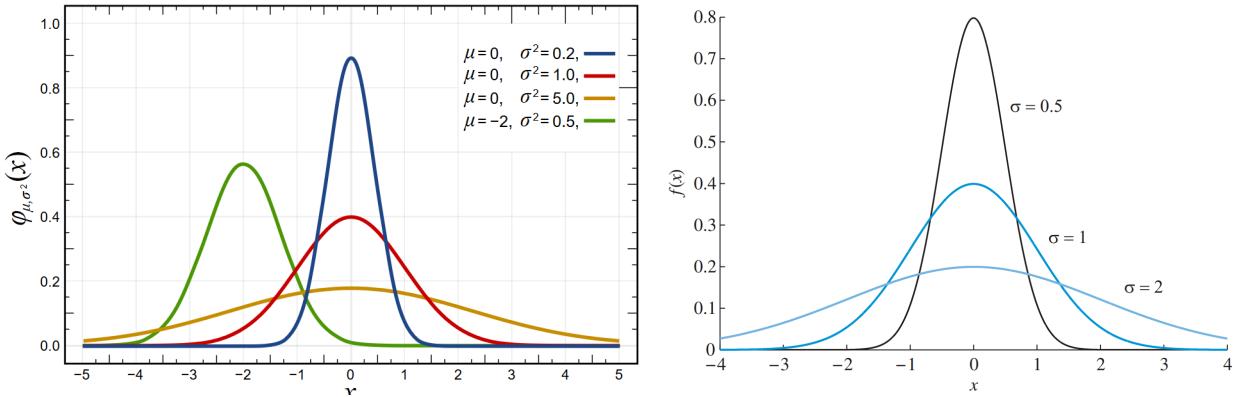
A probability distribution of a continuous random variable X denoted $f_X(x)$ is called the **probability density function** (PDF).

Note! that the function $f_X(x)$ does not represent the probability of any event. It is only when the function is integrated between two points that it yields a probability.

$$(13.3) \quad \Pr(a_1 < x < a_2) = \int_{a_1}^{a_2} f_X(x) dx$$

Useful Distribution: Gaussian or Normal Distribution

A random variable X normally distributed with mean m and variance σ^2 is denoted $X \sim N(m, \sigma^2)$.



The probability density function of a normal distribution for various values of the standard deviation. The spread of the distribution increases with increasing σ ; the height is reduced because the area is always equal to unity.

Its probability density function is given by:

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}. \quad -\infty < x < \infty$$

The mean is

$$E(X) = \mu$$

The variance is:

$$\text{var}(X) = \sigma^2$$

Generating a realisation in MATLAB

To generate N random numbers for a Gaussian random variable with mean μ and variance σ^2 we use following MATLAB functions

```
clear variables;

N = 1000;
variance = 25;
std_dev = sqrt(variance);
mu = 500;
w = std_dev.*randn(N,1) + mu;

% Calculate the sample mean, standard deviation, and variance.
stats = [mean(w) std(w) var(w)]
```

```
stats = 1x3
500.1909    5.1244    26.2591
```

Properties of normal random variables

Any linear combination of normal random variables is itself a normal random variable.

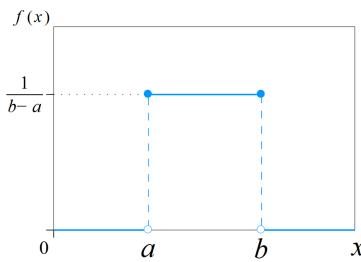
An important property of normal random variables is that if X is normal with mean m and variance σ^2 , then $aX + b$ is normally distributed with mean $am + b$ and variance $a^2\sigma^2$ (see [Tutorial Problem 4](#)), that is

$$X \sim N(m, \sigma^2) \Rightarrow Y = aX + b \sim N(am + b, a^2\sigma^2). \quad (13.17)$$

Central Limit Theorem: the sum of a large number of independent random variables has approximately a normal distribution.

Useful Distribution: Uniform Distribution

A random variable X uniformly distributed over the interval (a, b) is denoted by $X \sim U(a, b)$.



The probability density function is given by:

$$f_X(x) = \begin{cases} 1/(b-a), & \text{if } a < x < b \\ 0, & \text{otherwise} \end{cases}$$

The mean is

$$E(X) = \frac{b+a}{2}$$

The mean of squared is:

$$E(X^2) = \frac{a^2 + b^2 + ab}{3}$$

The variance is:

$$\text{var}(X) = E(X^2) - [E(X)]^2 = \frac{1}{12}(b-a)^2$$

Generate realisation in MATLAB

The MATLAB function **rand** generates random numbers according to $X \sim U(0, 1)$

By default, `rand` returns normalized values (between 0 and 1) that are drawn from a uniform distribution. To change the range of the distribution to a new range, (a, b) , multiply each value by the width of the new range, $(b - a)$ and then shift every value by a .

```
a = 50;
b = 100;
r = (b-a).*rand(4, 1) + a
```

```
r = 4x1
82.2744
69.2663
54.8620
98.1518
```

Generate a random value for the uniform distribution $\phi \sim U(0, 2\pi)$:

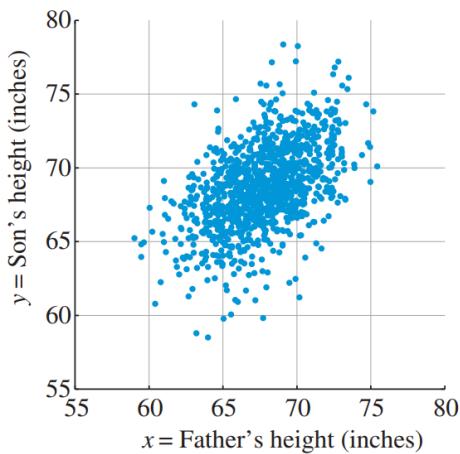
```
phi = 2*pi*rand()
```

```
phi = 5.5722
```

Jointly distributed random variables

Convariance

Covariance provides a measure of the association between two random variables X and Y . The covariance can be used to summarise the information provided by the following scatter plot:

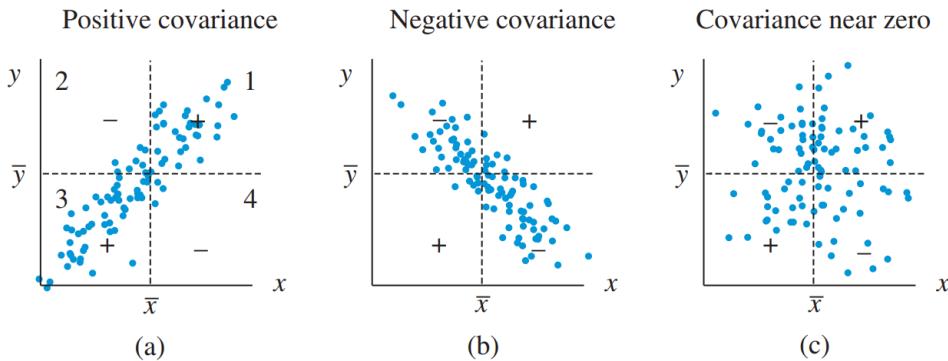


Covariance The *covariance* of two random variables X and Y is defined by

$$(13.25) \quad c_{xy} \triangleq \text{cov}(X, Y) \triangleq E[(X - m_x)(Y - m_y)] = E(XY) - E(X)E(Y)$$

Two random variables with zero covariance are said to be *uncorrelated*. Thus,

$$(13.27) \quad X, Y \text{ uncorrelated} \Leftrightarrow \text{cov}(X, Y) = 0 \text{ or } E(XY) = E(X)E(Y).$$



- a) The relationship between X and Y is positive means that as X increases so does Y
- b) The relationship between X and Y is negative meaning that as X increases, Y will decrease.
- c) The random variables X and Y are not related i.e. uncorrelated.

Correlation

The correlation between two random variables X and Y is defined as:

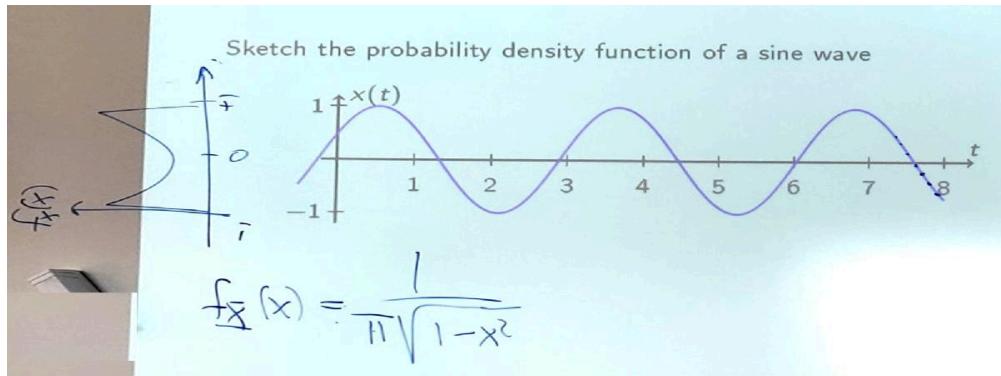
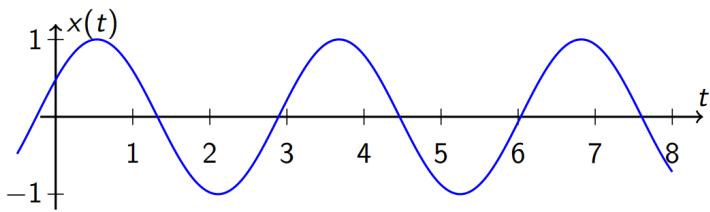
$$r_{xy} = E[XY]$$

Correlation and covariance may create some confusion. Zero correlation implies that X and Y are orthogonal whereas zero covariance implies that X and Y are uncorrelated.

Table 13.1 Relationship between orthogonal and uncorrelated random variables.

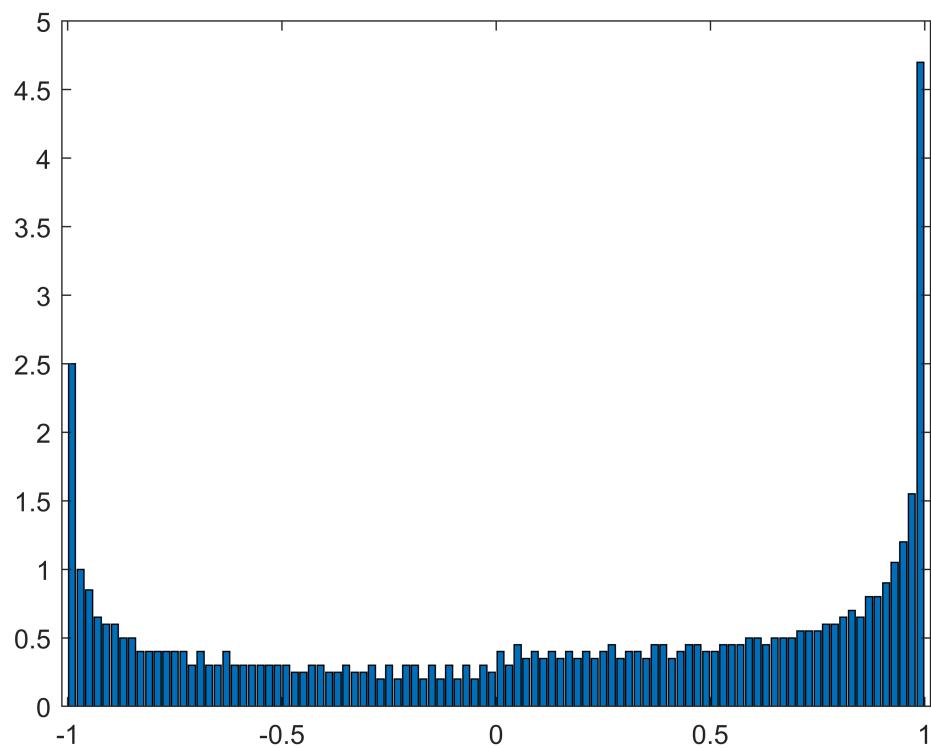
Correlation:	$r_{xy} = E(XY)$	$r_{xy} = 0 \Rightarrow X, Y$ orthogonal
Covariance:	$c_{xy} = E[(X - m_x)(Y - m_y)]$	$c_{xy} = 0 \Rightarrow X, Y$ uncorrelated

[✓] Quiz: sketch the probability density function of a sine wave



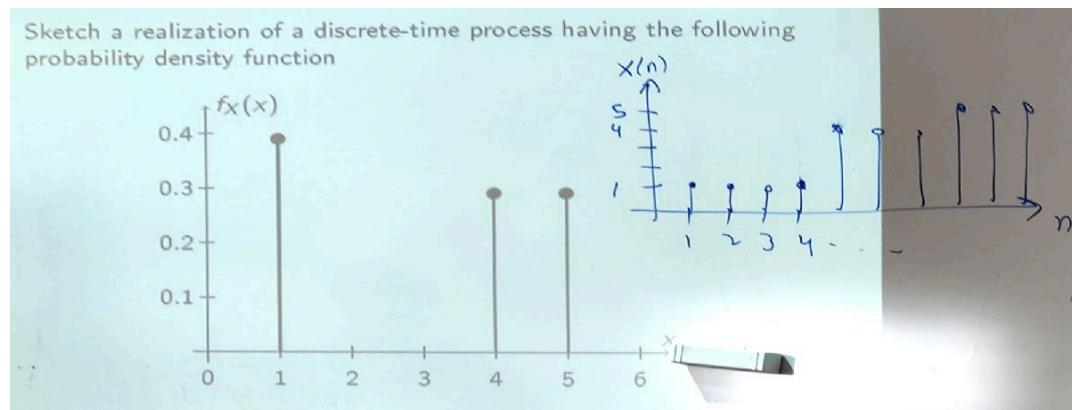
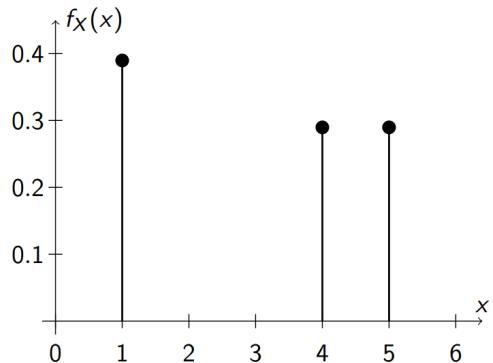
Try it out in MATLAB:

```
n = linspace(0, 8, 1000);
y = sin(n);
[xo, px] = epdf(y, 100);
bar(xo, px);
```



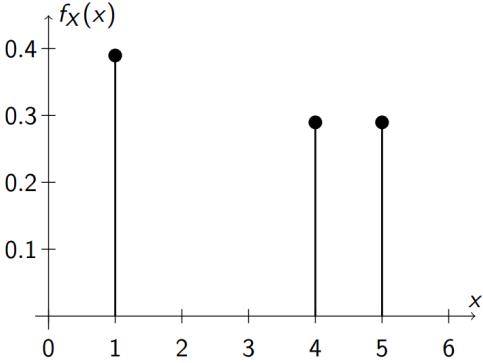
[✓] Quiz: sketch a realisation of a discrete-time random process

Sketch a realisation of a discrete-time process having the following probability density function



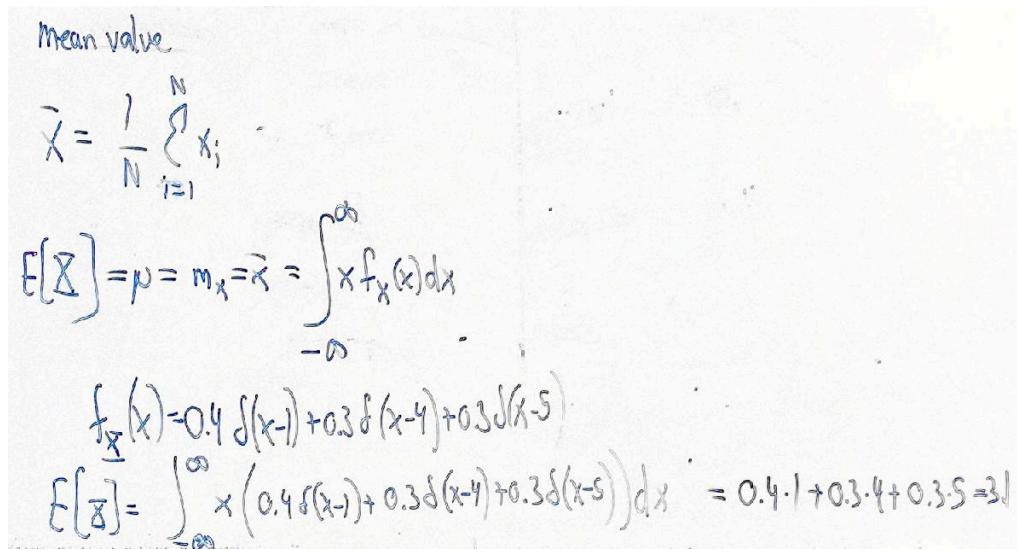
[✓] Quiz: compute the expectation of a random variable given probability mass function

Calculate the mean value of a random variable with the following probability density function:



We can write the probability density function as:

$$\begin{aligned} f_X(x) &= \frac{4}{10} \delta[x - 1] + \frac{0}{10} [x - 2] + \frac{0}{10} [x - 3] + \frac{3}{10} [x - 4] + \frac{3}{10} [x - 5] \\ &= \frac{4}{10} \delta[x - 1] + \frac{3}{10} [x - 4] + \frac{3}{10} [x - 5] \end{aligned}$$



[✓] Quiz: find the probability density function from a realisation signal

For this problem we will use two random variables.

The random variable X_0 tells us the value of a signal at time t_0 whereas X_1 tells us the value of the same signal at time $t_0 + 1$ i.e., one time unit later. We are making two measurements at the same signal.

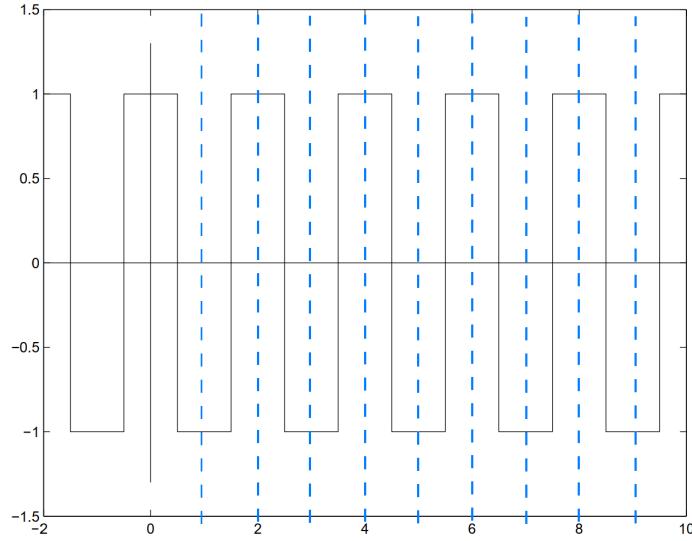
We use a generic formula for the joint PDF:

$$f_{X_0, X_1}(x_0, x_1) = p_0 \delta[x_0 + 1] \delta[x_1 + 1] + p_1 \delta[x_0 + 1] \delta[x_1 - 1] + p_2 \delta[x_0 - 1] \delta[x_1 + 1] + p_3 \delta[x_0 - 1] \delta[x_1 - 1]$$

where:

1. p_0 is the probability that if the signal is +1 at time t_0 then it will remain +1 at $t_0 + 1$
2. p_1 is the probability that if the signal is +1 at time t_0 then it will change -1 at $t_0 + 1$
3. p_2 is the probability that if the signal is -1 at time t_0 then it will change +1 at $t_0 + 1$
4. p_3 is the probability that if the signal is -1 at time t_0 then it will remain -1 at $t_0 + 1$

At each given signal, we make two measurements and see what is the likelihood that the signal will remain the same or change.



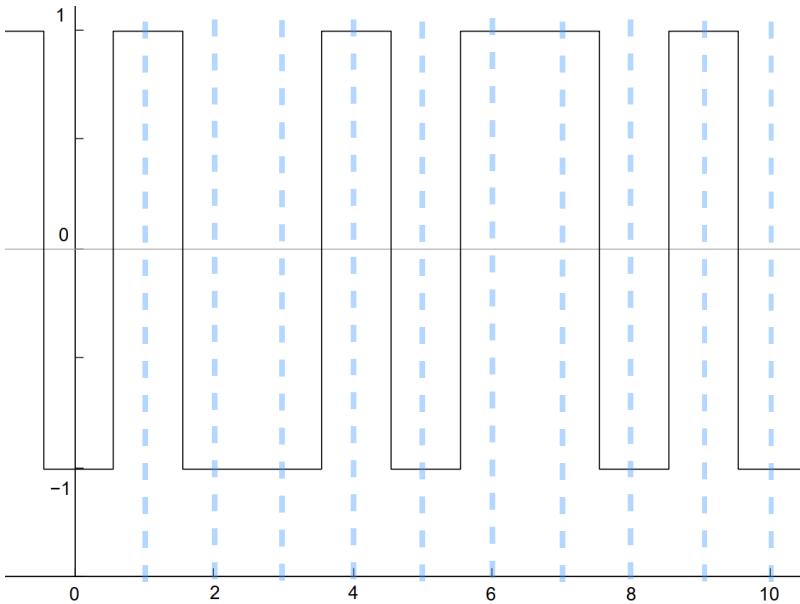
From the signal above, we make following observations:

1. $p_0 = 0$ because we never observe that when the signal is +1 at time t_0 then it will remain +1 at time $t_0 + 1$
2. $p_1 = 0.5$ because we observe that when the signal is +1 at time t_0 then it always changes to -1 at time $t_0 + 1$.
3. $p_2 = 0.5$ because when the signal is -1 at t_0 then it flips to +1 at $t_0 + 1$
4. $p_3 = 0$ because we never observe that when the signal is -1 at time t_0 then it will remain -1 at time $t_0 + 1$

Thus, the joint PDF for square signal is:

$$f_{X_0, X_1}(x_0, x_1) = \frac{1}{2} \delta[x_0 + 1] \delta[x_1 - 1] + \frac{1}{2} \delta[x_0 - 1] \delta[x_1 + 1]$$

Another more interesting problem is the signal below:



This is a short realisation so it is difficult to find the likelihoods by looking at the signal alone.

Problem 13.13: Decompose expectation (MSE objective function)

- 13.** Consider the mse objective function (13.56)

$$J(a, b) = E[(Y - aX - b)^2].$$

a) Express the objective function in terms of its parameters

- (a) Express $J(a, b)$ in terms of the parameters a , b , and the moments of X and Y .

Use MATLAB to expand the expression inside the expected value:

```
syms a b X Y
expand((Y-a*X - b)^2)
```

$$\text{ans} = X^2 a^2 - 2 X Y a + 2 X a b + Y^2 - 2 Y b + b^2$$

So we have:

$$J(a, b) = E[X^2a^2 - 2XYa + 2Xab + Y^2 - 2Yb + b^2]$$

Let X and Y be two random variables and a and b be two constants. Following expectation identities apply:

1. $E[a] = a$ e.g. $E(42) = 42$
2. $E[aX] = aE[X]$ e.g. if you multiply every value by 2, the expectation doubles
3. $E[a \pm X] = a \pm E[X]$ e.g. if you add 42 to every case, the expectation increases by 42
4. $E[X + Y] = E[X] + E[Y]$
5. If X and Y are independent, then $E[XY] = E[X]E[Y]$

Use rule 4:

$$J(a, b) = E[X^2a^2 - 2XYa + 2Xab + Y^2 - 2Yb + b^2]$$

$$J(a, b) = E[X^2a^2] - E[2XYa] + E[2Xab] + E[Y^2] - E[2Yb] + E[b^2]$$

Use rule 1 and rule 2:

$$J(a, b) = a^2E[X^2] - 2aE[X]Y + 2abE[X] + E[Y^2] - 2bE[Y] + b^2$$

b) Using partial derivatives to determine the values of parameters

(b) Using partial derivatives $\frac{\partial J}{\partial a}$ and $\frac{\partial J}{\partial b}$, determine the values of a and b by solving the equations $\partial J / \partial a = 0$ and $\partial J / \partial b = 0$ that minimize $J(a, b)$ to obtain optimum values given in (13.58) and (13.62).

First, take the partial derivatives:

$$\frac{\partial J(a, b)}{\partial a} = 2aE[X^2] - 2E[X]Y + 2bE[X]$$

$$\frac{\partial J(a, b)}{\partial b} = 2aE[X] - 2E[Y] + 2b$$

Next, solve the equations:

$$(Eq. 1) 2aE[X^2] - 2E[X]Y + 2bE[X] = 0$$

$$(Eq. 2) 2aE[X] - 2E[Y] + 2b = 0$$

Isolate b in (Eq. 2):

$$2b = -2aE[X] + 2E[Y]$$

$$b = -aE[X] + E[Y]$$

$$b = E[Y] - aE[X]$$

This corresponds to (13.58) in the book:

$$(13.58) \quad b_0 = m_y - am_x$$

Now, plug the expression for b into Eq. 1 in order to find an expression for a :

$$\begin{aligned} 2aE[X^2] - 2E[XY] + 2bE[X] &= 0 \\ 2aE[X^2] - 2E[XY] + 2(E[Y] - aE[X])E[X] &= 0 \\ 2aE[X^2] - 2E[XY] + 2E[X]E[Y] - 2aE[X]E[X] &= 0 \\ 2aE[X^2] - 2aE[X]E[X] - 2E[XY] + 2E[X]E[Y] &= 0 \\ 2a(E[X^2] - E[X]E[X]) - 2E[XY] + 2E[X]E[Y] &= 0 \\ 2a(E[X^2] - E[X]E[X]) &= 2E[XY] - 2E[X]E[Y] \\ a(E[X^2] - E[X]E[X]) &= E[XY] - E[X]E[Y] \\ a &= \frac{E[XY] - E[X]E[Y]}{E[X^2] - E[X]E[X]} = \frac{E[XY] - E[X]E[Y]}{E[X^2] - E[X]^2} \end{aligned}$$

We have found an expression for a . The numerator looks like it is the covariance:

Covariance The *covariance* of two random variables X and Y is defined by

$$(13.25) \quad c_{xy} \triangleq \text{cov}(X, Y) \triangleq E[(X - m_x)(Y - m_y)] = E(XY) - E(X)E(Y)$$

The denominator looks like it is the variance:

$$(13.11) \quad \text{var}(X) = E[X^2] - E[X]^2 = E[X^2] - m_x^2$$

Therefore, the derived expression is the same as (13.62) in the book.

$$(13.62) \quad a_0 = \frac{c_{xy}}{\sigma_x^2} = \rho_{xy} \frac{\sigma_y}{\sigma_x}$$

Problem 13.22: Marginal and conditional distributions

22. Consider two jointly distributed random variables X and Y with pdf

$$f(x, y) = \begin{cases} 8xy, & 0 \leq x \leq 1, 0 \leq y \leq x \\ 0, & \text{otherwise} \end{cases}$$

```
clear variables;
```

a) Determine marginal distributions and conditional probabilities

(a) Determine $f(x)$, $f(y)$, $f(x|y)$, and $f(y|x)$.

```
syms x y
```

```
fXY = 8*x*y;
```

The **marginal** distributions of random variables X and X are obtained by integration as follows:

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y) dy \quad \text{and} \quad f_Y(y) = \int_{-\infty}^{\infty} f_{X,Y}(x,y) dx, \quad (13.19)$$

```
fX = int(fXY, y, 0, x)
```

$$f_X = 4x^3$$

```
fY = int(fXY, x, 0, 1)
```

$$f_Y = 4y$$

To compute $f(x|y)$ we use following relation:

$$f_{X,Y}(x,y) = f_{Y|X}(y|x)f_X(x) = f_{X|Y}(x|y)f_Y(y). \quad (13.23)$$

From Eq. 13.23, we know that:

$$f(x|y) = \frac{f(x,y)}{f(y)} \quad \text{and} \quad f(y|x) = \frac{f(x,y)}{f(x)}$$

```
fX_given_Y = fXY / fY
```

$$f_X \text{given } Y = 2x$$

```
fY_given_X = fXY / fX
```

$$f_Y \text{given } X =$$

$$\frac{2y}{x^2}$$

We can do it by hand too.

Compute the marginal distribution of X :

$$f(x) = \int_{-\infty}^{\infty} f(x,y) dy$$

Since $f(x,y)$ has non-zero values for when $y \geq 0$ and $y \leq x$ i.e., y is between 0 and x , then we only need to find the integral w.r.t. y from 0 to x .

$$f(x) = \int_0^x 8xy dy = \left[\frac{1}{2} 8xy^2 \right]_0^x = \frac{1}{2} 8xx^2 - 0 = 4x^3$$

where $0 \leq x \leq 1$.

Compute the marginal distribution of Y :

$$f(y) = \int_{-\infty}^{\infty} f(x, y) dx$$

We are given that $f(x, y)$ is non-zero when $x \geq 0$ and $x \leq 1$. Since we are trying to find the integral w.r.t. y , we integrate from y to 1.

$$f(y) = \int_y^1 8xy dx = [4x^2y]_y^1 = 4y - 4y^3$$

where $0 \leq y \leq 1$

To compute $f(x|y)$ we use following relation:

$$f_{X,Y}(x, y) = f_{Y|X}(y|x)f_X(x) = f_{X|Y}(x|y)f_Y(y). \quad (13.23)$$

From Eq. 13.23, we know that:

$$f(x|y) = \frac{f(x, y)}{f(y)} = \frac{8xy}{4y - 4y^3} = \frac{4y \cdot 2x}{4y(1 - y^2)} = \frac{2x}{1 - y^2}$$

and

$$f(y|x) = \frac{f(x, y)}{f(x)} = \frac{8xy}{4x^3} = \frac{4x \cdot 2y}{4x \cdot x^2} = \frac{2y}{x^2}$$

b) Determine if two random variables are independent

(b) Are X and Y independent?

Random variables X and Y are statistically independent, if

$$f(y|x) = f(y) \text{ or } f(x|y) = f(x).$$

In a) we have computed the following expressions:

- $f(x) = 4x^3$
- $f(y) = 4y - 4y^3$
- $f(y|x) = \frac{2y}{x^2}$
- $f(x|y) = \frac{2x}{1 - y^2}$

Clearly $f(y|x) \neq f(y)$ and $f(x|y) \neq f(x)$. Therefore, the answer is no! The random variables X and Y are not statistically independent.

ADSI Problem 4.1: Discrete distribution (Poisson)

The Poisson distribution is a discrete probability distribution that is used in model counting events i.e. pixel noise in CCD cameras and in particle detectors. The density function for the Poisson distribution is given by

$$f(x) = \sum_{k=0}^{\infty} \frac{a^k e^{-a}}{k!} \delta(x - k).$$

Where a is a fixed positive constant. The density function gives the probability of observing a particular number of events (x) in a given experiment when these events occur with a known average number of events per experiment (a) and the events are independent of one another. Let $a = 3$.

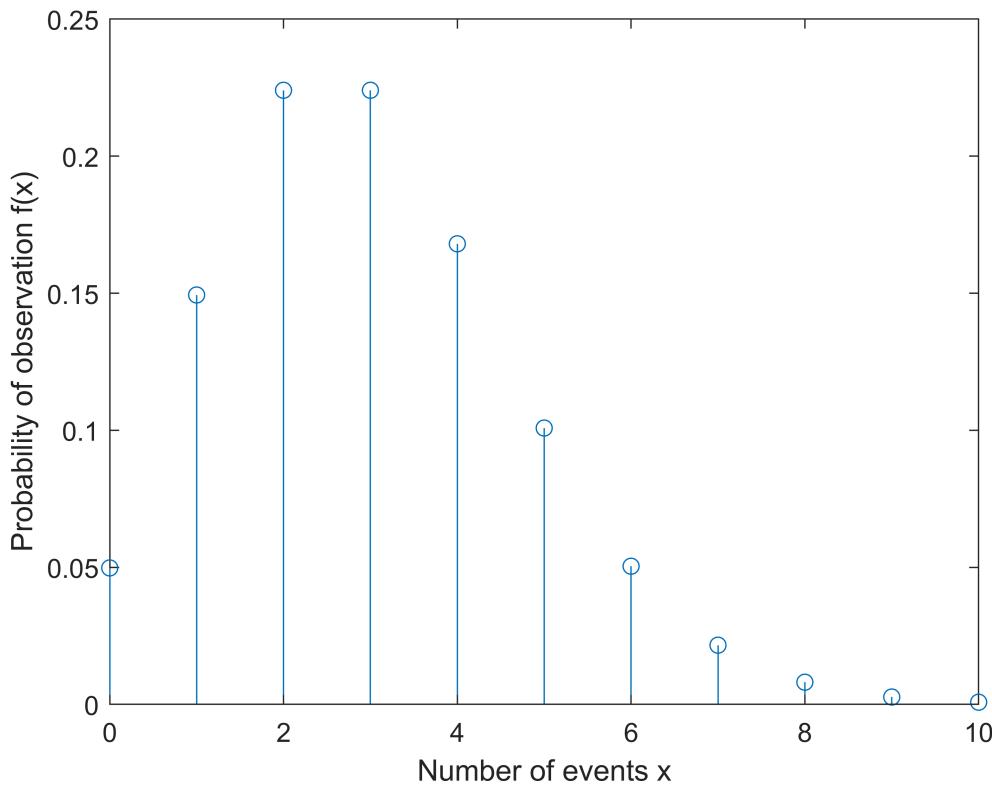
```
clear variables;
```

1) Sketch the density function

Use MATLAB to plot the density function. The probability mass function for the poisson distribution is:

$$f(k) = \frac{a^k}{k!} \exp(-a)$$

```
x = 0:10;
a = 3;
f = @(k) (exp(-a) * a.^k)/factorial(k);
%stem(x, f(x))
% applies the function f to the elements of x
y = arrayfun(f, x);
stem(x, y);
xlabel("Number of events x")
ylabel("Probability of observation f(x)")
```



2) Calculate the probability of measuring 4 or 5 events in an experiment.

When we measure, we cannot get both 4 and 5 at the same time. This means that the event of measuring 4 and the event of measuring 5 are mutually exclusive since these two events cannot happen at the same time.

Therefore, the probability that one of the mutually exclusive events occur is the sum of their individual probabilities.

$$\Pr(x = 4 \text{ or } x = 5) = \Pr(x = 4) + \Pr(x = 5) = f(4) + f(5)$$

$$f(4)+f(5)$$

$$\text{ans} = 0.2689$$

Final answer is: $\Pr(x = 4 \text{ or } x = 5) = 0.2689$

3) Calculate the probability of measuring 3 or more events in an experiment.

To compute the probability of measuring 3, or more events:

$$\Pr(x \geq 3) = \sum_{k=3}^{\infty} f(k)$$

Since we have an infinite sum, it is a bit difficult to compute. Instead, we can use on the fact that the sum over the mass function is 1. This means that we only need to compute the probabilities of $1 - (f(0) + f(1) + f(2))$ is the same as computing the infinite sum:

$$\Pr(x \geq 3) = 1 - \sum_{k=0}^2 f(k)$$

Let us do it in MATLAB:

```
1-(f(0) + f(1) + f(2))
```

```
ans = 0.5768
```

The final answer is: $\Pr(x \geq 3) = 0.5768$

ADSI Problem 4.3: Realisation a discrete random process

Let two random variables, X_1 and X_2 be defined from a random process $X(t)$ as $X_1 = X(t_1)$ and $X_2 = X(t_1 + 1)$. The joint density function for the two random variables is given by

$$f_{X_1 X_2}(x_1, x_2) = \frac{1}{8}\delta(x_1 - 1)\delta(x_2 - 1) + \frac{3}{8}\delta(x_1 - 1)\delta(x_2 + 1) \\ + \frac{3}{8}\delta(x_1 + 1)\delta(x_2 - 1) + \frac{1}{8}\delta(x_1 + 1)\delta(x_2 + 1).$$

The random variable X_1 tells us the value of a signal at time t_1 whereas X_2 tells us the value of the same signal at time $t_1 + 1$ i.e., one time unit later. We are making two measurements at the same signal.

```
clear variables;
```

1) Draw a realisation of a random process

1. Draw a realization of $X(t)$ and explain the considerations that you made in the preparation of the realization.

From the δ functions, we see that the signal can only take on two values: -1 and +1. This means that the realisation of the random process $X(t)$ flips between -1 and +1 i.e., $X(t) \in \{-1, 1\}$.

From each term we can read off the likelihood of the behaviour of the signal.

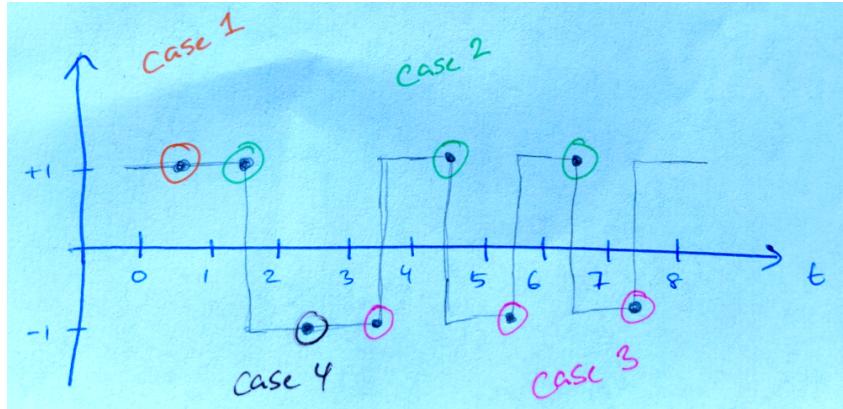
1. If the signal is +1 at t_1 then there is 1/8 probability that the signal will remain +1 at $t_1 + 1$
2. If the signal is +1 at t_1 then there is 3/8 probability that the signal will change to -1 at $t_1 + 1$
3. If the signal is -1 at t_1 then there is 3/8 probability that the signal will change to +1 at $t_1 + 1$
4. If the signal is -1 at t_1 then there is 1/8 probability that the signal will remain at -1 at $t_1 + 1$

This means that during a time window of 8 time units, the signal will behave as follows:

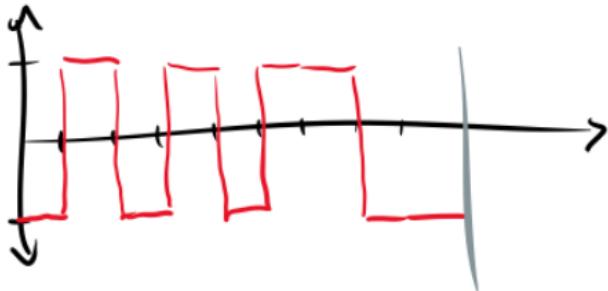
1. Given time t_0 when signal is +1, only once will the signal remain +1 at next time unit
2. Given time t_0 when signal is +1, three of the times the signal changes to -1 at next time unit

3. Given time t_0 when signal is -1, three of the times the signal changes to +1 at next time unit
4. Given time t_0 when signal is -1, only once will the signal remain -1 at next time unit

Thus, one realisation of the random process is as follows:



Another realisation of the random process is as follows:



2) Is the random process is deterministic or non-deterministic?

2. Discuss, using only the joint density function, whether the random process is deterministic or nondeterministic and state your arguments. Calculations are not called upon in this question.

At each given time instant t_0 the likelihood of the signal value changing sign at next time unit (either from +1 to -1 or from -1 to +1) is $\frac{6}{8} = \frac{3}{4}$.

And the likelihood of the signal value remaining the same is $\frac{2}{8} = \frac{1}{4}$.

So the joint probability density function only tells us that the signal will change sign or stay the same with some probability.

However, it does not say at what time instant the signal remain the same and at what time instants the signal will change.

Therefore, the random process is non-deterministic.

NOTE! We can make rather educated guesses for the next value of the signal so it wouldn't be completely unjustified to denote the signal as 'semi'-deterministic.

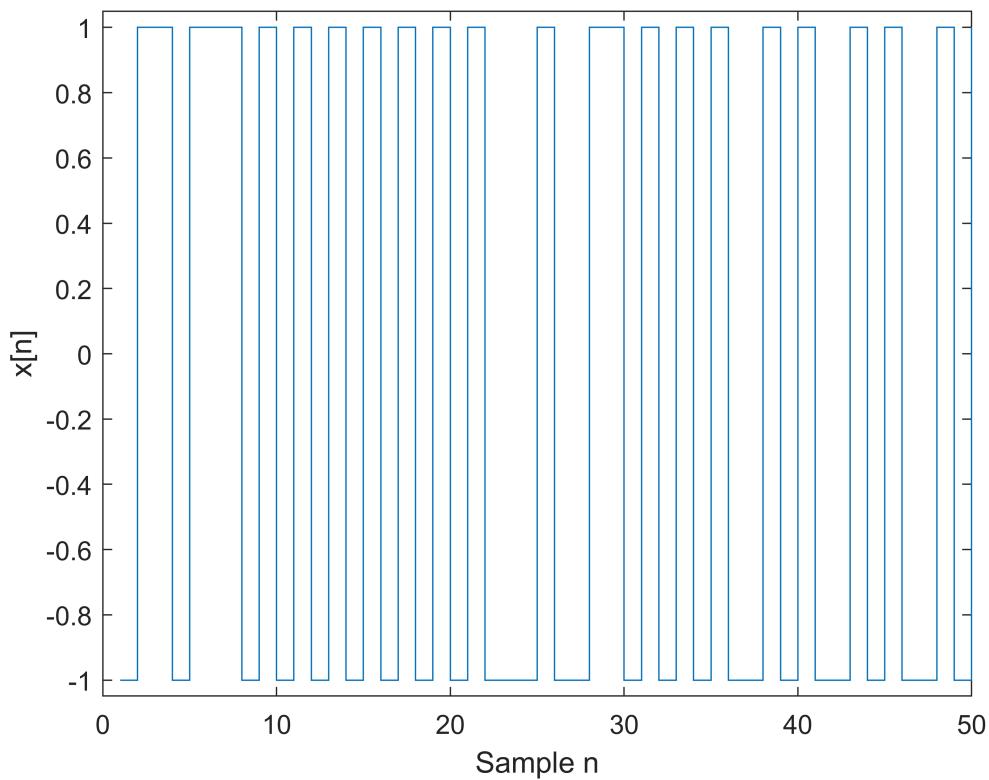
3) Plot realisations of the random process in MATLAB

3. Write a MATLAB script that can create and plot realizations of the random process.

At each given time instant t_0 the likelihood of the signal value changing sign at next time unit (either from +1 to -1 or from -1 to +1) is $\frac{6}{8} = \frac{3}{4}$. We will use this fact to implement a realisation of the random process.

```
% Setup
N = 50;
x=zeros(N,1);

%% equal probability of -1 or 1 in x(1)
if randn(1) >= 0
    x(1)=1;
else
    x(1)=-1;
end
%% change sign with 75% probability
for n=2:N
    if rand(1)<0.75
        x(n)=-x(n-1);
    else
        x(n)=x(n-1);
    end
end
%% plot result
stairs(x)
ylim([-1.05 1.05])
xlabel('Sample n')
ylabel('x[n]')
```



Exam 2018 Problem 5: Given PDS function, marginal, correlations

Consider the following joint probability density function.

$$f_{X,Y}(x, y) = \begin{cases} \frac{1}{2} & 1 \leq y \leq 2 \text{ and } 1 \leq x \leq 3 \\ 0 & \text{elsewhere} \end{cases}$$

[✓] 1) Show that the probability density function is valid

1. Show that $\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{X,Y}(x, y) dx dy = 1$.

Integrate using MATLAB:

```
syms x y f(x,y) fx(x) fy(y)
f(x,y) = 1/2;
inner = int(f(x,y), x, 1, 3);
outer = int(inner, y, 1, 2)
```

```
outer = 1
```

```
% Alternative method is to numerically evaluate double integral
fun = @(x,y) 1/2 + 0*(x+y); % Hack: 0*(x+y) is added for MATLAB
```

```
integral2(fun, 1, 3, 1, 2)
```

```
ans = 1
```

[✓] 2) Calculate the marginal probability density function

- Calculate the marginal probability density function $f_X(x)$.

The marginal probability density function is given as:

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y) dy$$

The marginal probability for X is given as:

$$f_X(x) = \int_1^2 \frac{1}{2} dy = \left[\frac{1}{2} y \right]_1^2 = 1 - \frac{1}{2} = \frac{1}{2}$$

Check it in MATLAB:

```
fX(x) = int(f(x,y), y, 1, 2)
```

```
fX(x) =  
1  
2
```

```
% Alternative method  
fun = @(y) 1/2 + 0*y; % Hack!  
integral(fun, 1, 2)
```

```
ans = 0.5000
```

The marginal probability for Y is given as:

$$f_Y(y) = \int_1^3 \frac{1}{2} dx = \left[\frac{1}{2} x \right]_1^3 = \frac{3}{2} - \frac{1}{2} = 1$$

```
fY(y) = int(f(x,y), x, 1, 3)
```

```
fY(y) = 1
```

```
% Alternative method  
fun = @(x) 1/2 + 0*x; % Hack!  
integral(fun, 1, 3)
```

```
ans = 1.0000
```

[✓] 3) Without calculations, argue whether $\text{var}(X)$ or $\text{var}(Y)$ is larger

The variance is a measure of the spread of the distribution about its mean value. A large variance indicates that the values of the random variable are spread over a wider interval about the mean. In this problem, X has a wider span of possible values. Therefore, $\text{var}(X) > \text{var}(Y)$.

[✓] 4) Show that X and Y are uncorrelated

Two random variables are uncorrelated if

$$\text{cov}(X, Y) = E(XY) - E[X]E[Y] = 0 \quad \text{or} \quad E[XY] = E[X]E[Y]$$

The simplest way to show that X and Y are uncorrelated is if the two random variables are independent.

Two random variables X and Y are said to be independent if:

$$f_{X,Y}(x, y) = f_X(x)f_Y(y)$$

The joint probability is given as:

$$f_{X,Y}(x, y) = \frac{1}{2}$$

The marginal probability for X is given as:

$$f_X(x) = \int_1^2 \frac{1}{2} dy = \left[\frac{1}{2} y \right]_1^2 = 1 - \frac{1}{2} = \frac{1}{2}$$

The marginal probability for Y is given as:

$$f_Y(y) = \int_1^3 \frac{1}{2} dx = \left[\frac{1}{2} x \right]_1^3 = \frac{3}{2} - \frac{1}{2} = 1$$

Since they are the random variables X and Y are independent they are also uncorrelated.

Another approach involves computing the covariance and showing that

$$\text{cov}(X, Y) = E(XY) - E[X]E[Y] = 0$$

We compute $E[X]$

$$E[X] = \int_{-\infty}^{\infty} xf_X(x) dx = \int_1^3 \frac{1}{2} x dx = \left[\frac{1}{4} x^2 \right]_1^3 = \frac{1}{4} (3)^2 - \frac{1}{4} (1)^2 = 2$$

Check in MATLAB:

```
EX = int(f(x,y)*x, x, 1, 3)
```

```
EX = 2
```

```
% Alternative method
```

```
fun = @(x) 1/2*x;
integral(fun, 1, 3)
```

```
ans = 2
```

We compute $E[Y]$:

$$E[Y] = \int_{-\infty}^{\infty} y f_Y(y) dy = \int_1^2 y dy = \left[\frac{1}{2} y^2 \right]_1^2 = \frac{1}{2} (2)^2 - \frac{1}{2} = \frac{3}{2}$$

```
EY = int(fY(y)*y, y, 1, 2)
```

```
EY =
```

$$\frac{3}{2}$$

```
% Alternative method
```

```
fun = @(y) y;
integral(fun, 1, 2)
```

```
ans = 1.5000
```

We compute $E[XY]$:

$$E[XY] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} xy f_{X,Y}(x, y) dy dx$$

$$E[XY] = \int_1^3 \left(\int_1^2 \frac{1}{2} xy dy \right) dx$$

First we compute the inner integral:

$$\int_1^2 \frac{1}{2} xy dy = \left[\frac{1}{4} x y^2 \right]_1^2 = \frac{1}{4} x (2)^2 - \frac{1}{4} x = \frac{3}{4} x$$

```
inner = int(x*y*f(x,y), y, 1, 2)
```

```
inner =
```

$$\frac{3}{4} x$$

Next, we compute the outer integral:

$$\int_1^3 \frac{3}{4}x \, dx = \left[\frac{3}{8}x^2 \right]_1^3 = \frac{3}{8}(3)^2 - \frac{3}{8} = \frac{27}{8} - \frac{3}{8} = \frac{24}{8} = 3$$

```
EXY = int(inner, x, 1, 3)
```

EXY = 3

Two random variables are uncorrelated if the covariance $\text{cov}(X, Y) = E[XY] - E[X]E[Y] = 0$.

```
EXY - EX * EY
```

ans = 0

Since $\text{cov}(X, Y) = E[XY] - E[X]E[Y] = 0$ then X and Y are uncorrelated.

Exam 2018 Problem 5: Continuous Probability Density Function

Consider the following probability density function

$$f_X(x) = \begin{cases} \alpha x^2 + \frac{1}{4} & \text{for } -1 < x < 1 \\ 0 & \text{elsewhere} \end{cases}$$

where α is a real number.

[✓] 1) Find a valid probability density function

1. Determine α so that $f_X(x)$ is a valid probability density function.

A valid probability density function must be non-negative everywhere and the integral from minus infinity to positive infinity must equal 1

A valid probability density function is given by:

$$\int_{-\infty}^{\infty} f_X(x) \, dx = 1$$

We need to compute:

$$\int_{-1}^1 \alpha x^2 + \frac{1}{4} \, dx$$

For convenience and to avoid silly mistakes, use MATLAB:

```
syms x a  
int(a*x^2 + 1/4, x, -1, 1)
```

ans =

$$\frac{2a}{3} + \frac{1}{2}$$

Solve the equation for a in MATLAB:

```
solve(int(a*x^2 + 1/4, x, -1, 1) - 1)
```

ans =

$$\frac{3}{4}$$

Let us check the results:

```
int(3/4 * x^2 + 1/4, x, -1, 1)
```

ans = 1

For $f_X(x)$ to be a valid probability density function, a must be:

$$a = \frac{3}{4}$$

As a is positive, the probability density function is a upward curving parabola and therefore positive in its entire domain.

[✓] 2) Compute the probability given a probability density function?

2. What is the probability that $1/4 < X \leq 3/4$?

```
p = int(3/4 * x^2 + 1/4, x, 1/4, 3/4)
```

p =

$$\frac{29}{128}$$

```
vpa(p)
```

ans = 0.2265625

The answer is:

$$\Pr\left(\frac{1}{4} < X \leq \frac{3}{4}\right) = \frac{29}{128} = 0.2265625$$

[✓] 3) Compute the expected value a function:

Let a function be given by $g(x) = e^{-|x|}$.

3. Compute $E[g(x)]$.

We need to compute:

$$E[g(x)] = \int_{-\infty}^{\infty} g(x)f_X(x) dx = \int_{-1}^{1} e^{-|x|} \cdot \frac{3}{4}x^2 + \frac{1}{4} dx$$

```
syms x
f = exp(-abs(x)) * 3/4 * x^2 + 1/4
```

```
f =

$$\frac{3x^2e^{-|x|}}{4} + \frac{1}{4}$$

```

```
int(f , x, -1, 1 )
```

```
ans =

$$\frac{7}{2} - \frac{15e^{-1}}{2}$$

```

```
vpa(int(f , x, -1, 1 ))
```

```
ans = 0.74090419121418258803357172378904
```

The answer is:

$$E[g(x)] \approx 0.74$$

```
fxgx=@(x) (3/4*x.^2+1/4).*exp(-abs(x));
integral(fxgx,-1,1)
```

```
ans = 0.5570
```

Exam 2016 Problem 4: Discrete Random Process

Let two random variables, X_0 and X_1 be defined from a discrete time random process $X(n)$ as $X_0 = X(n)$ and $X_1 = X(n + 1)$. The joint density function for the process is given by

$$f_{X_0, X_1}(x_0, x_1) = \frac{1}{6}\delta(x_0 - 2)\delta(x_1 + 1) + \frac{2}{6}\delta(x_0 - 2)\delta(x_1 - 2) \\ + \frac{2}{6}\delta(x_0 + 1)\delta(x_1 + 1) + \frac{1}{6}\delta(x_0 + 1)\delta(x_1 - 2).$$

The given joint density function can put in a table where the green cells are the joint probabilities and the blue cells are the marginal probabilities:

	$x_1 = -1$	$x_1 = 2$	
$x_0 = -1$	$\frac{2}{6}$	$\frac{1}{6}$	$P(X_0 = -1) = \frac{3}{6}$
$x_0 = 2$	$\frac{1}{6}$	$\frac{2}{6}$	$P(X_0 = 2) = \frac{3}{6}$
	$P(X_1 = -1) = \frac{3}{6}$	$P(X_1 = 2) = \frac{3}{6}$	

```
clear variables;
```

1) Sketch a realization of a signal with this density function.

From the joint density function we observe that the signal can take on the two values; +2 or -1.

1. If the signal is +2 at t_0 then it will go to -1 at $t_0 + 1$ with probability $\frac{1}{6}$
2. If the signal is +2 at t_0 then it will remain -2 at $t_0 + 1$ with probability $\frac{2}{6}$
3. If the signal is -1 at t_0 then it will remain -1 at $t_0 + 1$ with probability $\frac{2}{6}$
4. If the signal is -1 at t_0 then it will go to +2 at $t_0 + 1$ with probability $\frac{1}{6}$

At given time, the value of the signal will remain with probability $\frac{2}{3}$,

A realisation coded in MATLAB:

```
pos_val = 2;
neg_val = -1;
change_proba = 2/3; % Probability for change

N = 50;
x = zeros(N,1);

if rand(1) < 0.5
    x(1)=pos_val;
```

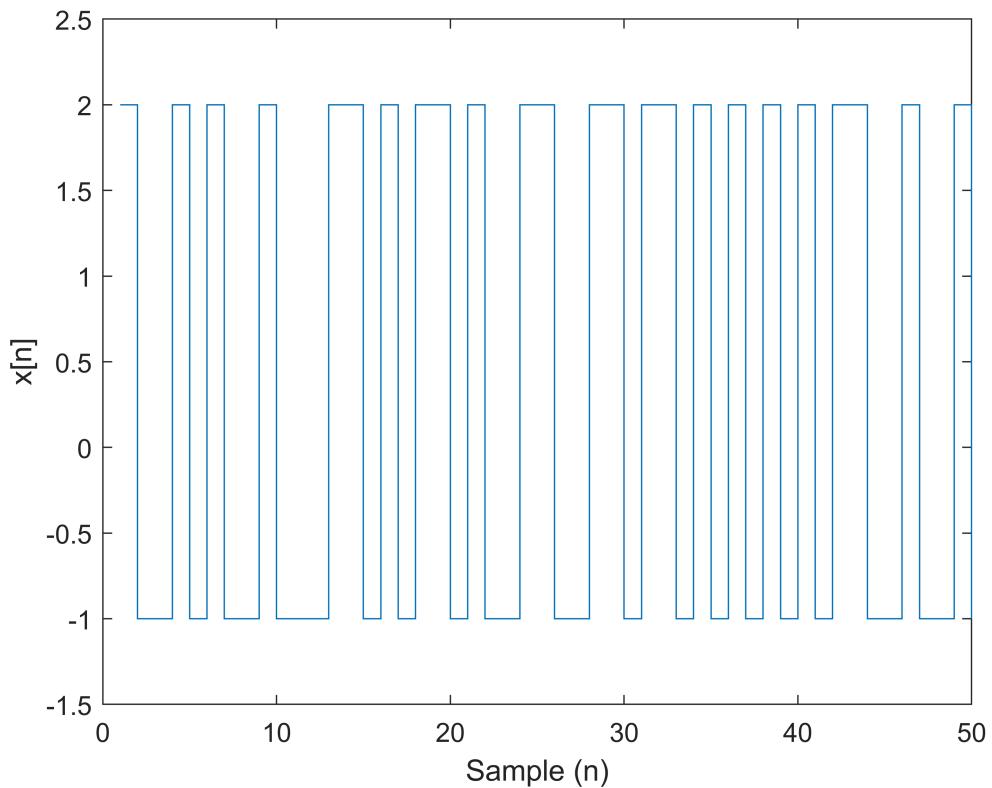
```

else
    x(1)=neg_val;
end

for n=2:N
    if rand(1) < change_proba
        if x(n-1) == pos_val
            x(n) = neg_val;
        else
            x(n) = pos_val;
        end
    else
        x(n)=x(n-1);
    end
end

stairs(x)
ylim([neg_val - 0.5, pos_val + 0.5])
xlabel('Sample (n)')
ylabel('x[n]')

```



2) Compute marginal density function

2. Show that the marginal density function for X_0 is $f_{X_0}(x_0) = \frac{1}{2}\delta(x_0 - 2) + \frac{1}{2}\delta(x_0 + 1)$ and calculate the mean value of the signal.

The given joint density function can put in a table where the green cells are the joint probabilities and the blue cells are the marginal probabilities:

	$x_1 = -1$	$x_1 = 2$	
$x_0 = -1$	$\frac{2}{6}$	$\frac{1}{6}$	$P(X_0 = -1) = \frac{3}{6}$
$x_0 = 2$	$\frac{1}{6}$	$\frac{2}{6}$	$P(X_0 = 2) = \frac{3}{6}$
	$P(X_1 = -1) = \frac{3}{6}$	$P(X_1 = 2) = \frac{3}{6}$	

The marginal density function is given as:

$$f_{X_0}(x_0) = \int_{-\infty}^{\infty} f_{X_0, X_1}(x_0, x_1) dx_1$$

Since the random variables are discrete, the above integral amounts to summing out X_1 . Doing that we get the required margin mass function:

$$f_{X_0}(x_0) = \frac{3}{6} \delta(x_0 - 2) + \frac{3}{6} \delta(x_0 + 1)$$

Alternatively, we can compute:

$$\begin{aligned} f_{X_0}(x_0) &= \int_{-\infty}^{\infty} f_{X_0, X_1}(x_0, x_1) dx_1 \\ &= \int_{-\infty}^{\infty} \left(\frac{1}{6} \delta(x_0 - 2) \delta(x_1 + 1) + \frac{2}{6} \delta(x_0 - 2) \delta(x_1 - 2) \right. \\ &\quad \left. + \frac{2}{6} \delta(x_0 + 1) \delta(x_1 + 1) + \frac{1}{6} \delta(x_0 + 1) \delta(x_1 - 2) \right) dx_1 \\ &= \frac{1}{6} \delta(x_0 - 2) + \frac{2}{6} \delta(x_0 - 2) + \frac{2}{6} \delta(x_0 + 1) + \frac{1}{6} \delta(x_0 + 1) \\ &= \frac{1}{2} \delta(x_0 - 2) + \frac{1}{2} \delta(x_0 + 1). \end{aligned}$$

The expected value of a discrete random variable X with outcomes x_1, x_2, \dots, x_k is given by:

$$E(X) = \sum_{i=1}^k x_i P(X = x_i)$$

In this problem, we have

$$E(X_0) = -1 \cdot P(X_0 = -1) + 2 \cdot P(X_0 = 2) = -1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2} = \frac{1}{2}$$

Alternative way:

$$\begin{aligned}
E[X_0] &= \int_{-\infty}^{\infty} x_0 f_{X_0}(x_0) dx_0 \\
&= \int_{-\infty}^{\infty} x_0 \left(\frac{1}{2}\delta(x_0 - 2) + \frac{1}{2}\delta(x_0 + 1) \right) dx_0 \\
&= 2\frac{1}{2} + (-1)\frac{1}{2} = \frac{1}{2}.
\end{aligned}$$

3) Determine whether signal is deterministic or random by calculating the correlation.

3. Calculate the correlation $E[X_0 X_1]$ and use this correlation to classify the signal as deterministic or random.

The correlation can be calculated as:

$$\begin{aligned}
E[X_0 X_1] &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_0 x_1 f_{X_0, X_1}(x_0, x_1) dx_0 dx_1 \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_0 x_1 \left(\frac{1}{6}\delta(x_0 - 2)\delta(x_1 + 1) + \frac{2}{6}\delta(x_0 - 2)\delta(x_1 - 2) \right. \\
&\quad \left. + \frac{2}{6}\delta(x_0 + 1)\delta(x_1 + 1) + \frac{1}{6}\delta(x_0 + 1)\delta(x_1 - 2) \right) dx_0 dx_1 \\
&= \frac{1}{6}(2)(-1) + \frac{2}{6}(2)(2) + \frac{2}{6}(-1)(-1) + \frac{1}{6}(-1)(2) \\
&= 1.
\end{aligned}$$

Since there is a non-zero correlation between the signal at time n and $n + 1$ this means that the signal is predictable. The signal is only partially and not perfectly predictable as evident from the above discussion, i.e. the signal is twice as likely to stay at the same value as to switch value. The full classification is thus **partly deterministic**.

Functions

```

function [xo, px]=epdf(x, bins)
[nx,xo]=hist(x,bins);
dx=xo(3)-xo(2);
px=nx/(dx*length(x));

```

end

Power Spectral Density

Table of Contents

Power Spectral Density.....	2
Definition.....	2
What is the average power of a stable LTI system?.....	2
Example: Power Spectrum Analyser.....	3
PSDs of Some Random Processes.....	4
White Noise.....	4
Random Sinusoid.....	4
Coloured Noise.....	4
PSD Estimation.....	5
Non-parametric Methods.....	5
Periodogram.....	6
Modified Periodogram.....	7
Blackman–Tukey Method.....	8
Welch's method.....	8
Spectral Leakage.....	10
Parametric Methods.....	12
Parametric Model: ARMA(p, q).....	12
Parametric Model: Sinusoids with Additive Noise.....	13
Non-parametric Methods vs Parametric Methods.....	13
Coherence Function.....	13
Cross Power Spectrum Density.....	13
Definition of the Coherence Function.....	14
Coherence Estimates in MATLAB.....	14
High-Pass IIR filter.....	14
High-pass filter with noise at the output.....	17
High-pass filter with noise at the input.....	19
Nonlinear filter.....	20
Nonlinear filter via clipping.....	22
Comb Filter.....	23
Problems.....	27
Can Leakage be reduced in estimates of PSD by applying a window function to the data?.....	27
Estimate PSD from an ACRS assuming MA(1) process.....	28
Estimate PSD from an ACRS assuming MA(2) process.....	29
Quiz: Determine if two sinusoids have the same PSD?.....	30
Quiz: How do we measure if a system is LTI?.....	31
[✓] Problem 14.57: Compare different PSD estimates using noise recorded on F-16.....	32
1) Compute and plot the ACRS estimate of the noise process.....	33
2) Estimate model parameters for an AR(2) and AR(4) models.....	34
3) Compute and plot the PSD estimate using the AR(2) and AR(4) models.....	37
4) Compute and plot the periodogram PSD estimate of the noise process.....	39
5) Compute the Bartlett PSD estimate.....	40
6) Compute the Blackman–Tukey PSD estimate.....	41
7) Compute the Welch PSD estimate.....	42
8) Compare the plots in the above four parts and comment on your observation.....	43
[✓] Problem 14.42: Compare periodogram, modified periodogram and Blackman-Tukey methods in terms of signal resolution.....	43
1) Estimate the PSD using the periodogram and plot the spectrum.....	44
2) Estimate the PSD using the modified periodogram with Bartlett window.....	45
3) Estimate the PSD using the Blackman–Tukey method.....	46
4) Which method performs best in terms of signal resolution?.....	47
ADSI Problem 4.7: Coherence function.....	48
1) What happens to the coherence function if gain is increased from 1 to 2?.....	48

2) What happens to the coherence function if a delay happens in the measurement?.....	49
[✓] ADSI Problem 4.21: Minimum variance spectral estimation.....	50
1) Create a stable signal model.....	50
2) Use the signal model to plot the true spectrum.....	51
3) Create a realization of the signal and determine the autocorrelation matrix Rx in an appropriate size.....	52
4) Calculate the spectrum using Eq. (14.4.12) from the note.....	54
5) Calculate the optimum filters.....	55
Problem 1: PSD from ACRS assuming AR(2) process.....	56
1) Compute AR(2) model coefficient given autocorrelation sequence.....	57
Functions.....	59

Power Spectral Density

Definition

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.

INSIGHT: 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:

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

What is the average power of a stable LTI system?

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

The average power 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 $r_y(0)$

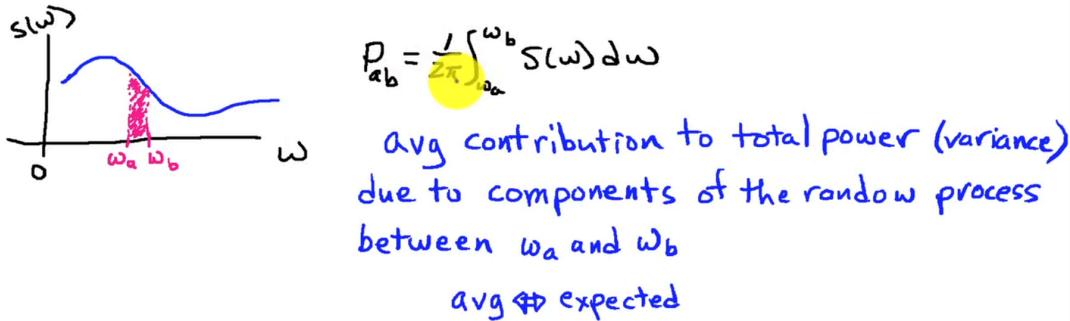
$$r_y(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{xx}(\omega) e^{j\omega \cdot 0} d\omega$$

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

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.

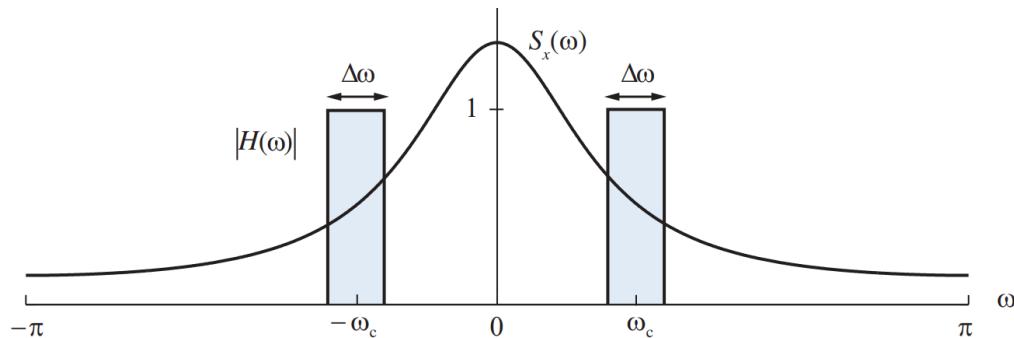


Figure 13.13 Physical interpretation of power spectrum density as power at the output of a narrowband LTI system.

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

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

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

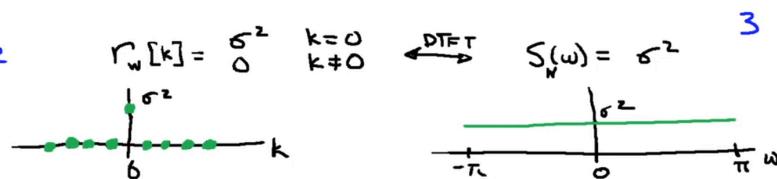
Since the spectrum is symmetric around zero, we get power contribution from both sides of zero. This is called a double-sided spectrum. Therefore, we have $2 \frac{1}{2\pi}$ in the equation above.

PSDs of Some Random Processes

White Noise

Examples

1) White noise



The power spectral density of white noise 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, $E[A^2] = 0, E[A^2] = \sigma_A^2$

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



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

$$r_c[k] = \begin{cases} 1 & k=0 \\ \nu_2 & k=\pm 1 \\ 0 & \text{otherwise} \end{cases}$$

DTFT $S_c(\omega) = 1 + \cos(\omega)$

PSD Estimation

Since we cannot have a mathematical model of random signals instead we use a statistical model. The Power Spectrum Density describes the variance (or power) of a random signal as a function of frequency. PSD can tell us where the energy is distributed.

Typically, we can estimate the power spectrum given a set of data. There are two main approaches for spectrum estimation:

- a) Non-parametric Methods: do not assume that a particular model generated the data
- b) Parameteric Methods: assume that the data is generated in a certain way e.g. by an AR(q) process

INSIGHT: There are various ways to compute PSD estimates. Now the question is, which PSD estimate should we believe in? Doing PSD estimation is a little bit like black art. It is really difficult to estimate PSD without knowing something about the process that generated the signal. Are some of the bumps in the PSD caused by the engine or the turbulence? In practice, we would record the sound from multiple places. This allows us to compare PSD estimates and compare energy.

Non-parametric Methods

a) Non-parametric Methods: do not assume that a particular model generated the data. In this case, we just use the FFT. Given a finite set of N observations, the general approach for the non-parametric methods for power spectrum estimation of a random signal is:

1. Window the observed signal $y(n)$

$$y_w(n) = w(n)y(n), 0 \leq n \leq N - 1$$

2. Take the Fourier Transform:

$$y_w(n) \leftrightarrow X_w(e^{j\omega})$$

3. Estimate the power spectrum

$$\hat{S}_{xx}(\omega) = \frac{1}{N \cdot F} |X_w(e^{j\omega})|^2 \text{ where } F \text{ is constant that depends on the window}$$

There are several techniques under the umbrella of non-parametric PSD estimation:

- Periodogram
- Bartlett's method
- Welch's method

Periodogram

A periodogram is a method for estimating the Power Spectrum Density of a random signal from a finite set of N observations given by:

$$I(\omega) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] e^{-j\omega n} \right|^2 \quad (14.34)$$

The periodogram uses a rectangular window $w(n) = 1$.

Mean

Mean of the periodogram Taking the mathematical expectation of (14.33) and using (14.26) (recall that $\hat{c}[\ell] = \hat{r}[\ell]$ for $m = 0$), we obtain the relation

$$E[I(\omega)] = \sum_{\ell=-(N-1)}^{N-1} E(\hat{r}[\ell]) e^{-j\omega\ell} = \sum_{\ell=-(N-1)}^{N-1} \left(1 - \frac{|\ell|}{N}\right) r[\ell] e^{-j\omega\ell}. \quad (14.40)$$

Since $E[I(\omega)] \neq S(\omega)$, the periodogram is a *biased* estimator of $S(\omega)$. However, for each $|\ell|$, the factor $(1 - |\ell|/N) \rightarrow 1$ as $N \rightarrow \infty$. Hence, we have

$$\lim_{N \rightarrow \infty} E[I(\omega)] = S(\omega), \quad (14.41)$$

Eq. (14.41) says that as the window length N increases towards infinity, the mean of the periodogram converges to the true power spectrum. Asymptotically (as N increases) the periodogram is an unbiased estimator of $S(\omega)$.

For finite N , it has some bias which is reflected by the window function.

Variance

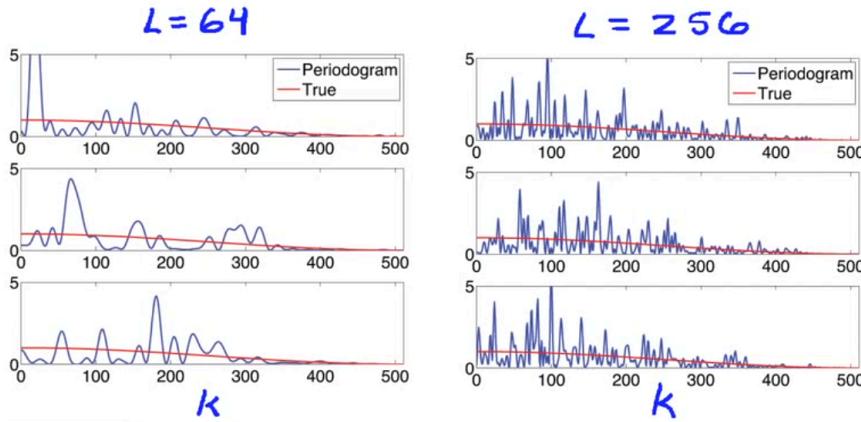
Under a variety of assumptions which holds for most random processes of practical interest, the variance of the periodogram estimator behaves like the square of the true PSD:

$$\text{var}[I(\omega)] \approx S^2(\omega), \quad 0 < \omega < \pi \quad (14.50)$$

The main implication of (14.50) is that the **periodogram is not a good estimator of PSD** because, independently of N , the standard deviation of the estimator is as large as the quantity to be estimated.

In other words, the periodogram is not a consistent estimator; that is, its distribution does not tend to cluster more closely around the true PSD as N increases.

Thus the definition of the PSD by $S(\omega) = \lim_{N \rightarrow \infty} I(\omega)$ is not valid because even if $\lim_{N \rightarrow \infty} E[I(\omega)] = S(\omega)$, the variance of $I(\omega)$ does not tend to zero as $N \rightarrow \infty$.



The variance is high at lower frequencies where the true power spectrum is larger. As the true spectrum goes to zero, the variance gets smaller. This is in agreement with (14.50). If we get better resolution i.e., L goes from 64 to 256, the behaviour is still present.

Covariance

The covariance between values of the periodogram at harmonically spaced frequencies is given by

$$\text{cov}[I(2\pi k_1/N), I(2\pi k_2/N)] \approx 0, \quad k_1 \neq k_2 \quad (14.51)$$

which states that closely spaced values of the periodogram are uncorrelated.

Modified Periodogram

The term modified periodogram is used to denote a periodogram where the window $w(n)$ is explicitly given:

$$\tilde{I}(\omega) \triangleq \frac{1}{N} \left| \sum_{n=0}^{N-1} w[n]x[n]e^{-j\omega n} \right|^2 \quad (14.52)$$

were $w[n]$ is a data window of size N .

The purpose of using a data window is to **reduce the spectral leakage** caused by strong narrowband components by lowering the level of sidelobes.

The data window must be normalised to ensure that the periodogram of $x[n]$ is asymptotically unbiased :

$$\sum_{n=0}^{N-1} w^2(n) = N. \quad (14.60)$$

```
N = 64;
w = bartlett(N);
w = w / (norm(w)/sqrt(N));
sum(w.^2)
```

ans = 64.0000

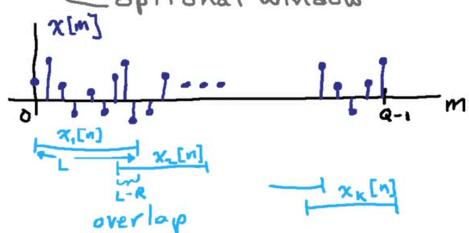
Blackman–Tukey Method

Welch's method

- average K independent random variables: reduce variance by factor of K

- Divide $x[n]$, $n=0, 1, \dots, Q-1$ into (possibly) overlapping segments of length L . r^{th} segment:

$$x_r[n] = x[rL+n], n=0, 1, \dots, L-1, r=1, 2, \dots, K$$

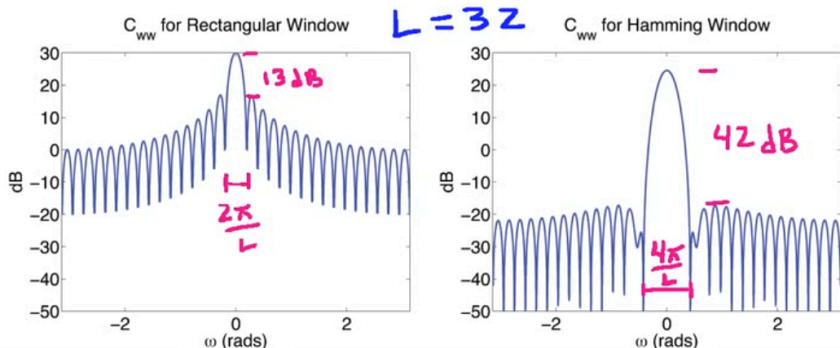


- Average periodograms from each segment

$$I_r(\omega) = \frac{1}{2\pi} \sum_{n=0}^{L-1} w[n] \left| \sum_{r=0}^{k-1} x_r[n] e^{-j\omega n} \right|^2$$

$$\hat{S}_{xx}^{\text{av}}(\omega) = \frac{1}{K} \sum_{k=1}^K I_r(\omega)$$

Variance: $\text{Var}\{\hat{S}_{xx}^{\text{per}}(\omega)\} \geq \text{Var}\{\hat{S}_{xx}^{\text{av}}(\omega)\} \geq \frac{1}{K} \text{Var}\{\hat{S}_{xx}^{\text{per}}(\omega)\}$
1/K reduction if $I_r(\omega)$ are independent



The nature of bias depends on the chosen window. There is a trade-off between main lobe width and side-lobe height.

- Rectangular window gives you the narrowest main lobe but a fairly large side-lobe height.
- Hamming window gives you twice the main lobe width but the side-lobes are much lower.

Hamming window sacrifices some resolution for increased dynamic range.

Dynamic range is the ratio of the largest signal amplitude and the smallest signal amplitude. How well can we realible distinguish the largest from the smallest.

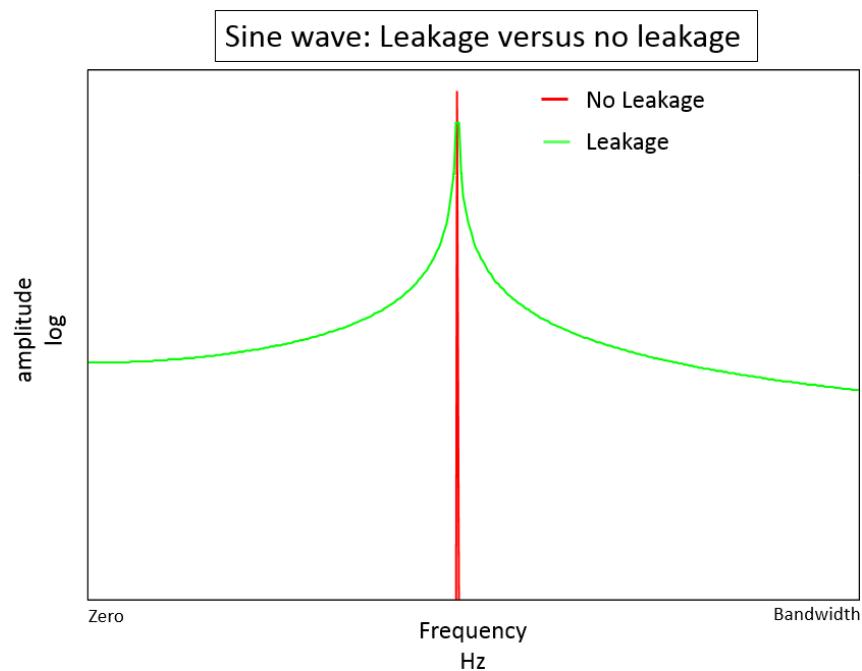
Windows are used to manage bias.

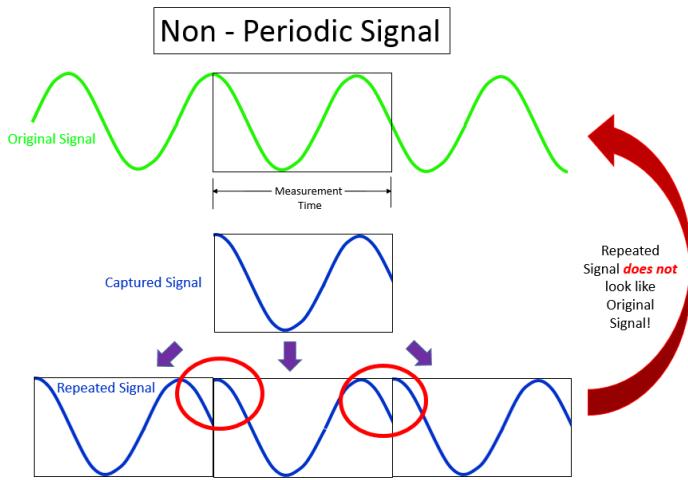
Typically, we will trade resolution (main-lobe width) for dynamic range (side-lobe height)

Spectral Leakage

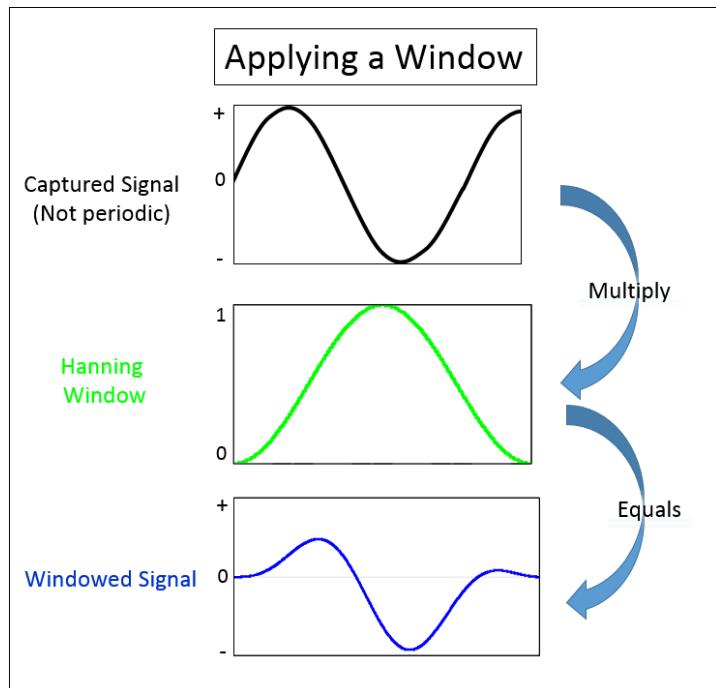
Leakage is an artefact of an FFT applied to a non-periodic data. Even if a signal is periodic like a sinusoid, when the signal is measured with some interval it can be non-periodic. The issue is that FFT assumes that the signal is periodic and repeats itself after the measured interval. This is not the case when the data is non-periodic, which contains sharp transitions at the end of each measured interval. These sharp changes have a broad frequency response which lead to spectral leakage. Windows can help **minimize** the effects of leakage by smoothing the time domain signal, but cannot eliminate leakage.

A signal with leakage (green in *Figure 2*) has lower amplitude and a broader frequency response than a signal with no leakage (red in *Figure 2*). This makes it difficult to quantify the signal properly in the frequency domain.



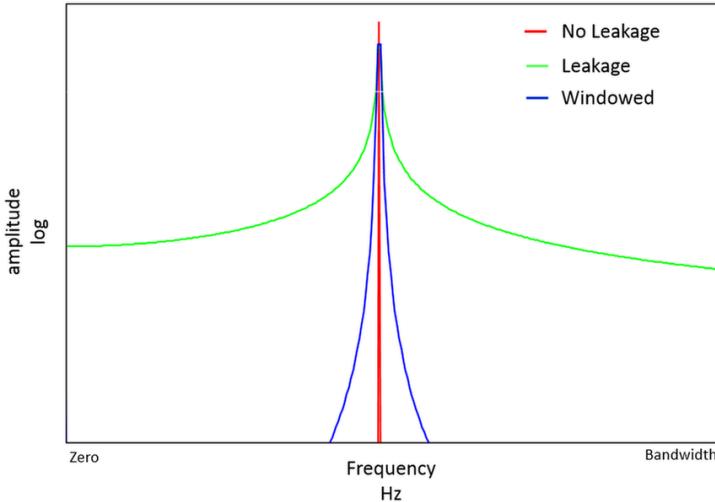


To reduce leakage, a mathematical function called a window is applied to the data. Windows are designed to reduce the sharp transient in the re-created signal as much as possible.



The main benefit of windowing is that the leakage is now confined over a smaller frequency range, instead of affecting the entire frequency bandwidth of the measurement.

Sine wave: No Leakage, Leakage, Windowed



Parametric Methods

b) Parametric Methods: assume that the data is generated in a certain way. The general approach is:

1. Measure and get some data $x(n)$
2. Use the data to estimate the model parameters e.g. ARMA coefficients
3. Express the estimated PSD $\hat{S}_{xx}(\omega)$ as a function of the model parameters.

Parametric Model: ARMA(p, q)

For example, we might assume that the data is the output of the LTI system with frequency response $H(e^{j\omega})$ given white noise signal as input:

$$\begin{array}{ccc}
 \text{white noise } w[n] & \xrightarrow{\text{LTI system }} & x[n] \\
 \xrightarrow{\quad H(e^{j\omega}) \quad} & & \Rightarrow S_{xx}(\omega) = S_{ww}(\omega) |H(e^{j\omega})|^2 \\
 & & = \sigma^2 |H(e^{j\omega})|^2
 \end{array}$$

There are several different models that can be used to estimate the PSD.

- AR(q) model, autoregressive model, all-pole model, only poles in its frequency response.
- MA(p) model, moving average, only zeros in its frequency response.
- ARMA(q, p) model

Whether a model has poles, zeros or pole-zeros results in different characteristic of the PSD estimate.

Parametric Model: Sinusoids with Additive Noise

Another example of a parametric method is Sinusoid in Noise model. Here, the model assumes that the data that we have measured $x[n]$ consists of combination of multiple sinusoids with additive noise $w(n)$. With this model, the PSD is:

Sinusoids in noise model

$$x[n] = \sum_{i=1}^L A_i e^{j\omega_i n} + w[n] \Rightarrow S_{xx}(\omega) = \sum_{i=1}^L \sigma_i^2 S(\omega - \omega_i) + \sigma_w^2$$

unknown: $\sigma_i^2, \omega_i, \sigma_w^2$

If we know the parameters σ_i^2, ω_i and σ_w^2 then we our PSD.

Non-parametric Methods vs Parameteric Methods

Comparison

If model is correct -

- High quality spectrum estimates
- Significantly less data required

If model is wrong -

- Parametric can give wrong/misleading estimates

.

Coherence Function

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 signal are not in phase at ω_0 .

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

Definition of the 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:

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

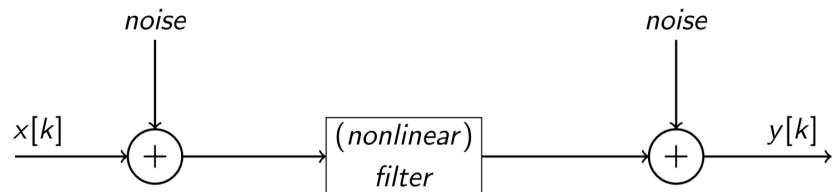
where $0 \leq |\gamma_{yx}(\omega)|^2 \leq 1$ i.e., the function yields a number between zero and one.

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

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

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



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

- 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

```
clear variables;
```

High-Pass IIR filter

```

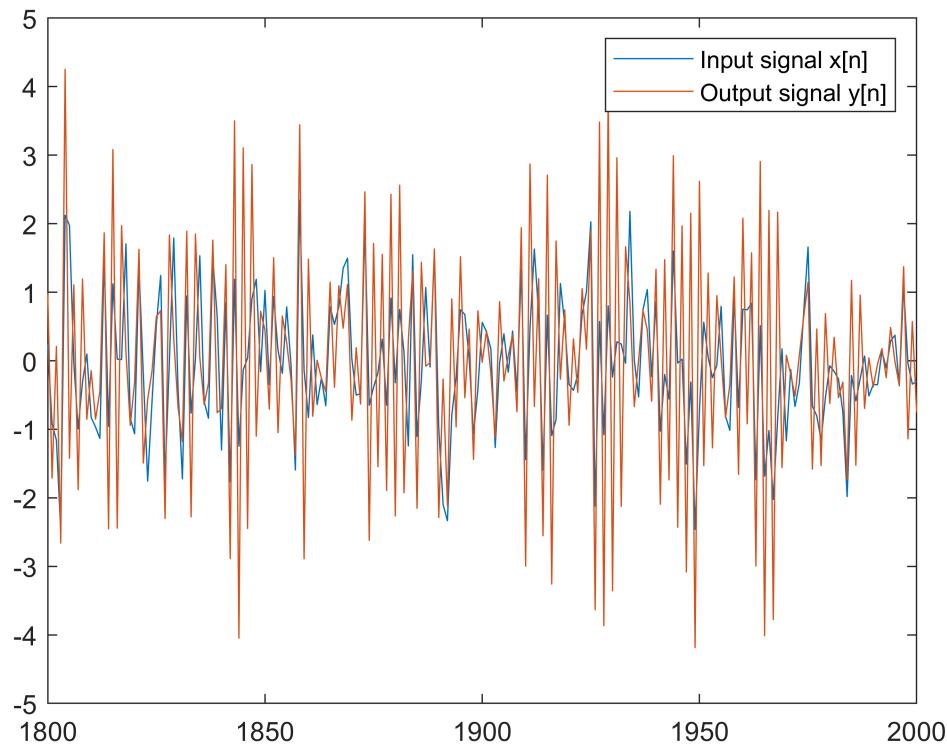
N = 2000;
n = 1800:N;

% 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]')

```

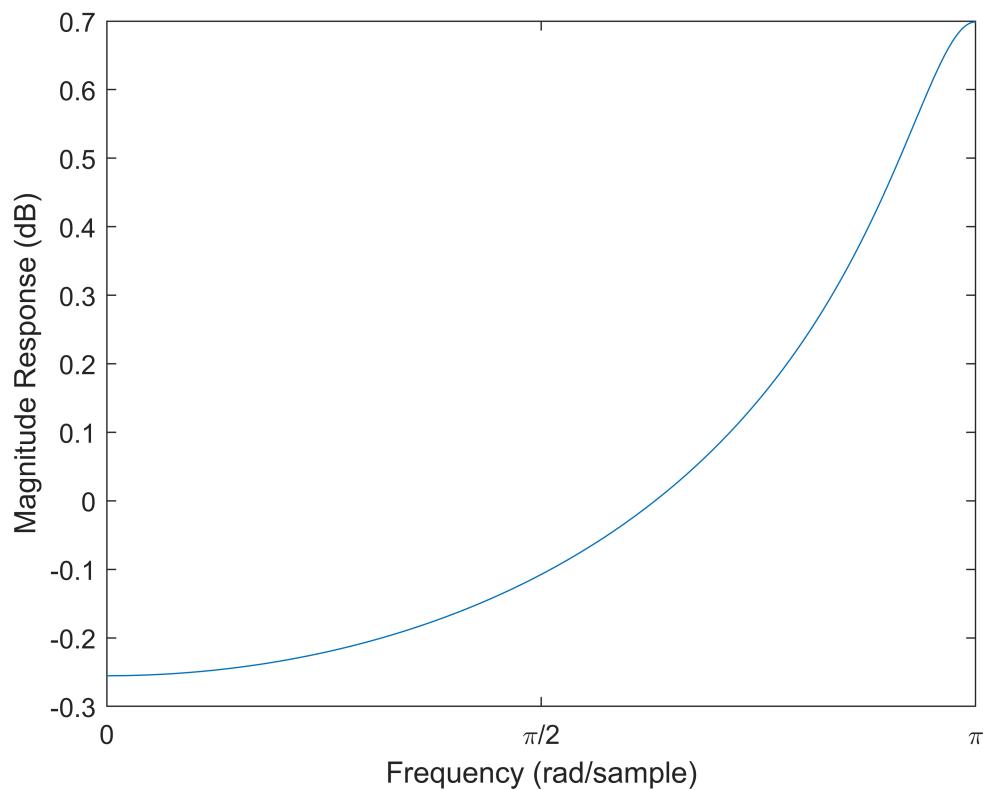


We can plot the magnitude response to see which frequencies the filter is attenuating and which frequency components it is amplifying:

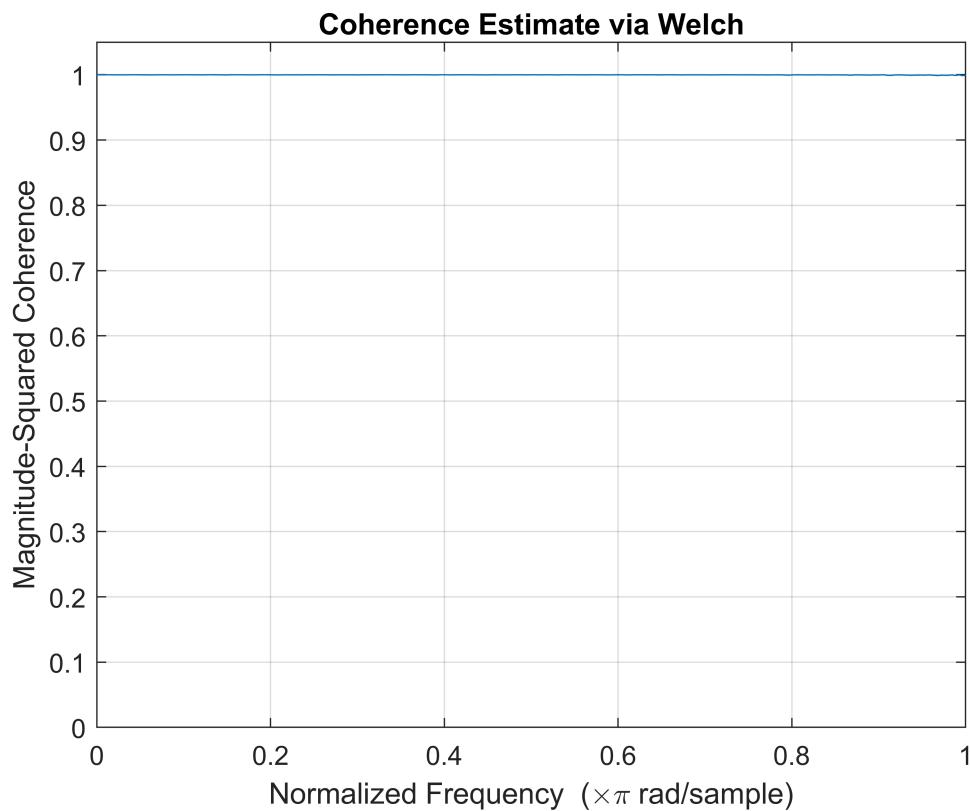
```

[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]);
```

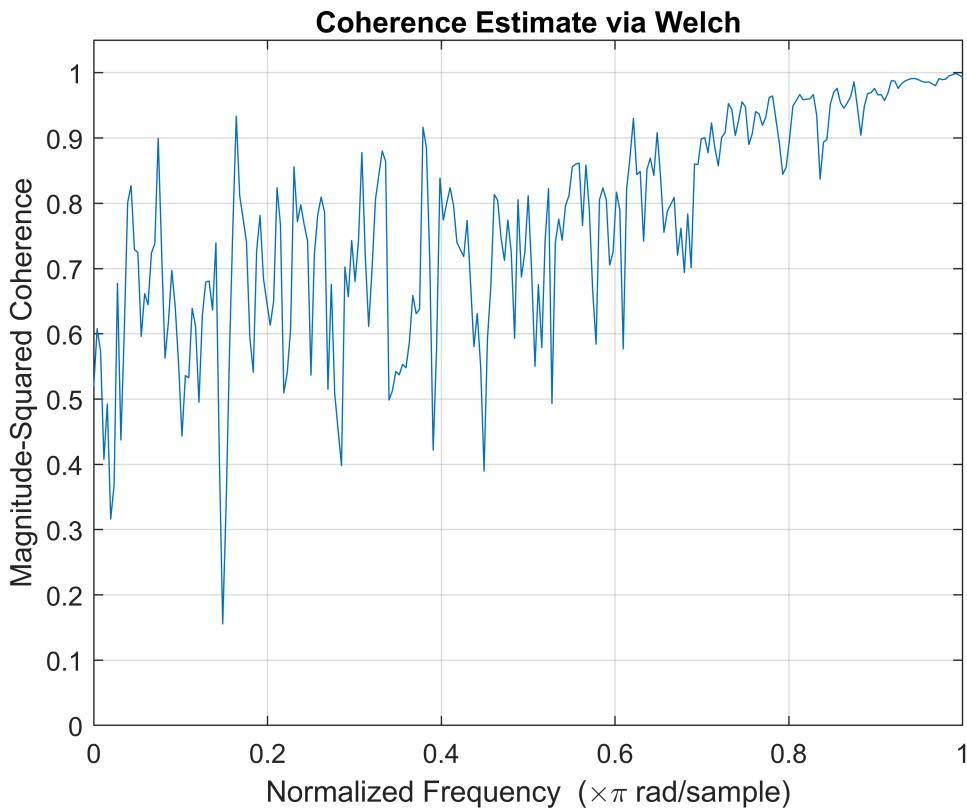


Notice that the estimate coherence function is 1. This is not a surprise because we are sending the input signal through a LTI system.

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.

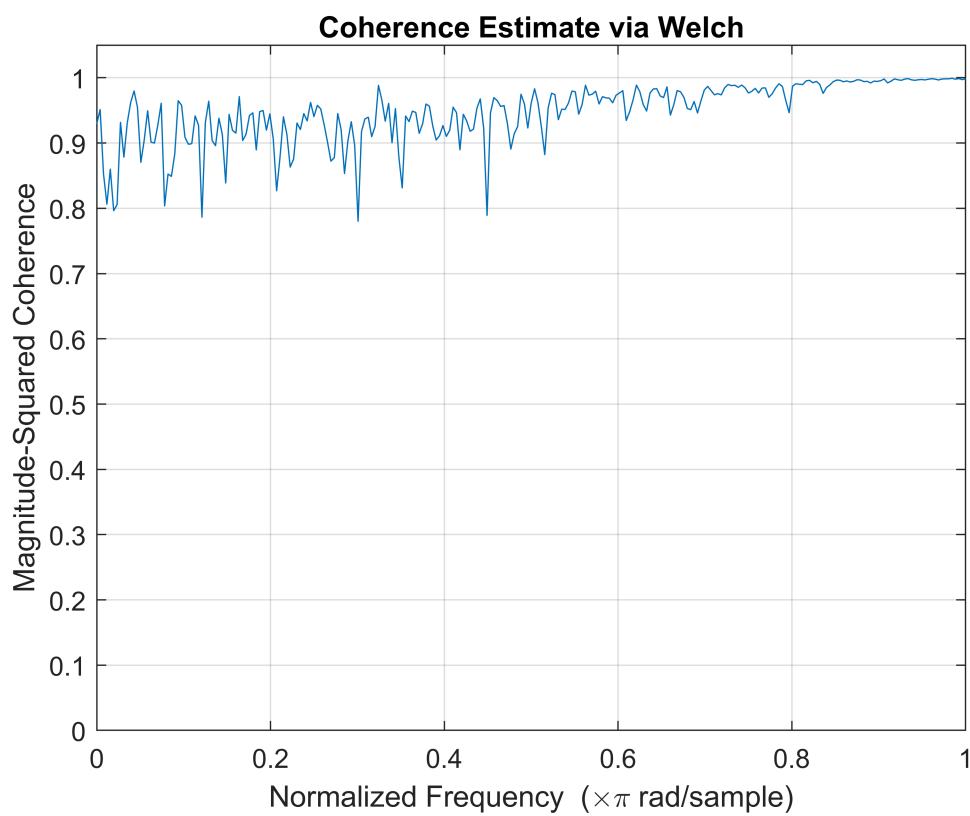
If the noise is reduced, we get close to 1:

```

noise = 0.2;
y = filter(b, a, x);
y = y + noise*randn(N, 1);

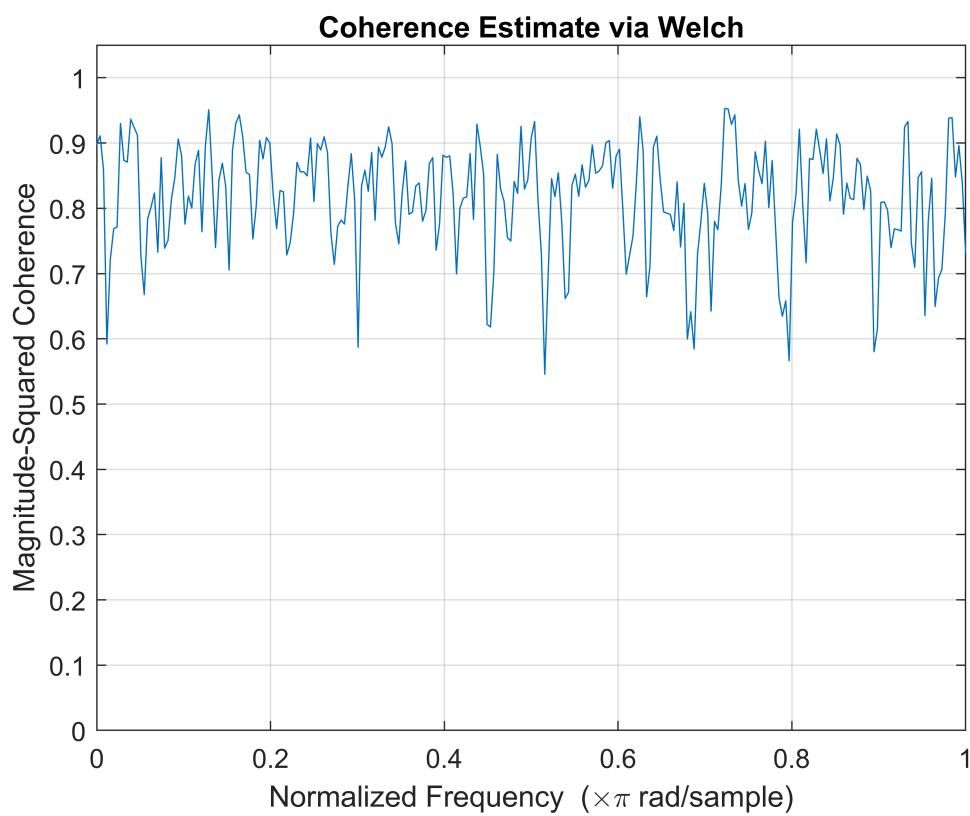
% Compute and plot the coherence estimate
mscohere(y, x);
ylim([0, 1.05]);

```



High-pass filter with noise at the input

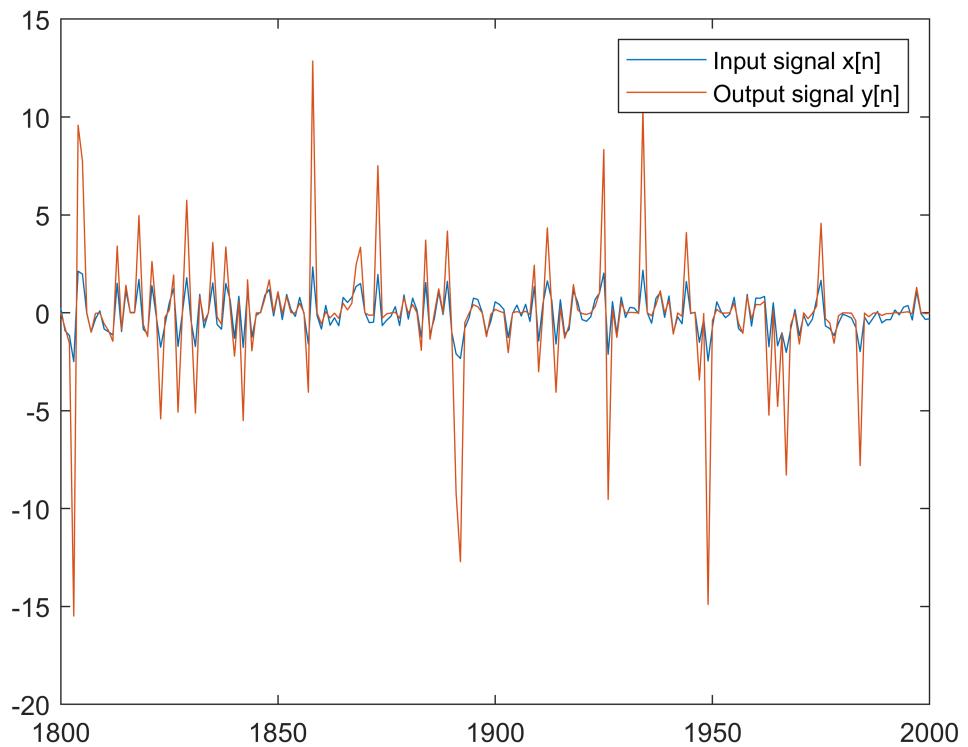
```
noise = 0.5;
y = filter(b, a, x + noise*randn(N, 1));
mscohere(y, x);
ylim([0, 1.05]);
```



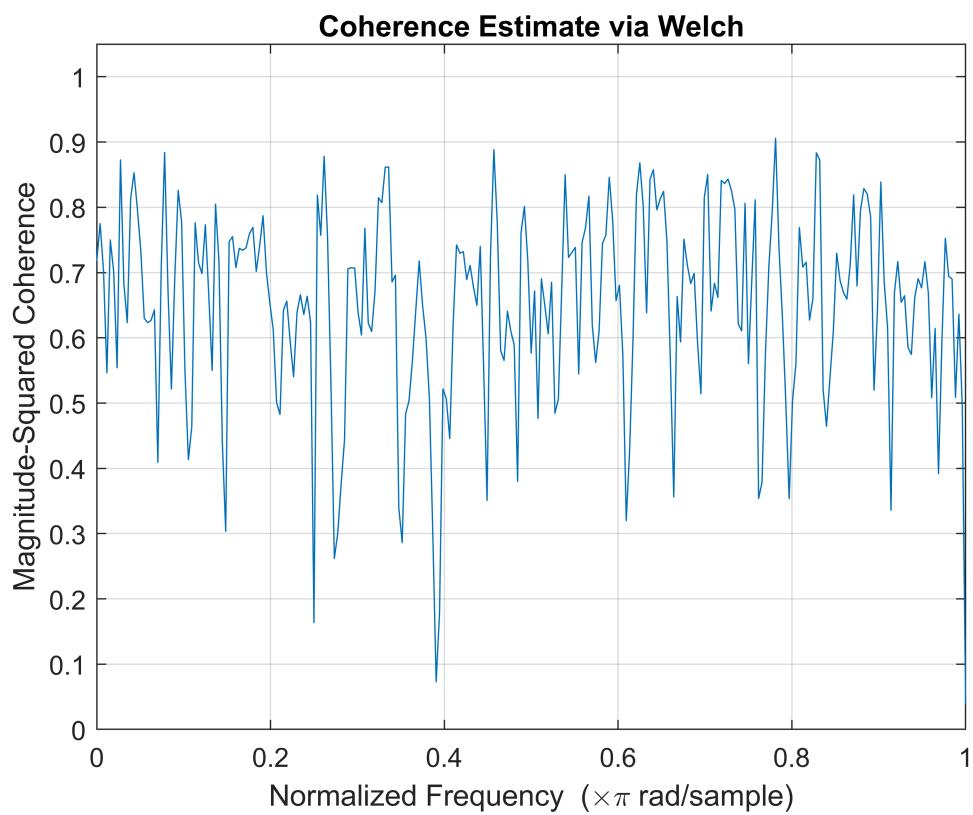
The additional noise is also being filtered so we have equal amount.

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]')
```



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



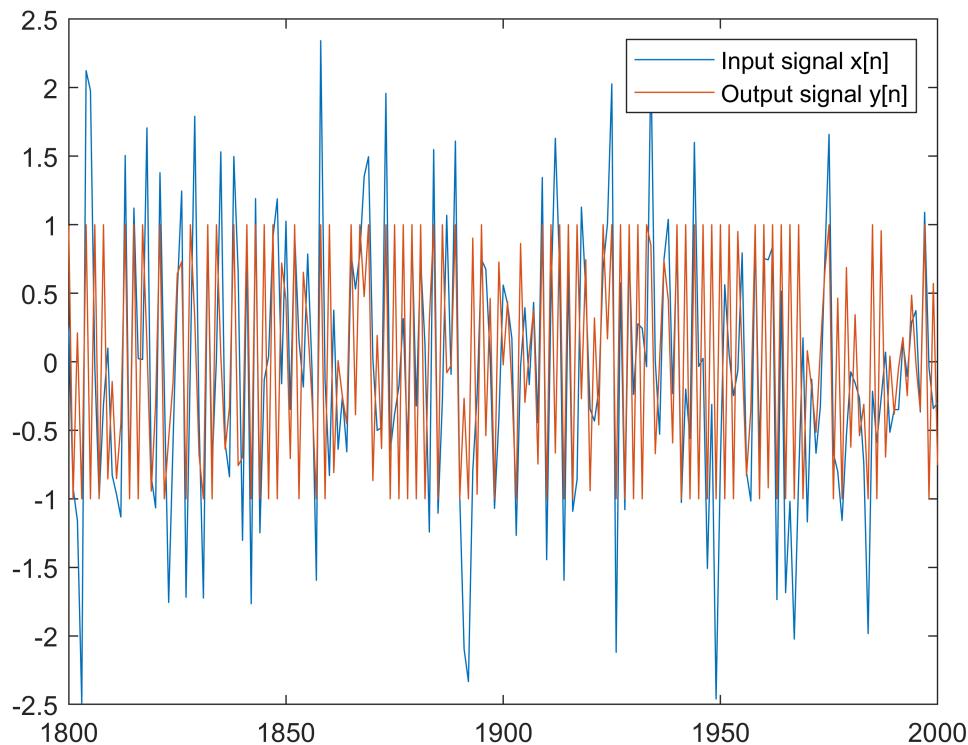
The coherence is all over the place.

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.

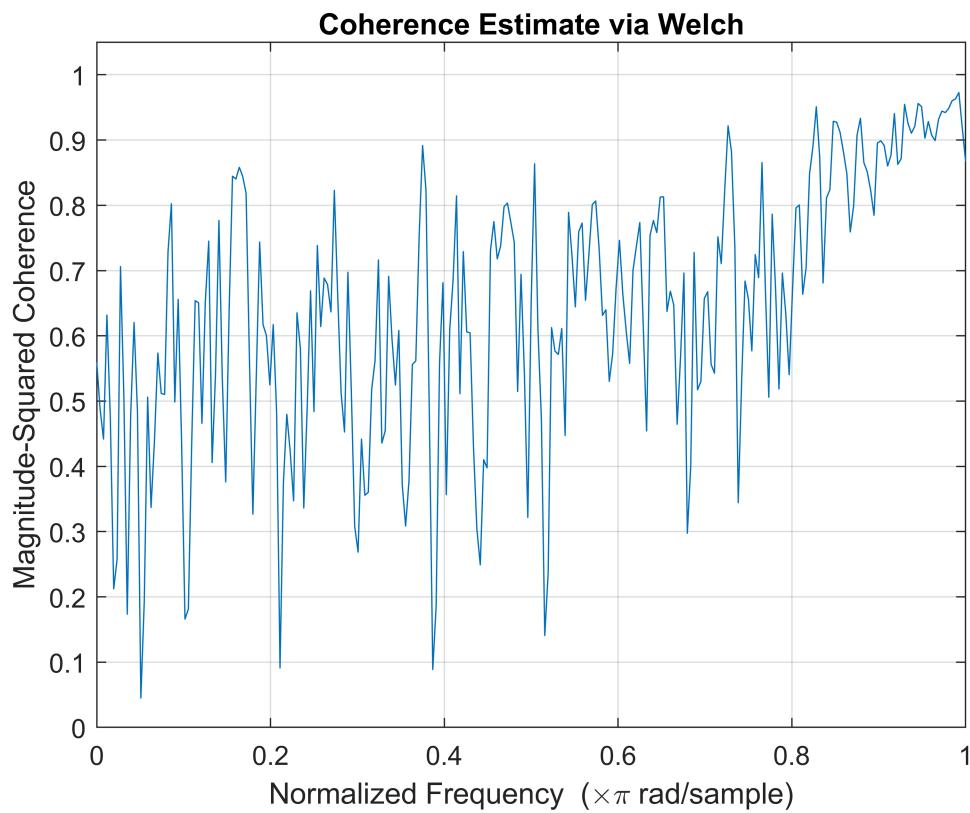
In practice, the coherence function is close to 1. The difference $1 - \text{mscohore}(y, x)$ tells us how well we can remove noise.

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]')
```



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



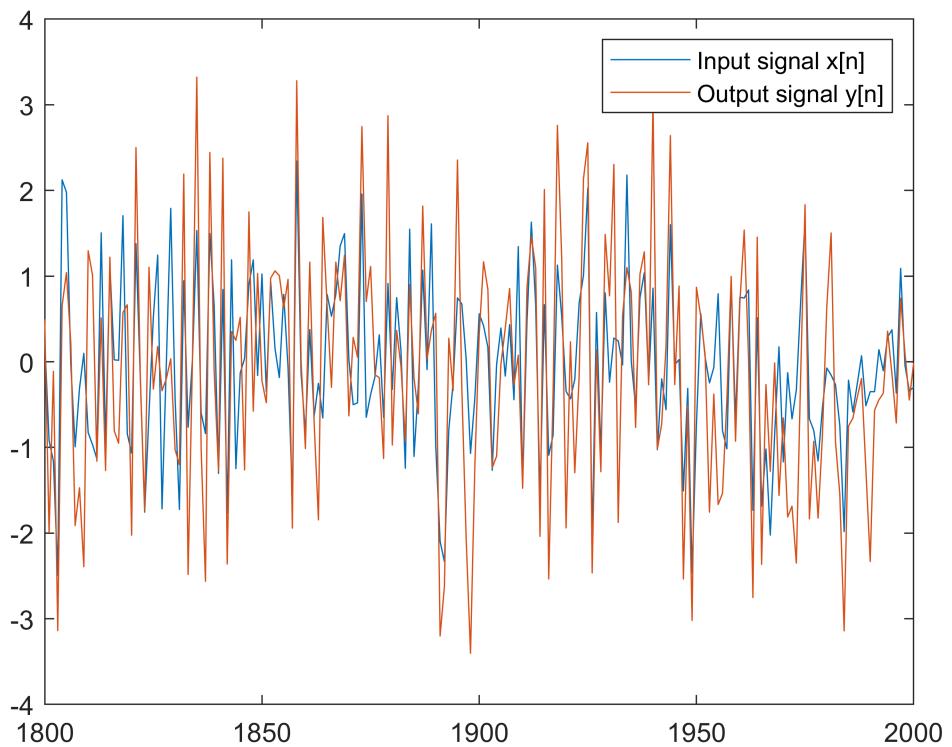
We see more coherence in the high frequencies than the low frequencies. Research why?

Comb Filter

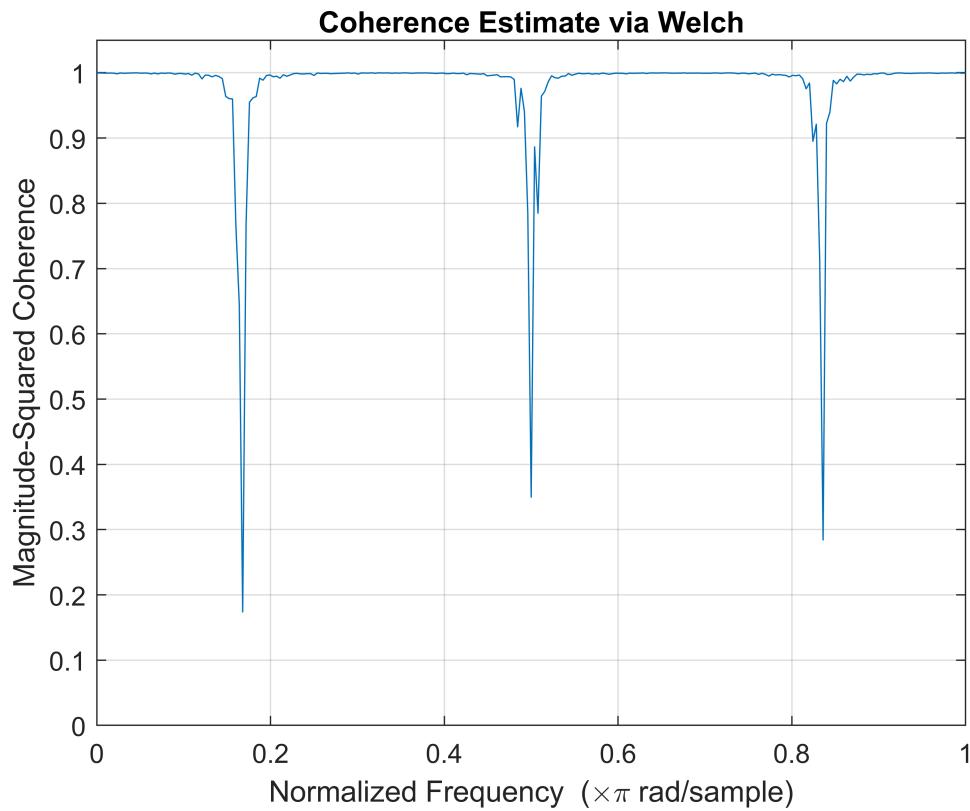
Let us define a comb filter which has the difference equation:

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

```
% 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]')
```



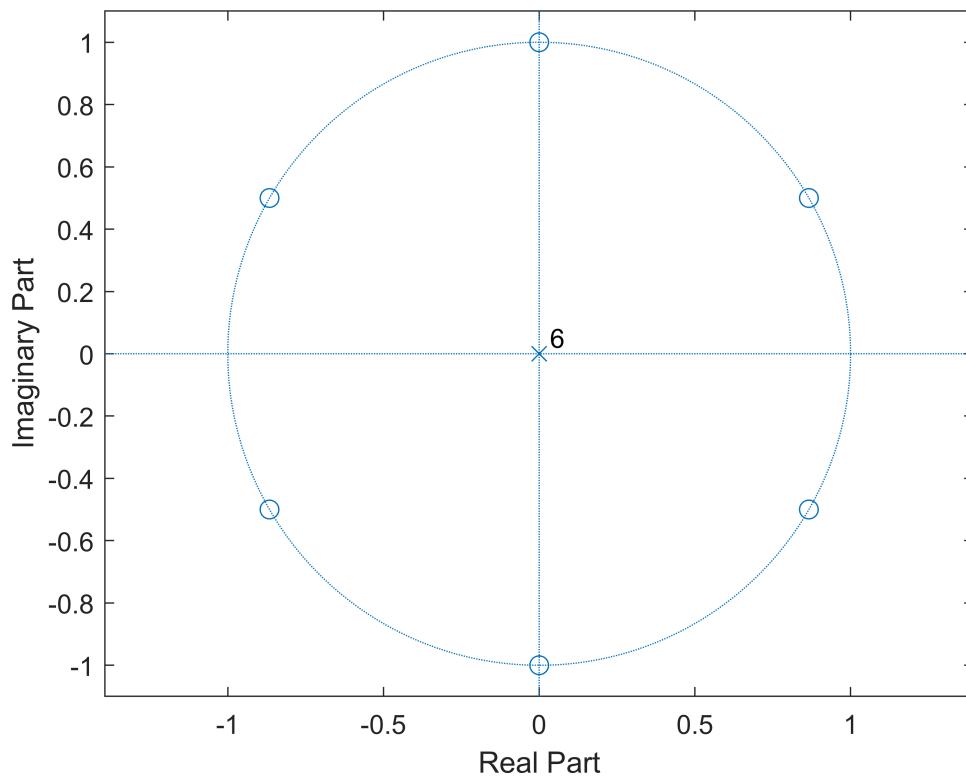
```
% Compute and plot the coherence estimate  
mscohore(y, x);  
ylim([0, 1.05]);
```



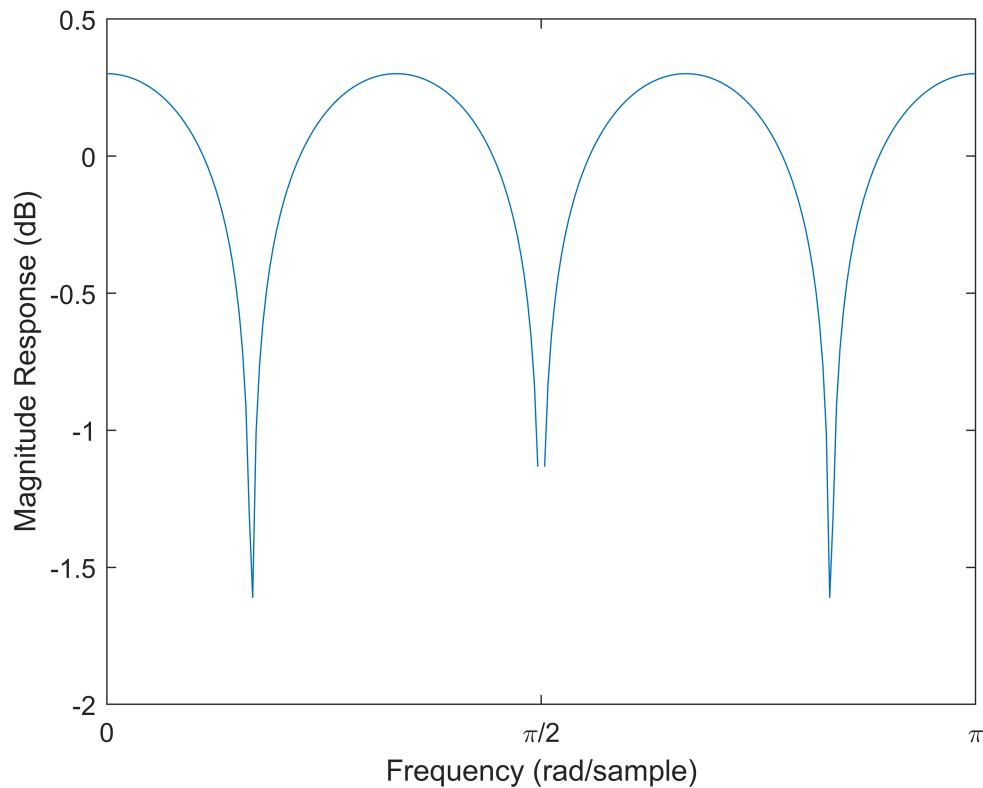
Notice that the coherence function dips at certain frequencies. This is because the comb filter removes certain frequency components of the input signal. Therefore, there is no coherence between the input signal and the output signal at those specific frequencies since the output signal does not contain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



Problems

Can Leakage be reduced in estimates of PSD by applying a window function to the data?

3. Leakage can be reduced in estimates of power spectral density by applying a window function to the data.

Answer: TRUE.

Leakage is an artefact of an FFT applied to a non-periodic data. Even if a signal is periodic like a sinusoid, when the signal is measured with some interval it can be non-periodic. The issue is that FFT assumes that the signal is periodic and repeats itself after the measured interval. This is not the case when the data is non-periodic, which contains sharp transitions at the end of each measured interval. These sharp changes have a broad frequency response which lead to spectral leakage. Windows can help minimize the effects of leakage by smoothing the time domain signal, but cannot eliminate leakage.

Estimate PSD from an ACRS assuming MA(1) process

The autocorrelation function for a MA(1) process has been found to be

$$\begin{array}{c|cc} |l| & r_{yy}(l) \\ \hline 0 & 2 \\ 1 & 1 \\ |l| \geq 2 & 0 \end{array}$$

```
clear variables;
```

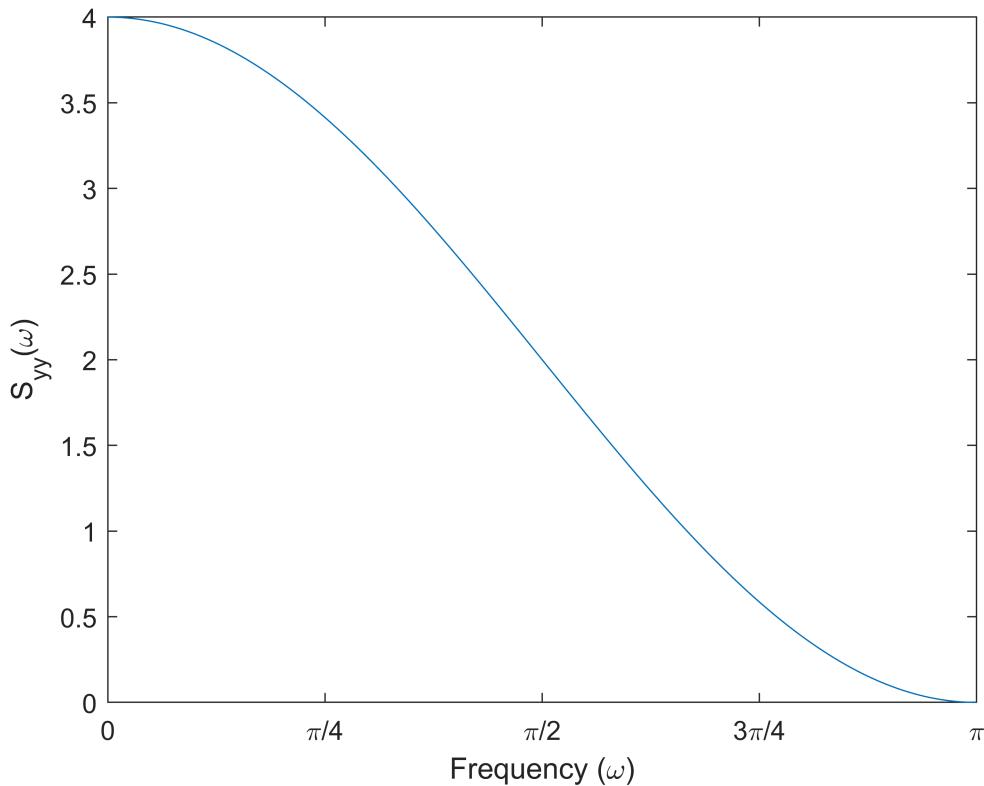
The PSD for an MA(q) process can be computed using Eq. (13.119):

$$S_{xx}(\omega) = r_{xx}[0] + 2 \sum_{\ell=1}^{\infty} r_{xx}[\ell] \cos \omega \ell, \quad (13.119)$$

Plugging in our values, we get:

$$S_{yy}(\omega) = 2 + 2(1 \cos(1\omega)) = 2 + 2\cos(\omega)$$

```
w = 0:0.001:2*pi;
r_xx = [2, 1, 0];
S = r_xx(1);
for l = 2:numel(r_xx)
    S = S + 2 * r_xx(l)*cos(w*(l-1));
end
plot(w, S);
xlabel('Frequency (\omega)');
ylabel('S_{yy}(\omega)');
set(gca,'XTick',0:pi/4:pi)
set(gca,'XTickLabel',{'0', '\pi/4', '\pi/2', '3\pi/4', '\pi'})
xlim([0, pi])
```



Estimate PSD from an ACRS assuming MA(2) process

For a given random process $\{x(n)\}$ the autocorrelation has been estimated and is given by:

$ m $	$r_x(m)$
0	4
1	2
2	-1

Estimate the power density spectrum under the assumption that $\{x(n)\}$ can be described as a *MA(2)* process.

```
clear variables;
```

The PSD for an *MA(q)* process can be computed using Eq. (13.119):

$$S_{xx}(\omega) = r_{xx}[0] + 2 \sum_{\ell=1}^{\infty} r_{xx}[\ell] \cos \omega \ell, \quad (13.119)$$

This gives following expression:

$$\begin{aligned} S_{xx}(\omega) &= 4 + 2(2\cos(\omega) + (-1)\cos(2\omega)) \\ &= 4 + 4\cos(\omega) - 2\cos(2\omega) \end{aligned}$$

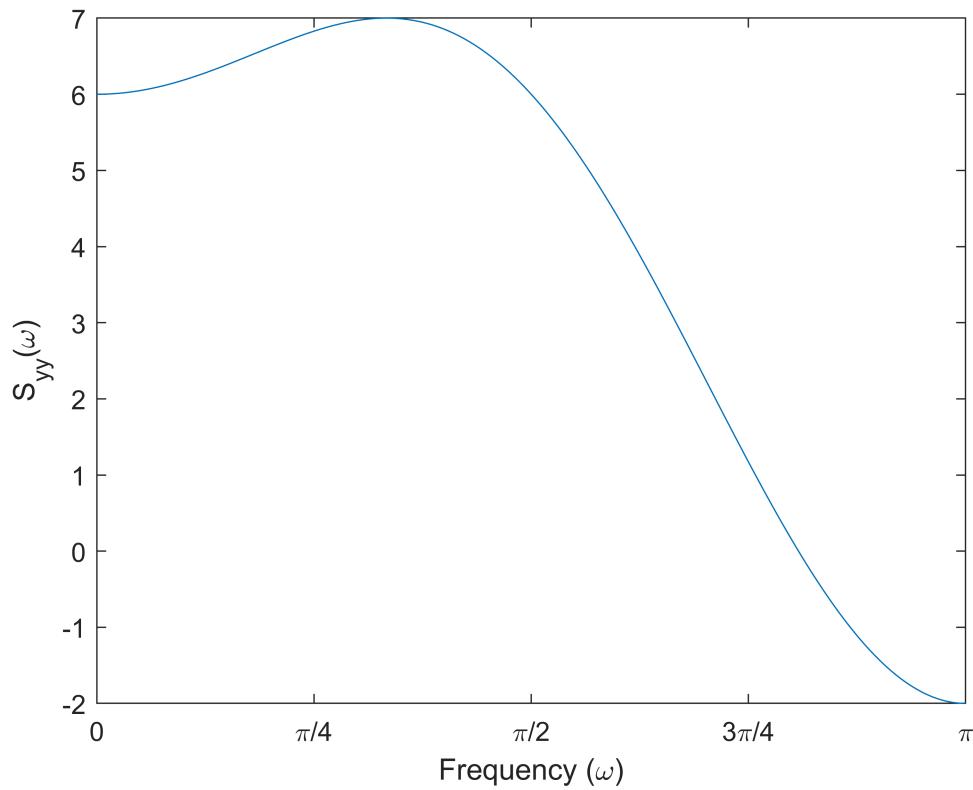
```

w=0:0.001:pi;
r_xx = [4, 2, -1];

S = r_xx(1);
for l = 2:numel(r_xx)
    S = S + 2 * r_xx(l)*cos(w*(l-1));
end

plot(w, S);
xlabel('Frequency (\omega)');
ylabel('S_{yy}(\omega)');
set(gca,'XTick',0:pi/4:pi)
set(gca,'XTickLabel',{'0', '\pi/4', '\pi/2', '3\pi/4', '\pi'})
xlim([0, pi])

```



Quiz: Determine if two sinusoids have the same PSD?

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 $x(n) = \text{Acos}(\omega n + \phi)$, 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.

Quiz: How do we measure if a system is LTI?

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:

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

[✓] Problem 14.57: Compare different PSD estimates using noise recorded on F-16

This problem is basically the same as 14.42. The only difference is that we will work with data from the real world.

This problem uses the signal file f16.mat that contains noise recorded at the copilot's seat of an F-16 airplane using a 16 bit A/D converter with $F_s = 19.98$ kHz.

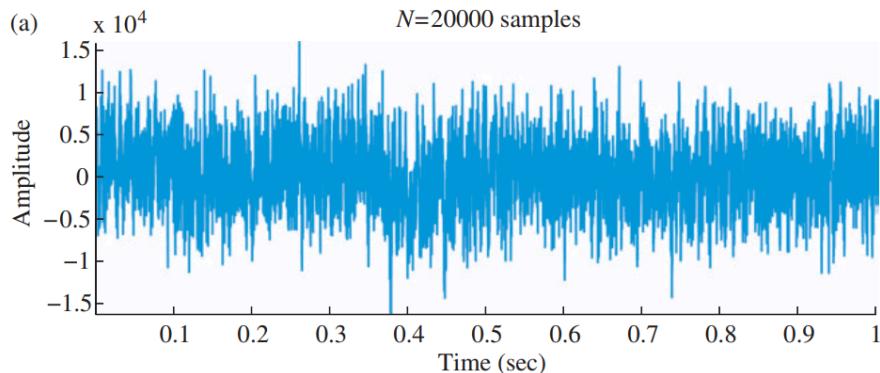


Figure (a) shows a waveform of F-16 noise recorded at the co-pilot's seat with a sampling rate of 19.98 kHz using a 16 bit ADC.

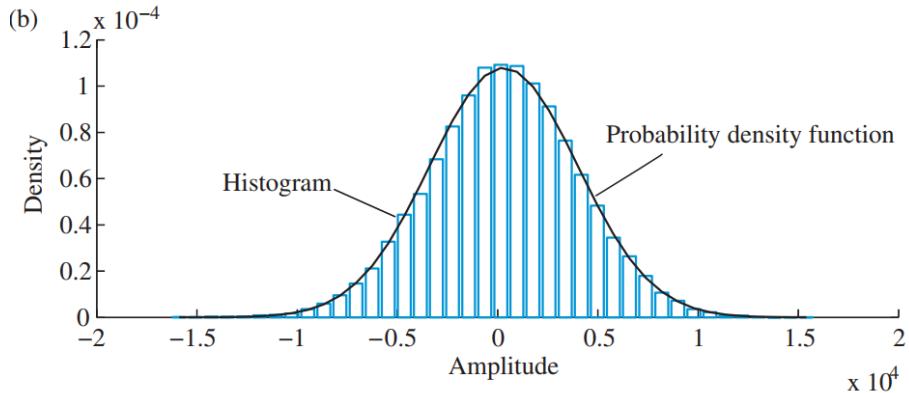
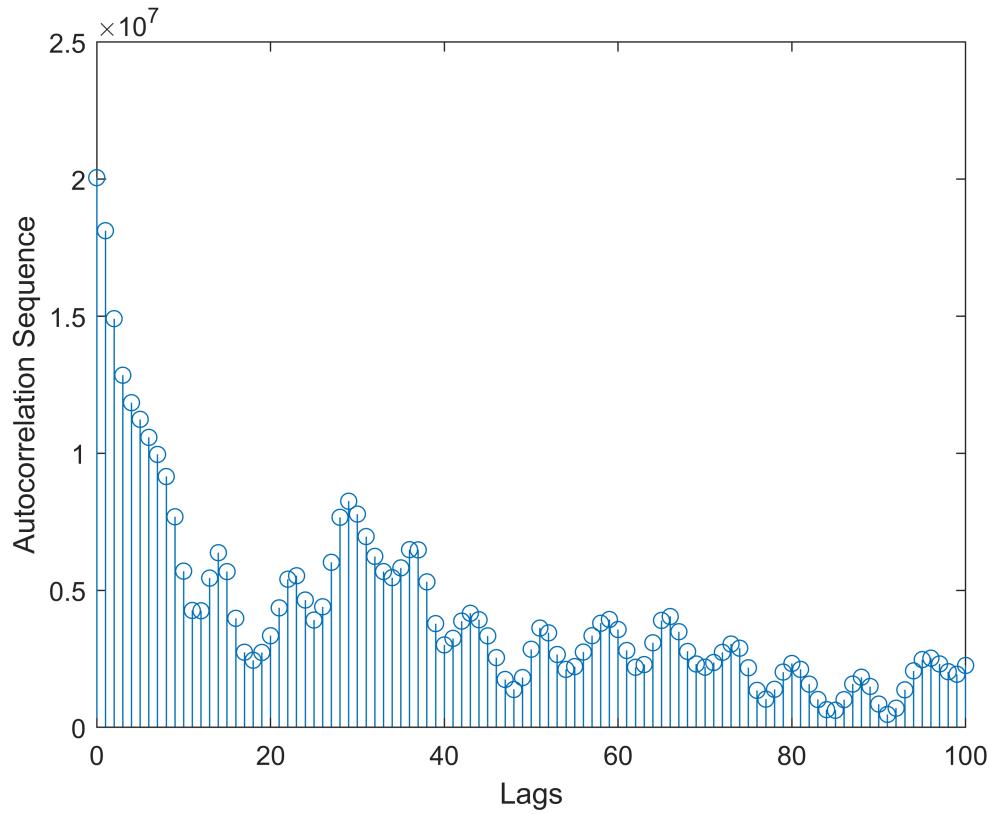


Figure (b) shows the histogram and theoretical probability density function of the F-16 noise.

We want to analyze this signal in terms of its ACRS and its spectral characteristics.

1) Compute and plot the ACRS estimate of the noise process.

```
load('f16.mat')
y = f16;
L = 100;
[lags, r_yy] = xcorr(y, L, 'biased');
mid = ceil(numel(r_yy)/2);
stem(r_yy(mid:end), lags(mid:end))
xlabel('Lags')
ylabel('Autocorrelation Sequence')
```



How should we interpret the ACRS?

- From the autocorrelation sequence, we see that there is some structure in the signal. The sequence is not decaying smoothly from lag zero to 100.
- The ACRS does not change sign so it seems to be some kind is low-frequency signal
- In order for the ACRS to go from lags zero to 100, it must be some higher-order process AR(p) model. It is definitely not a MA(q) process model.

2) Estimate model parameters for an **AR(2)** and **AR(4)** models.

An AR(p) model is given by:

$$y(n) = -\sum_{k=1}^p [a_k y(n-k)] + b_0 x(n)$$

The autocorrelation of AR(q) model was derived in Eq. 13.141 as:

$$r_{yy}(\ell) = -\sum_{k=1}^p a_k r_{yy}(\ell-k), \quad \ell > 0$$

This is useful because we can use it to find the coefficients $\{a_k\}$ of an AR(q) model using the autocorrelation $r_{yy}(\ell)$ computed numerically in MATLAB. This means that the expression in Eq. 13.141 becomes a set of p linear equations.

An AR(2) model is given by:

$$y(n) = -(a_1 y(n-1) + a_2 y(n-2)) + b_0 x(n)$$

We can estimate the model parameters of a second-order AR model $p = 2$ by creating two equations with two unknowns:

$$r_{yy}(1) = -a_1 r_{yy}(0) - a_2 r_{yy}(-1)$$

$$r_{yy}(2) = -a_1 r_{yy}(1) - a_2 r_{yy}(0)$$

We can write it into matrix form using the Toeplitz matrix:

$$\begin{bmatrix} r_{yy}(0) & r_{yy}(-1) \\ r_{yy}(1) & r_{yy}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} r_{yy}(1) \\ r_{yy}(2) \end{bmatrix}$$

In the general case, it becomes:

$$\begin{bmatrix} r_{yy}[0] & r_{yy}[1] & \dots & r_{yy}[p-1] \\ r_{yy}[1] & r_{yy}[0] & \dots & r_{yy}[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_{yy}[p-1] & r_{yy}[p-2] & \dots & r_{yy}[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = -\begin{bmatrix} r_{yy}[1] \\ r_{yy}[2] \\ \vdots \\ r_{yy}[p] \end{bmatrix},$$

More compactly written as:

$$\mathbf{R}_y \mathbf{a} = -\mathbf{r}_y$$

This is systems of linear equations which can be solved in MATLAB.

```

p = 2; % Model order
[r_yy, lags] = xcorr(y, p, 'biased');

% Select elements r_yy[0] to r_yy[p-1]
R_elems = r_yy(p+1:2*p);

% Create the Toeplitz matrix
R = toeplitz(R_elems);

% Select elements r_yy[1] to r_yy[p]
r = r_yy(p+2:2*p+1);

% Solve systems of linear equations using mldivide function
a = mldivide(R, -r)

a =
    2x1
    -1.2628
    0.3975

```

Once we have found the coefficients, we can predict the signal using the previous two samples:

$$y(n) = -(-1.2628 y(n-1) + 0.3975 y(n-2)) + b_0 x(n)$$

```

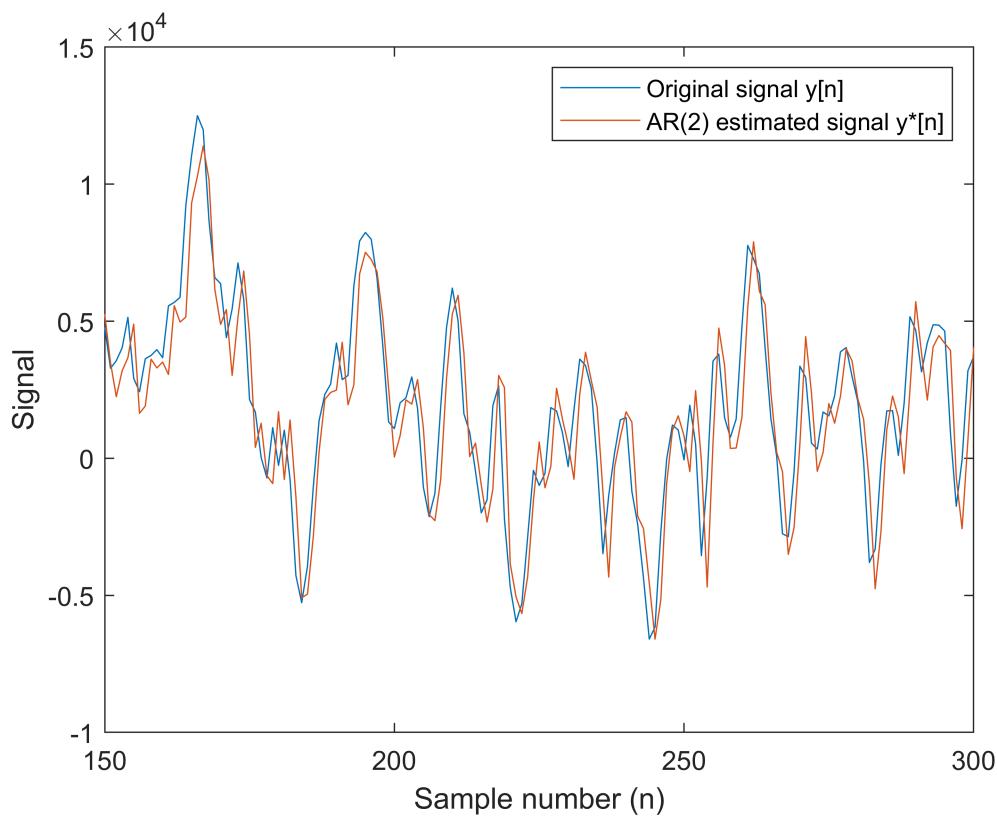
N = 1000;

% Generate the input
x = randn(N, 1)'; % Random noise with zero mean and unit variance.
b0 = 1;

```

```
% Estimate the output signal using the AR(2) model
y_hat = zeros(N, 1);
n = 3:N; % Trick to make it work in MATLAB
y_hat(n) = -(a(1)*y(n-1) + a(2)*y(n-2));
y_hat = y_hat + b0*x;

% Plotting code
n = 1:N;
plot(n, y(n), n, y_hat(n))
legend('Original signal y[n]', 'AR(2) estimated signal y*[n]');
xlabel('Sample number (n)')
ylabel('Signal')
xlim([150, 300]) % Zoom
```



```
[a, v] = arfit(y, 4)
```

```
a = 4x1
-1.4473
0.9584
-0.4942
0.0705
v = 2.6038e+06
```

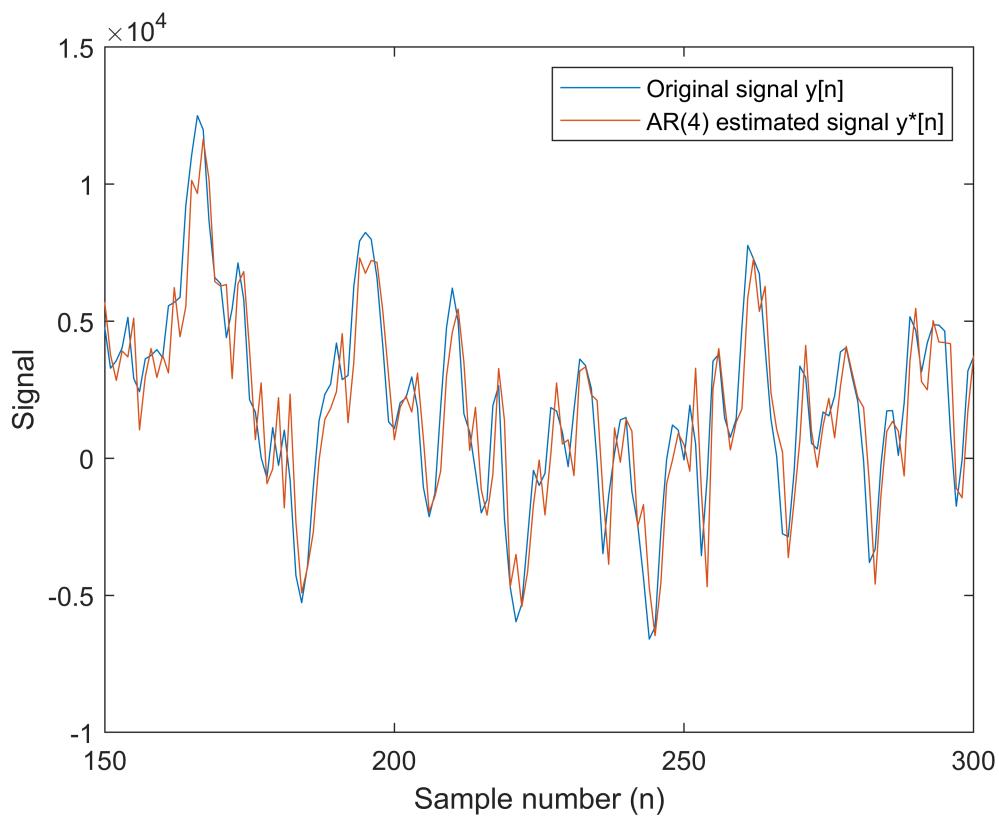
```
N = 1000;
y_hat = zeros(N);
```

```

n = 5:N; % Trick to make it work in MATLAB
y_hat(n) = -(a(1)*y(n-1) + a(2)*y(n-2) + a(3)*y(n-3) + a(4)*y(n-4));

% Plotting code
n = 1:N;
plot(n, y(n), n, y_hat(n))
legend('Original signal y[n]', 'AR(4) estimated signal y*[n]');
xlabel('Sample number (n)')
ylabel('Signal')
xlim([150, 300]) % Zoom

```



3) Compute and plot the PSD estimate using the AR(2) and AR(4) models.

The power spectrum of an ARMA(p, q) process is given by:

$$S_{yy}(\omega) = \sigma_x^2 |H(e^{j\omega})|^2 = \sigma_x^2 \left| \frac{\sum_{k=0}^q b_k e^{-j\omega k}}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2. \quad (13.133)$$

The power spectrum of an AR(p) process is given by:

$$S_{yy}(\omega) = \sigma_x^2 \left| \frac{1}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2$$

The algorithm is as follows:

1. Find the coefficients $\{a_1, a_2, \dots, a_p\}$ for the AR(p) model
2. Compute the transfer for the AR(p) by computing the sum and finding its reciprocal
3. Compute the conjugate of the transfer function: $|H(e^{j\omega})|^2$
4. Multiply it with the variance σ_x^2

The algorithm is implemented in the functions `arfit()` and `ar2psd()` functions:

```
N = 256;

[a2, v2] = arfit(y, 2);
[S2, w] = ar2psd(a2, v2, N);

[a4, v4] = arfit(y, 4);
[S4, w] = ar2psd(a4, v4, N);

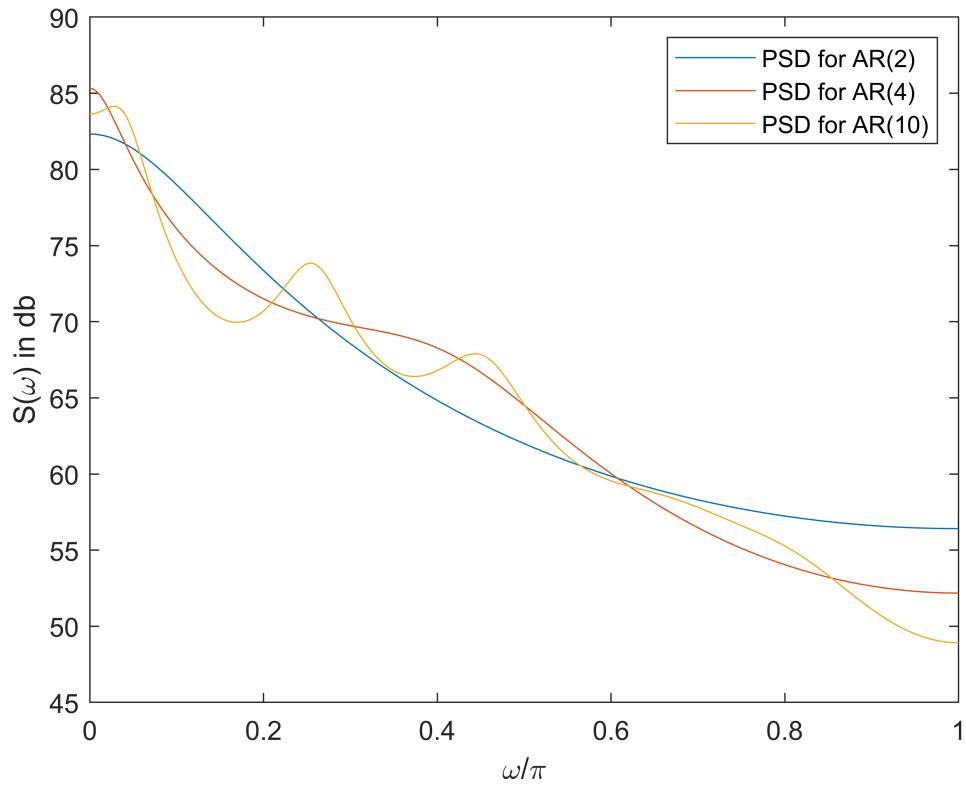
[a10, v10] = arfit(y, 10);
[S10, w] = ar2psd(a10, v10, N);

w_over_pi = w/pi;
S2db = pow2db(S2); % Use real(S2) to get rid of warning
S4db = pow2db(S4);
S10db = pow2db(S10);

plot(w_over_pi, S2db, w_over_pi, S4db, w_over_pi, S10db)
```

Warning: Imaginary parts of complex X and/or Y arguments ignored

```
legend('PSD for AR(2)', 'PSD for AR(4)', 'PSD for AR(10)')
xlabel('\omega/\pi')
ylabel('S(\omega) in db')
```



From this plot, we see that the majority of the energy is located at low frequencies. Since an AR(2) has only two model parameters, it has the smoothest spectrum. When we increase the number of model parameters, we see more structure and finer details of the spectrum. Notice that we get some peaks at frequencies $0.25\frac{\omega}{\pi}$ and $0.45\frac{\omega}{\pi}$.

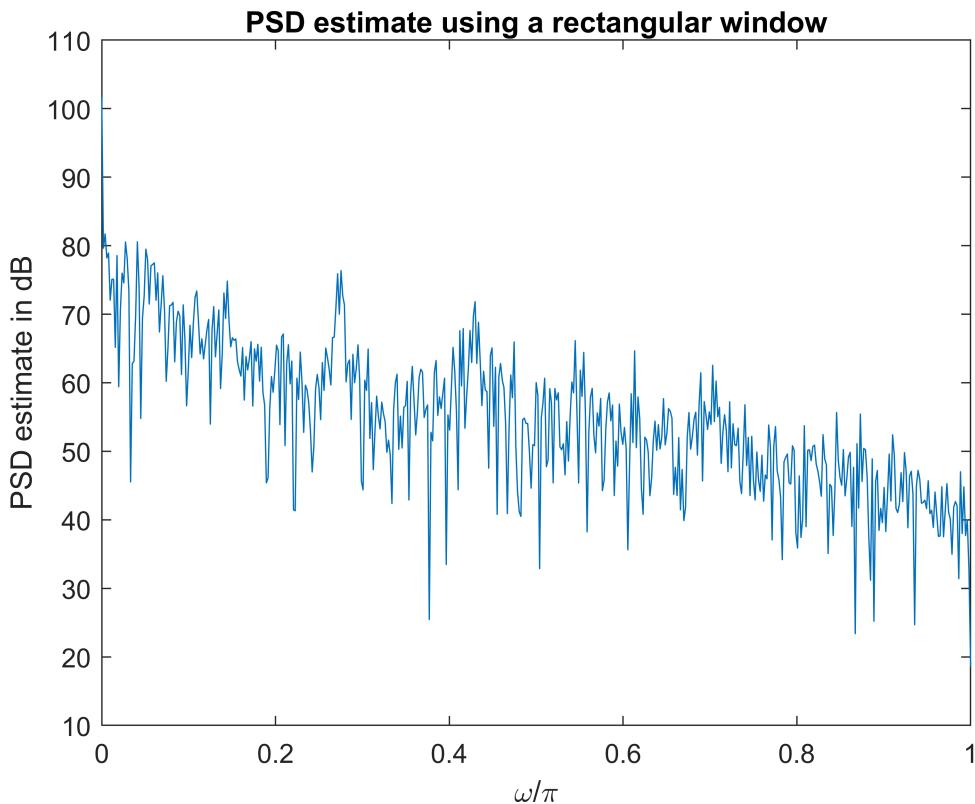
Why use AR(p) models to estimate the PSD? Because it is simple method if we assume that the given signal $y(n)$ is created by sending a white noise signal at an IIR filter. If this assumption is true, then we can model the signal $y(n)$ using the AR(p) model. The model parameters can be found by solving a linear problem which is easily solved. In comparison to the MA(q) models which yields nonlinear equations.

PSD estimate using AR(p) models does not reduce the uncertainty of whether to believe the estimate. For example, recomputing the PSD estimate on the next segment of the data, the autocorrelation will probably be slightly different. This means that the model parameters will be different which results in a different spectrum.

4) Compute and plot the periodogram PSD estimate of the noise process.

We can compute a PSD estimate using the built-in `periodogram()` function in MATLAB:

```
N = 1024;
[I, w] = periodogram(y, [], N);
plot(w/pi, pow2db(I));
xlabel('\omega/\pi')
ylabel('PSD estimate in dB')
title('PSD estimate using a rectangular window')
```



Again, we observe the similar picture than we fitted an AR(p) models and computed the corresponding PSD estimates.

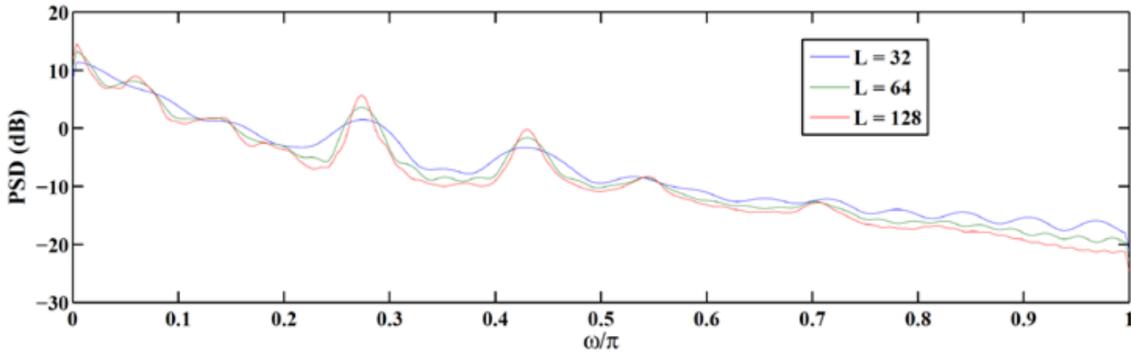
Notice some peaks at frequencies $0.27 \frac{\omega}{\pi}$ and $0.42 \frac{\omega}{\pi}$. Note how the periodogram has a large variance.

5) Compute the Bartlett PSD estimate

Compute the Bartlett PSD estimate using $L = 32, 64$, and 128 . Plot your result of the averaged estimate.

To Bartlett PSD estimate, we use the modified periodogram with the triangular window. In this problem, we are asked to segment the signal in smaller parts of sizes $L = 32, L = 64$ and $L = 128$, compute the estimates for each segment and average them together to get the PSD estimate. This is basically the Welch method without any overlap.

```
N = 1024;
% This results in an error:
% "The WINDOW must be a vector of the same length as the signal"
% Can we just pad the window signal with zeros?
%[I_L32, w] = periodogram(y, bartlett(32), N);
%[I_L64, w] = periodogram(y, bartlett(64), N);
%[I_L128, w] = periodogram(y, bartlett(128), N);
%plot(w/pi, pow2db(I_L32));
```



As we increase from 32 to 64, we observe a large change in the peaks. We increase the resolution by using longer windows. The window size determines the number of samples that we take the Fourier Transform on.

Going from 64 to 128, the change is less pronounced. It appears that it is saturated so thus a window size of 128 seems to be right size i.e., we have enough number of samples to estimate frequency peaks.

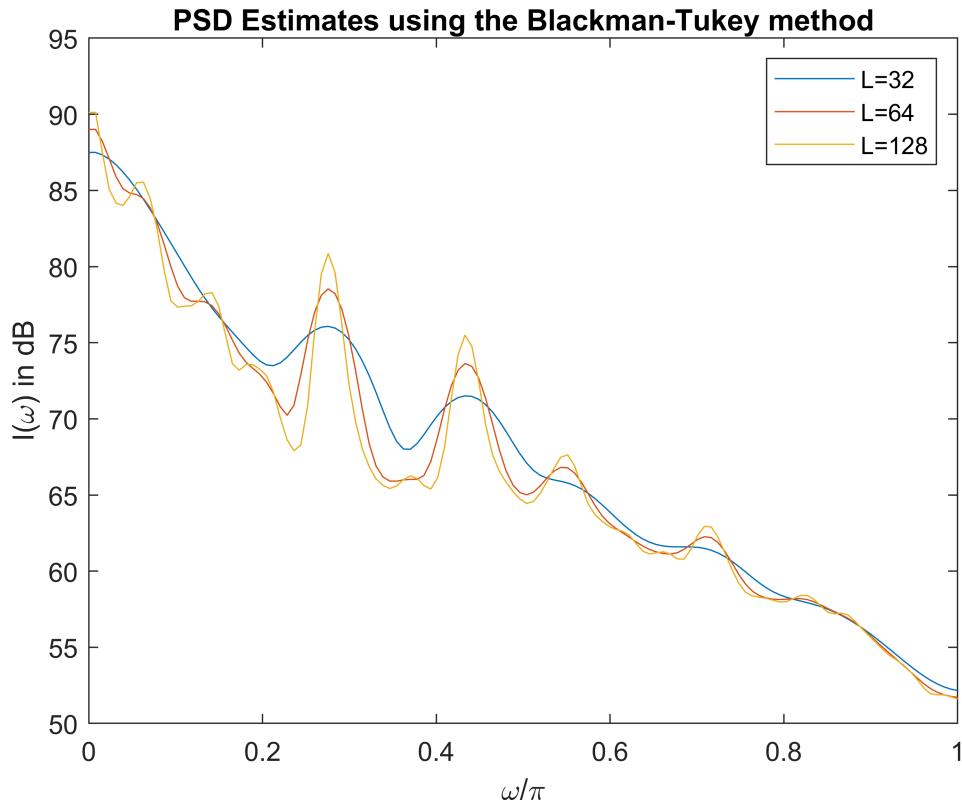
6) Compute the Blackman–Tukey PSD estimate

Compute the Blackman-Tukey PSD estimate using $L = 32, 64$, and 128 using the Bartlett lag window. Plot your result of the averaged estimate.

```
load('f16.mat'); y = f16;

N = 256;
w = linspace(0, 1, N/2);
I_L32 = psdbt(y, 32, N);
I_L64 = psdbt(y, 64, N);
I_L128 = psdbt(y, 128, N);

plot(w, pow2db(I_L32), w, pow2db(I_L64), w, pow2db(I_L128));
legend('L=32', 'L=64', 'L=128')
title('PSD Estimates using the Blackman-Tukey method')
xlabel('\omega/\pi')
ylabel('I(\omega) in dB')
```



As we increase the window size, we see peaks at the roughly the same frequencies as before. Whether these frequencies are real is difficult to say.

7) Compute the Welch PSD estimate

Compute the Welch PSD estimate using 50% overlap, Hamming window, and $L = 32, 64$, and 128 . Plot your result of the averaged estimate.

```

N = 256;
N_fft = 512;
overlap_pct = 0.5;
w = linspace(0, 1, N+1);

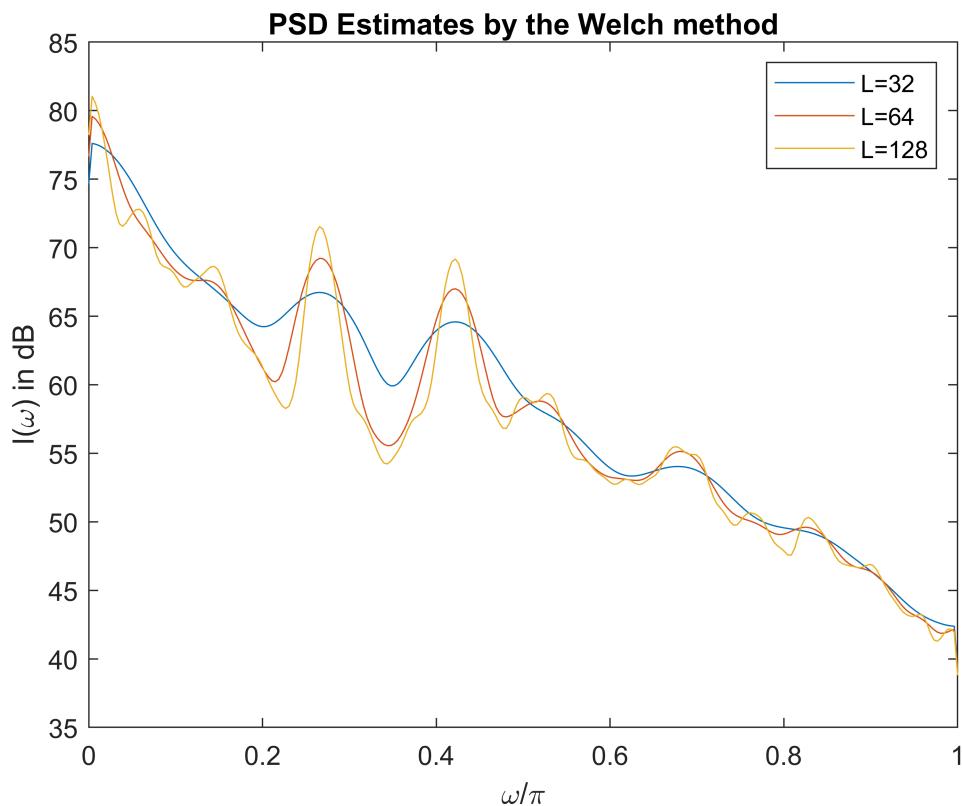
% Speed up computation by using a subset of the data
y_reduced = y(1:N*10);

I_L32 = pwelch(y_reduced, hamming(32), overlap_pct*32, N_fft);
I_L64 = pwelch(y_reduced, hamming(64), overlap_pct*64, N_fft);
I_L128 = pwelch(y_reduced, hamming(128), overlap_pct*128, N_fft);

plot(w, pow2db(I_L32), w, pow2db(I_L64), w, pow2db(I_L128));

legend('L=32', 'L=64', 'L=128')
title('PSD Estimates by the Welch method')
xlabel('\omega/\pi')
ylabel('I(\omega) in dB')

```



Again, we see that when the window size is 32, we get wider "peaks". As we increase the window size, these peaks become narrower. At window size 128, we cannot really say whether we have saturated.

8) Compare the plots in the above four parts and comment on your observation.

LESSON: Now the question is, which PSD estimate should we believe in? Doing PSD estimation is a little bit like black art. It is really difficult to estimate PSD without knowing something about the process that generated the signal. Are some of the bumps in the PSD caused by the engine or the turbulence? In practice, we would record the sound from multiple places. This allows us to compare PSD estimates and compare energy.

[✓] Problem 14.42: Compare periodogram, modified periodogram and Blackman-Tukey methods in terms of signal resolution

Consider the harmonic process signal model

$$x[n] = \cos(0.44\pi n + \phi_1) + \cos(0.46\pi n + \phi_2) + v[n], \quad 0 \leq n \leq 256$$

where ϕ_1 and ϕ_2 are IID random variables uniformly distributed over $[-\pi, \pi]$ and $v[n] \sim \text{WGN}(0, 2)$.

```
clear variables;
```

So from the signal model, we know from that there are two peaks and constant noise floor at 2. How do we know this? We can compute it:

The autocorrelation of real sinusoid $z(n) = A \cos(\omega_0 n + \phi)$ is $r_{zz}(\ell) = \frac{A^2}{2} \cos(\omega_0 \ell)$.

Since the PSD is the Fourier transform of the autocorrelation, we get:

$$S_{zz}(\omega) = \frac{A^2}{4} \delta(\omega + \omega_0) + \frac{A^2}{4} \delta(\omega - \omega_0)$$

This means that power is concentrated at $\pm\omega_0$. In the PSD, we will therefore see two peaks $\pm\omega_0$

The autocorrelation of white Gaussian noise is $r_{vv}(\ell) = \sigma_v^2 \delta(\ell) = 2\delta(\ell)$. The PSD of Gaussian noise is constant around σ_v^2 .

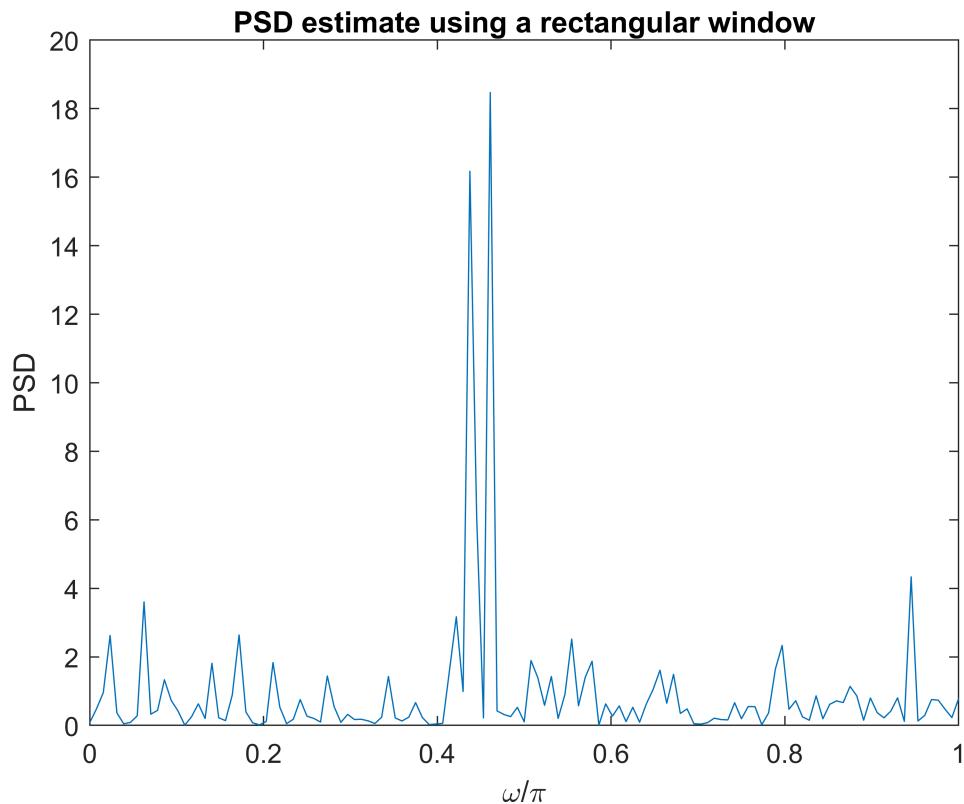
So the true PSD should be something like:

$$S_{xx}(\omega) = \frac{1}{4} \delta(\omega - 0.44\pi) + \frac{1}{4} \delta(\omega - 0.46\pi) + 2$$

1) Estimate the PSD using the periodogram and plot the spectrum.

```
N = 256;
n = 0:N-1;
v = randn(1,N) * sqrt(2);
phi = 2*pi*(rand(1,2)-0.5);
x = cos(0.44*pi*n + phi(1)) + cos(0.46*pi*n + phi(2)) + v;
%plot(n, x); xlim([0, N]);

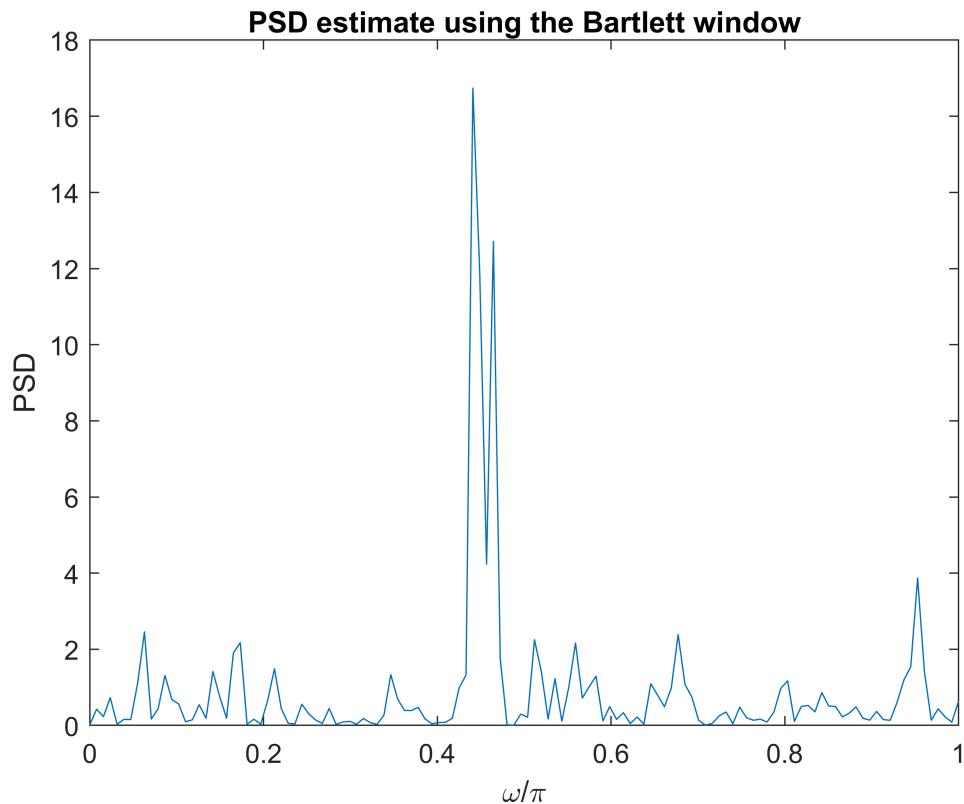
[I1, w] = periodogram(x, [], N);
plot(w/pi, I1);
xlabel('omega/pi')
ylabel('PSD')
title('PSD estimate using a rectangular window')
```



2) Estimate the PSD using the modified periodogram with Bartlett window

Estimate the PSD using the modified periodogram with Bartlett window and plot the spectrum.

```
L = N;
I = periodogram(x, bartlett(L));
w = linspace(0,1,N/2)*pi;
plot(w/pi, I(1:N/2));
xlabel('omega/pi')
ylabel('PSD')
title('PSD estimate using the Bartlett window')
```



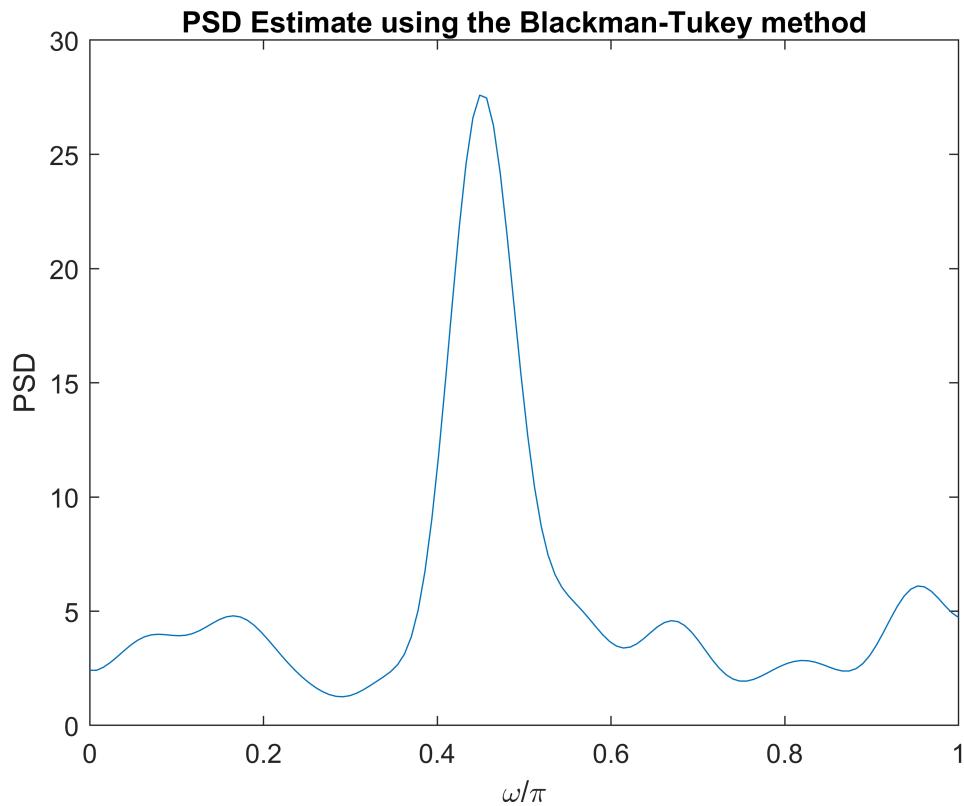
3) Estimate the PSD using the Blackman–Tukey method

Estimate the PSD using the Blackman–Tukey method with Parzen window and $L=32$ and plot the spectrum.

```

L = 32;
K = N;
I = psdbt(x,L,K);
plot(w/pi, I(1:N/2));
xlabel('\omega/\pi')
ylabel('PSD')
title('PSD Estimate using the Blackman-Tukey method')

```



4) Which method performs best in terms of signal resolution?

The PSD estimate using the periodogram (with rectangular window) shows two distinct peaks at the appropriate frequencies. However, the picture is not completely clear because the variance is high. Using the Bartlett (triangular) window yields similar characteristics. It is difficult to say whether the modified periodogram is better or worse. Theory tells us that the rectangular window yields the smallest main lobe-width.

Blackman-Tukey performs some smoothing in order to minimise the variance of the spectrum. (The BT method attempts to suppress the high frequencies.) The price is that the BT method **smears out** the two frequency peaks so it looks like one peak. Since in cases like this problem where we have two sinusoids, it will be impossible to distinguish between them.

In terms of signal resolution, the periodogram is clearly better to separate the two frequencies from each other. In general, when the frequencies are close to each other then we are forced to use the periodogram.

INSIGHT: if you know something about the system under study then we can get better solution.

ADSI Problem 4.7: Coherence function

The magnitude square coherence function of two signals $x[n]$ and $y[n]$ is given by

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

Assume that $x[n]$ is the input to a given system, $y[n]$ is the output from the system, and the coherence function has been measured. What happens to the coherence function if

1) What happens to the coherence function if gain is increased from 1 to 2:

The gain setting of the amplifier in front of the ADC measuring $y[n]$ is increased from 1 to 2, i.e. $y[n] \rightarrow 2y[n]$?

To answer this question, we need to compute the new coherence function. Power Spectral Density $S_x(\omega)$ and Cross-Power Spectral Density $S_{yx}(\omega)$ are essentially the discrete-time Fourier Transform of the corresponding auto-correlation and cross-correlation.

So we need to compute those first. The autocorrelation for the input signal is the same because the input signal has not changed. We need to compute the autocorrelation for the new output signal.

Let $z(n) = 2y(n)$ denote the new output signal.

The autocorrelation for the new output signal $r_z(\ell)$ can be computed as follows:

$$r_z(\ell) = E[z(n)z(n - \ell)]$$

$$r_z(\ell) = E[2y(n)2y(n - \ell)]$$

$$r_z(\ell) = 4E[y(n)y(n - \ell)]$$

$$r_z(\ell) = 4r_y(\ell)$$

To find the PSD, we take the Fourier Transform:

$$S_z(\omega) = R_z(e^{j\omega}) = 4R_y(e^{j\omega}) = 4S_y(\omega)$$

This means that if the output gain is increased by 2, the PSD increases by a factor of 4.

Next, we need compute the cross-correlation $r_{zx}(\ell)$:

$$r_{zx}(\ell) = E[z(n)x(n-\ell)]$$

$$r_{zx}(\ell) = E[2y(n)x(n-\ell)]$$

$$r_{zx}(\ell) = 2E[y(n)x(n-\ell)]$$

$$r_{zx}(\ell) = 2r_{yx}(\ell)$$

To find the PSD, we take the Fourier Transform:

$$S_{zx}(\omega) = R_{zx}(e^{j\omega}) = 2R_{yx}(e^{j\omega}) = 2S_{yx}(\omega)$$

Finally, we can compute the coherence function:

$$|\gamma_{zx}(\omega)|^2 = \frac{|S_{zx}(\omega)|^2}{S_z(\omega)S_x(\omega)} = \frac{|2S_{yx}(\omega)|^2}{4S_y(\omega)S_x(\omega)} = \frac{|2|^2|S_{yx}(\omega)|^2}{4S_y(\omega)S_x(\omega)} = \frac{|S_{yx}(\omega)|^2}{S_y(\omega)S_x(\omega)}$$

The coherence function with the new output signal is the same the original coherence function.

This means that increasing the output gain by 2, nothing happens to the coherence function.

2) What happens to the coherence function if a delay happens in the measurement?

A delay happens in the measurement of $y[n]$ i.e. $y[n] \rightarrow y[n - D]$?

Let $z(n) = y(n - D)$

First, compute the autocorrelation for the new output signal $r_z(\ell)$:

$$r_z(\ell) = E[y(n - D)y(n - D - \ell)]$$

Let $m = n - D$ then we can write the above equation as:

$$r_z(\ell) = E[y(m)y(m - \ell)]$$

This expression is the same as $r_y(\ell)$ so the delayed output signal does not change the autocorrelation:

$$r_z(\ell) = r_y(\ell)$$

Next, we compute the cross-correlation:

$$r_{zx}(\ell) = E[z(n)x(n - \ell)]$$

$$r_{zx}(\ell) = E[y(n - D)x(n - \ell)]$$

$$r_{zx}(\ell) = E[y(n)x(n - \ell + D)]$$

$$r_{zx}(\ell) = r_{yx}(\ell + D)$$

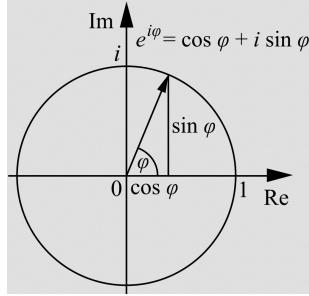
To find the Fourier Transform, we can use the Fourier pair: $f(t - t_0) \leftrightarrow F(e^{j\omega})e^{-j\omega t_0}$

$$R_{zx}(e^{j\omega}) = R_{yx}(e^{j\omega})e^{j\omega D} = S_{yx}(\omega)e^{j\omega D}$$

Finally, we can compute the coherence function:

$$|\gamma_{zx}(\omega)|^2 = \frac{|S_{zx}(\omega)|^2}{S_z(\omega)S_x(\omega)} = \frac{|e^{j\omega D}S_{yx}(\omega)|^2}{S_y(\omega)S_x(\omega)} = \frac{|e^{j\omega D}|^2 |S_{yx}(\omega)|^2}{S_y(\omega)S_x(\omega)}$$

The norm of the $e^{j\omega D}$ is 1 because this quantity describes a point on the unit circle:



Since the square of 1 is 1, the coherence function is the same:

$$|\gamma_{zx}(\omega)|^2 = \frac{|S_{yx}(\omega)|^2}{S_y(\omega)S_x(\omega)}$$

This means that a delay in the measurement does not effect the coherence function!

[✓] ADSI Problem 4.21: Minimum variance spectral estimation

The discussion of the improved filter bank method taught us to choose filters, that for a given center frequency reduced the influence of spectral leakage by selectively moving the sidelobes around to attenuate specific frequency regions of high spectral density. This problem takes a closer look at this interpretation.

The discussion of the improved filter bank method taught us to choose filters, that for a given center frequency reduced the influence of spectral leakage by selectively moving the sidelobes around to attenuate specific frequency regions of high spectral density. This problem takes a closer look at this interpretation.

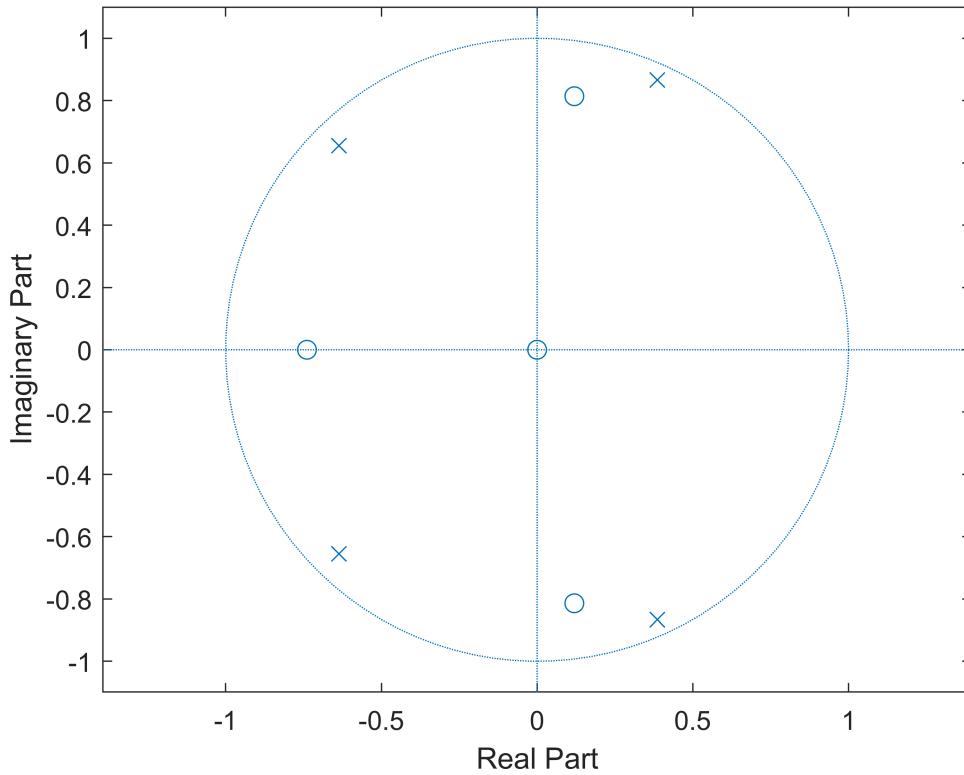
```
clear variables;
```

1) Create a stable signal model

Create a signal model with a suitable behaviour i.e. a couple of peaks in the spectrum. It is unimportant whether the model is AR, MA or ARMA as long as the model is stable.

Once we have created a signal, we can check whether it is stable by ensuring that all the poles and zeros are within the unit circle.

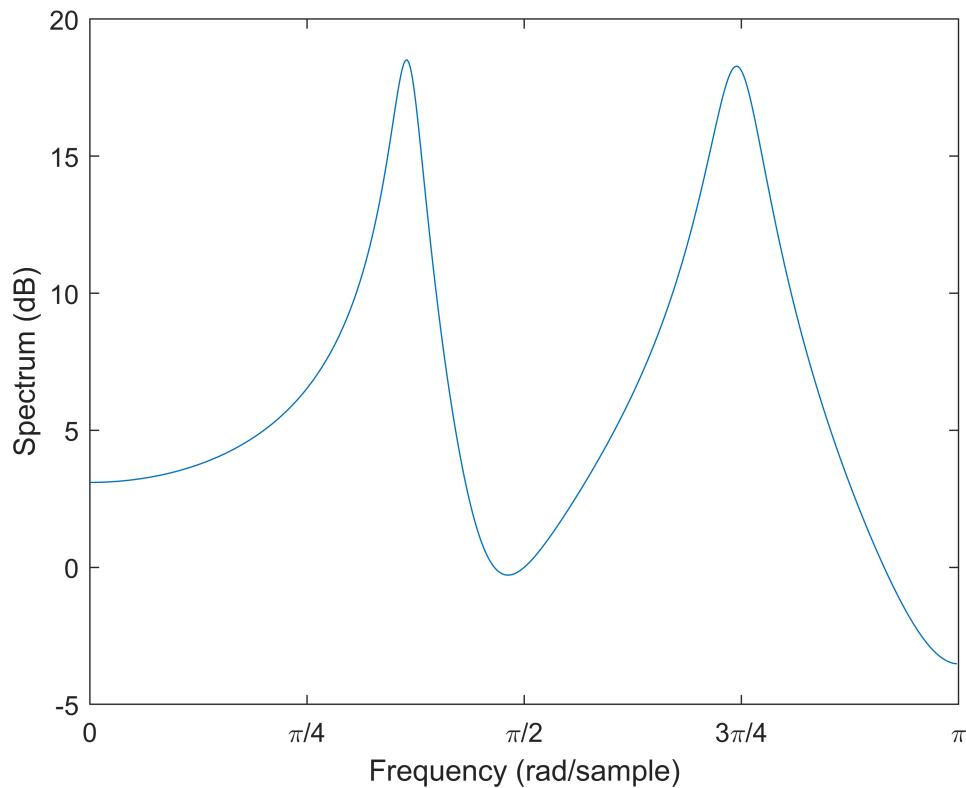
```
b = [2, 1, 1, 1];
a = [1, 0.5, 0.75, 0.5, 0.75];
zplane(b,a)
```



2) Use the signal model to plot the true spectrum.

We can plot the true spectrum by plotting the magnitude response of the transfer function or simply use the freqz function in MATLAB:

```
[H,w] = freqz(b, a);
plot(w, pow2db(H.*conj(H)));
set(gca,'XTick', 0:pi/4:pi)
set(gca,'XTickLabel', {'0','\pi/4','\pi/2','3\pi/4','\pi'})
xlabel('Frequency (rad/sample)')
ylabel('Spectrum (dB)')
xlim([0, pi]);
```



3) Create a realization of the signal and determine the autocorrelation matrix R_x in an appropriate size.

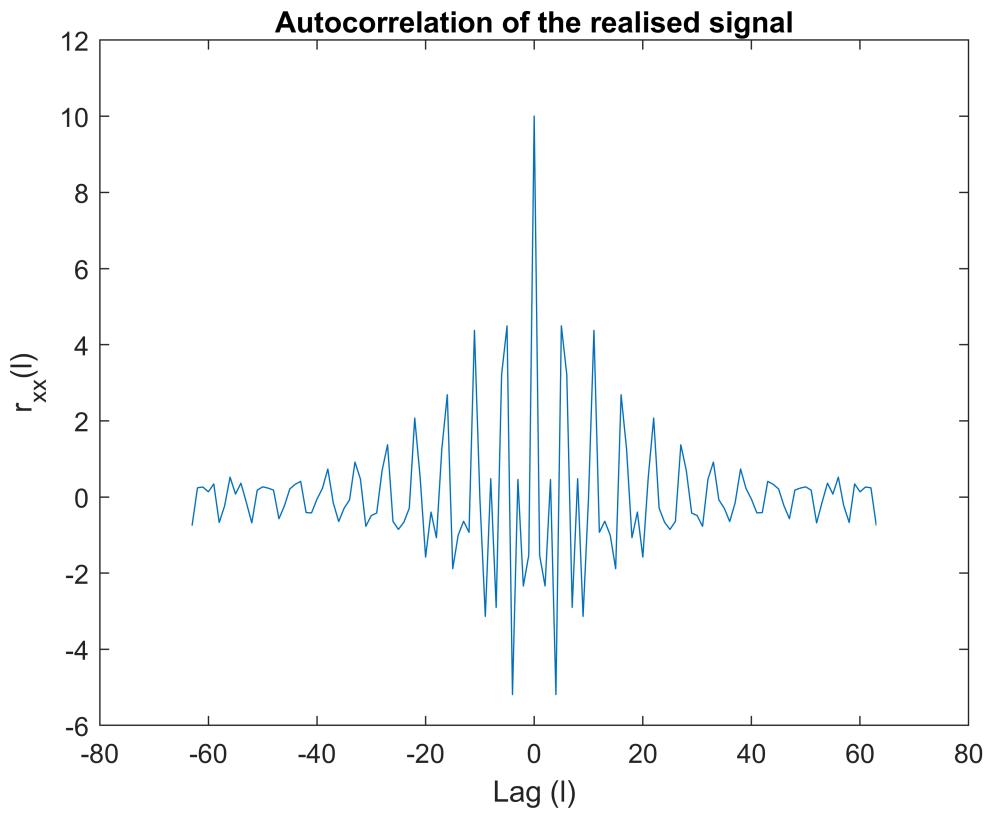
We can create a realisation of a signal using the filter function in MATLAB:

```

N = 4096;
n = randn(N,1);
x = filter(b, a, n);
N_Rx = 64; % size of autocorrelation matrix
[r_xx, lags] = xcorr(x, N_Rx-1, 'biased');

% Plot the autocorrelation for fun
plot(lags, r_xx);
xlabel('Lag (1)')
ylabel('r_{xx}(1)')
title('Autocorrelation of the realised signal')

```



```
% Create the autocorrelation matrix
R_xx = toeplitz(r_xx(N_Rx:end))
```

```
R_xx = 64x64
10.0044  -1.5305  -2.3385  0.4596  -5.1913  4.4950  3.2147  -2.9022 ...
-1.5305  10.0044  -1.5305  -2.3385  0.4596  -5.1913  4.4950  3.2147
-2.3385  -1.5305  10.0044  -1.5305  -2.3385  0.4596  -5.1913  4.4950
0.4596  -2.3385  -1.5305  10.0044  -1.5305  -2.3385  0.4596  -5.1913
-5.1913  0.4596  -2.3385  -1.5305  10.0044  -1.5305  -2.3385  0.4596
4.4950  -5.1913  0.4596  -2.3385  -1.5305  10.0044  -1.5305  -2.3385
3.2147  4.4950  -5.1913  0.4596  -2.3385  -1.5305  10.0044  -1.5305
-2.9022  3.2147  4.4950  -5.1913  0.4596  -2.3385  -1.5305  10.0044
0.4812  -2.9022  3.2147  4.4950  -5.1913  0.4596  -2.3385  -1.5305
-3.1373  0.4812  -2.9022  3.2147  4.4950  -5.1913  0.4596  -2.3385
:
:
```

```
% Compute the inverse of the autocorrelation matrix
R_xx_inv = inv(R_xx)
```

```
R_xx_inv = 64x64
0.2454  0.0049  0.0637  -0.0273  0.1656  -0.0906  -0.0109  -0.0195 ...
0.0049  0.2455  0.0062  0.0632  -0.0240  0.1638  -0.0908  -0.0113
0.0637  0.0062  0.2620  -0.0009  0.1062  -0.0474  0.1610  -0.0958
-0.0273  0.0632  -0.0009  0.2650  -0.0193  0.1163  -0.0462  0.1631
0.1656  -0.0240  0.1062  -0.0193  0.3767  -0.0806  0.1089  -0.0595
-0.0906  0.1638  -0.0474  0.1163  -0.0806  0.4100  -0.0766  0.1160
-0.0109  -0.0908  0.1610  -0.0462  0.1089  -0.0766  0.4105  -0.0757
-0.0195  -0.0113  -0.0958  0.1631  -0.0595  0.1160  -0.0757  0.4119
0.0738  -0.0180  0.0080  -0.1041  0.2127  -0.0870  0.1127  -0.0818
-0.0039  0.0737  -0.0191  0.0084  -0.1068  0.2142  -0.0868  0.1130
```

⋮

4) Calculate the spectrum using Eq. (14.4.12) from the note.

We can compute minimum variance power spectrum estimate as follows

$$P_{xx}^{\text{MV}}(f) = \frac{1}{\mathbf{E}^H(f)\mathbf{R}_{xx}^{-1}\mathbf{E}(f)} \quad (14.4.12)$$

where

- f is the frequency,
- \mathbf{R}_{xx} is the autocorrelation matrix
- $\mathbf{E}(f)$ is the constraint vector:

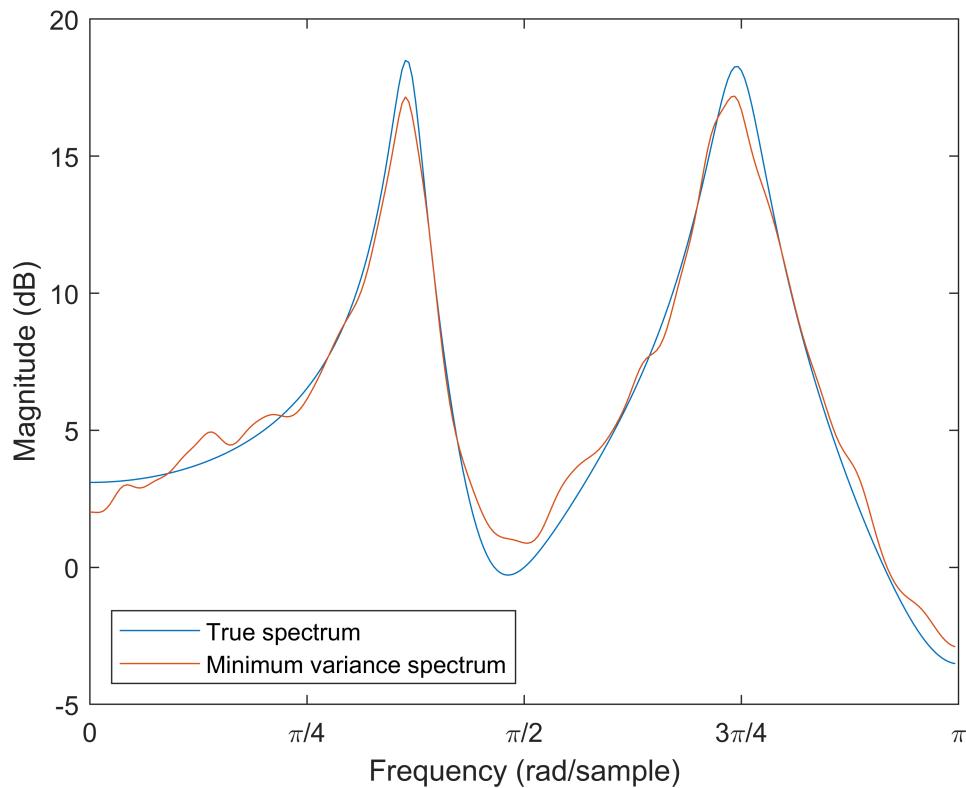
$$\mathbf{E}^t(f_l) = [1 \quad e^{j2\pi f_l} \quad \dots \quad e^{j2\pi p f_l}]$$

This is implemented in the function psdmv() function (see at the end of the document):

```
M = 64; % Size of the autocorrelation matrix
kF = 4; % Frequency resolution multiplier
[I, w] = psdmv(x, M, kF);
[H,w2] = freqz(b, a, M*kF);
plot(w, pow2db(H.*conj(H)), w, pow2db(I));
```

Warning: Imaginary parts of complex X and/or Y arguments ignored

```
set(gca,'XTick', 0:pi/4:pi)
set(gca,'XTickLabel', {'0','\pi/4','\pi/2','3\pi/4','\pi'})
xlabel('Frequency (rad/sample)')
ylabel('Magnitude (dB)')
legend({'True spectrum', 'Minimum variance spectrum'}, 'Location', 'southwest')
xlim([0, pi]);
```



5) Calculate the optimum filters

Choose a few representative frequencies in your spectrum. Calculate the optimum filters for estimation of these frequencies and plot the magnitude response of these. Are the filter magnitude responses in concordance with your intuition?

The optimum filters are given by

$$\hat{\mathbf{h}}_{\text{opt}} = \mathbf{\Gamma}_{xx}^{-1} \mathbf{E}(f_l) / \mathbf{E}^H(f_l) \mathbf{\Gamma}_{xx}^{-1} \mathbf{E}(f_l) \quad (14.4.10)$$

```
M = 64;
f1 = 1.1/(2*pi); % Correspond to the first peak
f2 = 1.5/(2*pi); % Middle valley
f3 = 2.3/(2*pi); % Second peak

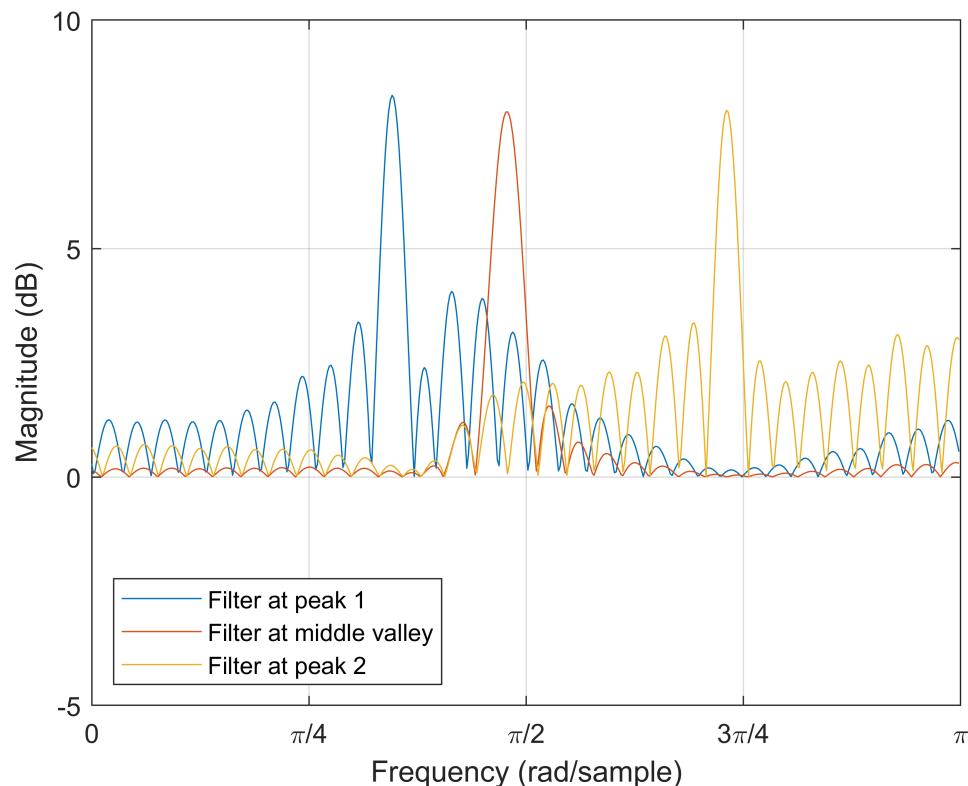
% Compute the optimal filters
h1 = optfilter(x, M, f1);
h2 = optfilter(x, M, f2);
h3 = optfilter(x, M, f3);

% Compute the transfer function
[H1, w] = freqz(h1, 1);
[H2, ~] = freqz(h2, 1);
[H3, ~] = freqz(h3, 1);
```

```

plot(w, abs(H1), w, abs(H2), w, abs(H3));
set(gca, 'XTick', 0:pi/4:pi)
set(gca, 'XTickLabel', {'0', '\pi/4', '\pi/2', '3\pi/4', '\pi'})
xlabel('Frequency (rad/sample)')
ylabel('Magnitude (dB)')
legend({'Filter at peak 1', 'Filter at middle valley', 'Filter at peak 2'}, 'Location', 'southwest')
grid on;
ylim([-5, 10]);
xlim([0, pi]);

```



Note how the blue and red peak filters goes to zero at the other peak.

Problem 1: PSD from ACRS assuming AR(2) process

The first few values of the autocorrelation function of a random process has been determined as

$ l $	$r_x(l)$
0	11.26
1	9.70
2	6.00
3	1.49
4	-2.53

Initially, it can be assumed that the random process can be modeled as an AR(2) process.

```
clear variables;
```

1) Compute AR(2) model coefficient given autocorrelation sequence

1. Calculate the model parameters and plot the power spectral density.

An AR(p) model is given by:

$$y(n) = -\sum_{k=1}^p [a_k y(n-k)] + b_0 x(n)$$

The autocorrelation of AR(q) model was derived in Eq. 13.141 as:

$$r_{yy}(\ell) = -\sum_{k=1}^p a_k r_{yy}(\ell-k), \quad \ell > 0$$

This is useful because we can use it to find the coefficients $\{a_k\}$ of an AR(q) model using the autocorrelation $r_{yy}(\ell)$ computed numerically in MATLAB. This means that the expression in Eq. 13.141 becomes a set of p linear equations.

An AR(2) model is given by:

$$y(n) = -(a_1 y(n-1) + a_2 y(n-2)) + b_0 x(n)$$

We can estimate the model parameters of a second-order AR model $p = 2$ by creating two equations with two unknowns:

$$r_{yy}(1) = -a_1 r_{yy}(0) - a_2 r_{yy}(-1)$$

$$r_{yy}(2) = -a_1 r_{yy}(1) - a_2 r_{yy}(0)$$

We can write it into matrix form using the Toeplitz matrix:

$$\begin{bmatrix} r_{yy}(0) & r_{yy}(-1) \\ r_{yy}(1) & r_{yy}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} r_{yy}(1) \\ r_{yy}(2) \end{bmatrix}$$

In the general case, it becomes:

$$\begin{bmatrix} r_{yy}[0] & r_{yy}[1] & \dots & r_{yy}[p-1] \\ r_{yy}[1] & r_{yy}[0] & \dots & r_{yy}[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_{yy}[p-1] & r_{yy}[p-2] & \dots & r_{yy}[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} r_{yy}[1] \\ r_{yy}[2] \\ \vdots \\ r_{yy}[p] \end{bmatrix},$$

More compactly written as *Yule–Walker equations*:

$$\mathbf{R}_y \mathbf{a} = -\mathbf{r}_y. \quad (13.143)$$

The input noise variance can be compute as follows:

$$\sigma_x^2 = \sigma_y^2 + \mathbf{a}^T \mathbf{r}_y = \sigma_y^2 - \mathbf{r}_y^T \mathbf{R}_y^{-1} \mathbf{r}_y \leq \sigma_y^2. \quad (13.144)$$

This is coded in MATLAB function (see end of document):

```
p = 2;
r_xx = [11.26, 9.70, 6.00, 1.49, -2.53]';
[a, v] = ar_from_acrs(r_xx, p)

a = 2x1
-1.5604
0.8114
v = 0.9922
```

The power spectrum of an ARMA(p, q) process is given by:

$$S_{yy}(\omega) = \sigma_x^2 |H(e^{j\omega})|^2 = \sigma_x^2 \left| \frac{\sum_{k=0}^q b_k e^{-j\omega k}}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2. \quad (13.133)$$

The power spectrum of an AR(p) process is given by:

$$S_{yy}(\omega) = \sigma_x^2 \left| \frac{1}{1 + \sum_{k=1}^p a_k e^{-j\omega k}} \right|^2$$

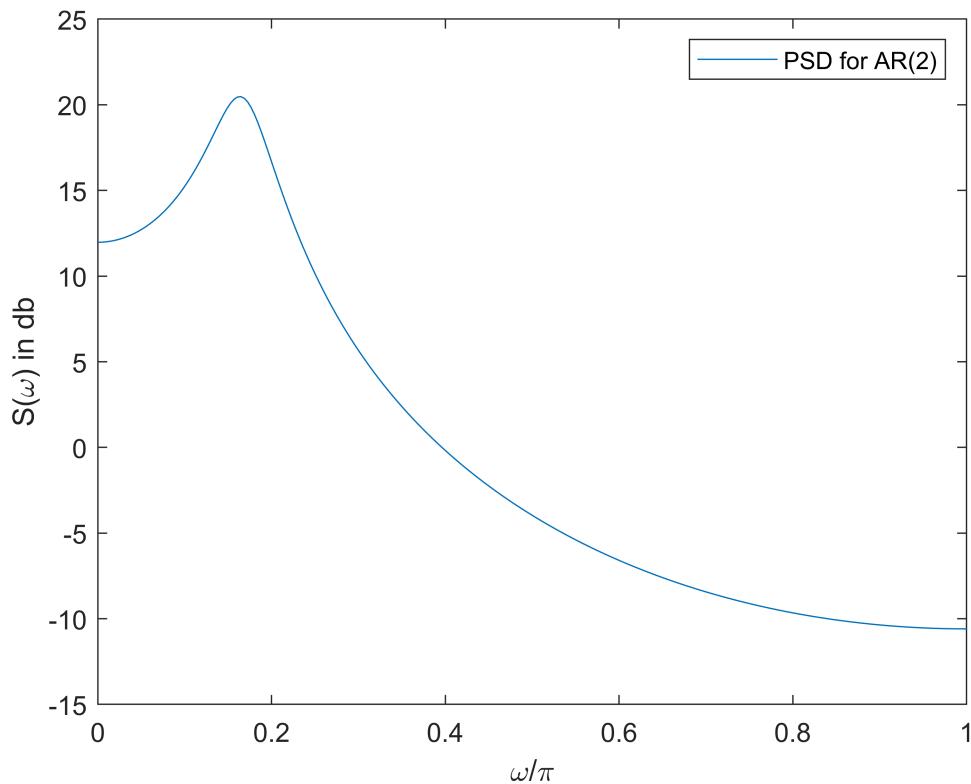
To solve this problem, the method is as follows:

1. Find the coefficents $\{a_1, a_2, \dots, a_p\}$ for the AR(p) model
2. Compute the transfer function for the AR(p) by computing the sum and finding its reciprocal
3. Compute the conjugate of the transfer function: $|H(e^{j\omega})|^2$
4. Multiply it with the variance σ_x^2

The method above is implemented in the function ar2psd (see at the end of document):

```
[S, w] = ar2psd(a, v, 256);
plot(w/pi, real(pow2db(S)))
legend('PSD for AR(2)')
xlabel('\omega/\pi')
```

```
ylabel('S(\omega) in db')
```



Functions

```
function r=acrs(x, L)
% Computes the ACRS r[m] for 0<= m <= L
% r=acrs(x-mean(x),L) yields the ACVS
N=length(x);
x1=zeros(N+L-1,1);
x2=x1;
x1(1:N,1)=x;
for m=1:L
    x2=zeros(N+L-1,1);
    x2(m:N+m-1,1)=x;
    r(m)=x1'*x2;
```

```

    end
    r=r(:)/N;
end

function [a, v] = ar_from_acrs(r_xx, p)
    R_xx = toeplitz(r_xx(1:p));
    % Select elements r_xx[1] to r_xx[p]
    r = r_xx(2:p+1);
    % Solve Yule-Walker equation (13.143)
    a = mldivide(R_xx, -r);
    % Compute the variance according to Eq. (13.144)
    v = r_xx(1) + a'*r;
end

function I=psdper(x, K)
    % Compute periodogram I(ω) using the FFT.
    % K-point FFT >= N
    N=length(x);
    X=fft(x,K);
    I=X.*conj(X)/N;
    I(1)=I(2); % Avoid DC bias
    I=I(:);
end

function r=acrsfft(x, L)
    % Compute the autocorrelation sequence using the FFT.
    % r=acrsfft(x-mean(x),L) yields the ACVS
    N=length(x);
    Q=nextpow2(N+L);
    X=fft(x,2^Q);
    r0=real(ifft(X.*conj(X)));
    r=r0(1:L)/N;
end

function I=psdmodper(x, K)
    % Compute the modified periodogram PSD estimate.
    % K-point FFT >= N
    N=length(x);
    w=hann(N); % choose window
    w=w/(norm(w)/sqrt(N)); % sum w^2[k]=N
    X=fft(x(:).*w(:,K));
    I=X.*conj(X)/N;
    I(1)=I(2); % Avoid DC bias
    I=I(:);
end

function S=psdbt(x, L, K)
    % Compute the Blackman-Tukey PSD estimate.
    % Blackman-Tukey PSD estimator of S(2*pi*k/K)
    if size(x,1) < size(x,2)
        x = x';
    end
    N=length(x);
    w=hann(N); % Data window
    w=w/(norm(w)/sqrt(N)); % sum w^2[k]=N
    x=x.*w; % Data windowing
    r=acrsfft(x,L);
    wc=parzenwin(2*L-1); % Lag window

```

```

rw=r.*wc(L:2*L-1); % Lag windowing
g=zeros(K,1);
g(1:L)=rw;
g(K-L+2:K)=flipud(rw(2:L));
G=fft(g,K); % f even => F real
S=2*real(G(1:K/2));
S(1)=S(2);
end

function S=psdwelch(x, L, K)
% Compute the Welch PSD estimate.
% Welch PSD estimator of S(2*pi*k/K)
M=fix((length(x)-L/2)/(L/2)) % 50% overlap
time=(1:L)';
I=zeros(K,1);
w=hanning(L); % Choose window
w=w/(norm(w)/sqrt(L)); % sum w^2[k]=L
for m=1:M
%xw=w.*detrend(x(time)); % detrending
xw=w.*x(time); % data windowing
X=fft(xw,K);
I=I+X.*conj(X);
time=time+L/2;
end
I=I/(M*L); S=2*I(1:K/2); S(1)=S(2);
end

function [a,v] = arfit(x,p)
% fit AR(p) model from data
% x: data
% p: model order
% a: a coefficients
% v: variance
if isrow(x)
x = x'; % Convert to a column vector
end

% Compute the autocorrelation
[r_xx, lags] = xcorr(x, p, 'biased');

% Select elements r_xx[0] to r_xx[p-1]
R_elems = r_xx(p+1:2*p);

% Create the Toeplitz matrix
R = toeplitz(R_elems);

% Select elements r_xx[1] to r_xx[p]
r = r_xx(p+2:2*p+1);

% Solve systems of linear equations using mldivide function
a = mldivide(R, -r);

% Compute the variance
v = r_xx(p+1) + a'*r;
end

function [S, w] = ar2psd(a, v, N)
% AR2PSD Compute the Power Spectral Density from AR(p) coefficients

```

```

% [S, w] = ar2psd(a, v, N)
% a: AR(p) coefficients
% v: the variance
% N: number of points in the range [1, pi]
% S: the estimated power spectrum
% w: frequencies
w = linspace(0, 1, N) * pi;

% Compute the transfer function
% Used Eq. (13.133) in the book
H = ones(N, 1);
for k=1:numel(a)
    H = H + a(k)*exp(-1j * w' * k);
end
H = 1./H;

% Finally compute the PSD
S = v * H.*conj(H);
end

function [psd, f] = psdmv(x, M, kF)
% Estimate PSD using Capon's Minimum Variance method
% x: input signal
% M: autocorrelation matrix size
% kF:frequency resolution multiplier
if nargin < 3
    kF = 4;
end
if nargin < 2
    M = 64;
end
[rx, ~] = xcorr(x, M-1, 'biased');
Rx=toeplitz(rx(M:end));% Autocorrelation matrix
RxInv=inv(Rx);
Nf = kF*M; % frequency resolution, Nf >= M
f=0.5/Nf*(0:Nf-1); % freq spectrum
psd=zeros(M,1); % pre-alloc
t=(0:M-1)';
for q=1:Nf
    Ef=exp(1i*2*pi*f(q)*t);
    psd(q)=M/(Ef'*RxInv*Ef);
end
f = 2*pi*f;
end

function h = optfilter(x, M, f)
% Compute the optimum filter for estimating a given frequency
% Computes Eq. (14.4.10) in the Minimum Variance material
% x: input signal
% M: autocorrelation matrix size (default: 64)
% f: the given frequency
if nargin < 2
    M = 64;
end
[rx,~]=xcorr(x, M-1, 'biased');
Rx=toeplitz(rx(M:end));
invRx=inv(Rx);
Ef = zeros(M, 1);

```

```
for t=1:M
    % Compute E(f) vector
    Ef(t)=exp(1i*2*pi*f*(t-1));
end
h = (sqrt(M)*invRx*Ef) / (Ef'*invRx*Ef);
end
```

Random Processes

Table of Contents

Random Processes.....	1
Stationary Process.....	1
Wide-sense Stationary Process.....	2
Autocorrelation of wide-sense stationary process.....	2
Examples of non-stationary signals.....	2
Examples of stationary signals.....	2
Problem 13.9: A stationary sinusodial random process.....	3
a) Determine the mean sequence.....	3
b) Determine the autocovariance sequence ACVS.....	5
c) Comment on the stationarity of the random process.....	7

Random Processes

A random process can be thought of a bin that contains multiple signals of infinite length. Every time, we perform a random experiment, we randomly pick one *realization* of the process.



Why work with random processes? Because we need a mathematical description of the random nature of the process that generated the observed values of a given signal.

We cannot predict the effects of an LTI system on any specific realization of the input process. But we can accurately predict its effect on **the average properties**. In other words, since we cannot have a mathematical model of random signals, we create a statistical model.

Stationary Process

A stationary random process is a process which characteristics that do not change over time i.e., for different values of n .

This implies:

- The mean and the variance of a signal $x[n]$ randomly picked from the stationary process does not depend on time n but is constant:

$$E(x[n]) = m_x \quad \text{and} \quad \text{var}(x[n]) = \sigma_x^2, \text{ for all } n. \quad (13.72)$$

b) For two signals $x[n]$ and $x[m]$ randomly picked from the stationary process, the autocorrelations and autocovariance only depend on the lag ℓ and not time:

$$c_{xx}[n, m] \triangleq \text{cov}(x[n], x[m]) = c_{xx}[\ell]. \text{ for all } m, n \quad (13.73)$$

Wide-sense Stationary Process

Wide-sense stationary (WSS): A random process that satisfies both a) and b) are called *wide-sense stationary* or *second-order stationary*.

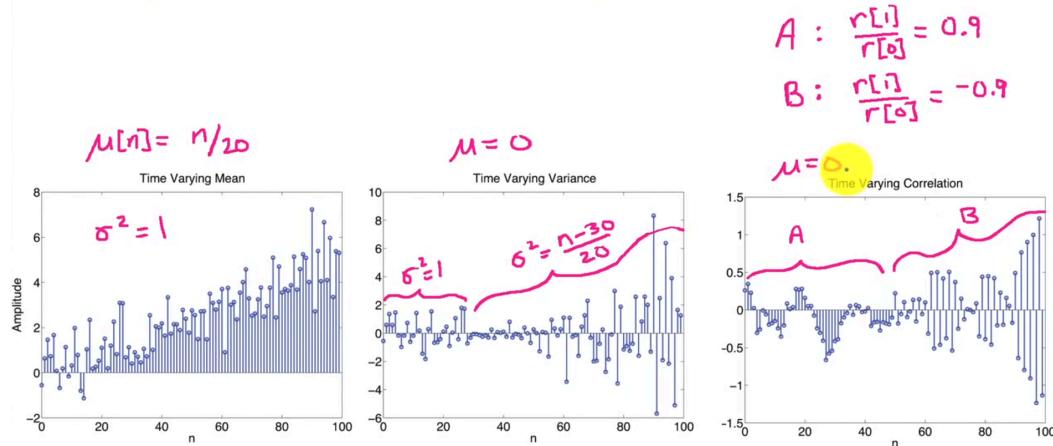
Autocorrelation of wide-sense stationary process

The *autocorrelation sequence* (ACRS) of a wide-sense stationary process is

$$r_{xx}[m + \ell, m] \triangleq E(x[m + \ell]x[m]) = r_{xx}[\ell] = c_{xx}[\ell] + m_x^2 \quad (13.74)$$

Examples of non-stationary signals

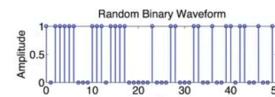
Examples of Nonstationary Signals 3



Examples of stationary signals

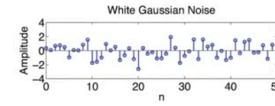
Examples of Stationary Signals -

1) Random Binary Waveform $x[n] = \begin{cases} 0 & P=1/2 \\ 1 & P=1/2 \end{cases}$
 $\mu = 1/2, \quad r[k] = \begin{cases} 1/2 & k=0 \\ 1/4 & k \neq 0 \end{cases} E[x^2[n]] = \sum_{n=0}^{\infty} x[n]^2 = 1/2$



2) White Gaussian Noise $w[n] \sim N(0, \sigma_w^2)$
 independent samples

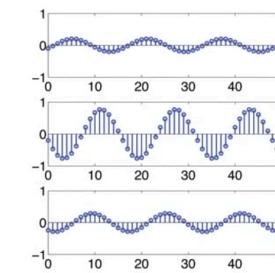
$$\mu = 0, \quad r[k] = \begin{cases} \sigma_w^2 & k=0 \\ 0 & k \neq 0 \end{cases}$$



3) Random Sinusoid $y[n] = A \cos(\omega_0 n + \phi)$

$$A \sim N(0, \sigma_A^2); \quad \phi \sim U(0, 2\pi)$$

$$\mu = 0, \quad r[k] = E \left\{ A^2 \cos(\omega_0 n + \phi) \cos(\omega_0 (n-k) + \phi) \right\} \\ = \frac{\sigma_A^2}{2} \cos(\omega_0 k)$$



Problem 13.9: A stationary sinusoidal random process

9. A random process $x[n]$ is characterized by

$$x[n] = A(\zeta) \cos [\Omega(\zeta)n + \Theta(\zeta)],$$

where random variables $A(\zeta)$, $\Omega(\zeta)$, and $\Theta(\zeta)$ are mutually independent. Random variables $A(\zeta) \sim U(0, 1)$ and $\Theta(\zeta) \sim U(-\pi, \pi)$ are of continuous type while $\Omega(\zeta)$ is of discrete type taking values 10 and 20 radians with equal probability.

a) Determine the mean sequence $m_x[n]$

We need to compute:

$$E[x[n]] = E[A \cos(\Omega n + \Theta)]$$

Since the three random variables are independent then we can write:

$$E[x[n]] = E[A]E[\cos(\Omega n + \Theta)]$$

We know that $E[A] = \frac{1}{2}$ since $A \sim U(0, 1)$.

To find the expected value of a function, we need to compute the integral of the function weighted by density of the random quantity $f_\Theta(\Theta)$. Formally, we can write:

$$E[\cos(\Omega n + \Theta)] = \int_{-\pi}^{\pi} f_\Theta(\Theta) \cdot \cos(\Omega n + \Theta) d\Theta$$

Since Θ is uniform between $-\pi$ and π , the density is:

$$f_\Theta(\Theta) = \frac{1}{2\pi}$$

So the expectation of the function becomes:

$$E[\cos(\Omega n + \Theta)] = \int_{-\pi}^{\pi} \frac{1}{2\pi} \cos(\Omega n + \Theta) d\Theta$$

We can bring out the constant outside the integral:

$$E[\cos(\Omega n + \Theta)] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \cos(\Omega n + \Theta) d\Theta$$

The integral of $\cos(n)$ is $\sin(n)$ so we have:

$$E[\cos(\Omega n + \Theta)] = \frac{1}{2\pi} [\sin(\Omega n + \Theta)]_{-\pi}^{\pi} = \frac{1}{2\pi} [\sin(\Omega n + \pi) - \sin(\Omega n - \pi)]$$

Looking up the trigonometric identities:

Shift by one quarter period	Shift by one half period ^[9]	Shift by full periods ^[10]	Period
$\sin(\theta \pm \frac{\pi}{2}) = \pm \cos \theta$	$\sin(\theta + \pi) = -\sin \theta$	$\sin(\theta + k \cdot 2\pi) = +\sin \theta$	2π
$\cos(\theta \pm \frac{\pi}{2}) = \mp \sin \theta$	$\cos(\theta + \pi) = -\cos \theta$	$\cos(\theta + k \cdot 2\pi) = +\cos \theta$	2π
$\tan(\theta \pm \frac{\pi}{4}) = \frac{\tan \theta \pm 1}{1 \mp \tan \theta}$	$\tan(\theta + \frac{\pi}{2}) = -\cot \theta$	$\tan(\theta + k \cdot \pi) = +\tan \theta$	π
$\csc(\theta \pm \frac{\pi}{2}) = \pm \sec \theta$	$\csc(\theta + \pi) = -\csc \theta$	$\csc(\theta + k \cdot 2\pi) = +\csc \theta$	2π
$\sec(\theta \pm \frac{\pi}{2}) = \mp \csc \theta$	$\sec(\theta + \pi) = -\sec \theta$	$\sec(\theta + k \cdot 2\pi) = +\sec \theta$	2π
$\cot(\theta \pm \frac{\pi}{4}) = \frac{\cot \theta \pm 1}{1 \mp \cot \theta}$	$\cot(\theta + \frac{\pi}{2}) = -\tan \theta$	$\cot(\theta + k \cdot \pi) = +\cot \theta$	π

We see that $\sin(\Omega n + \pi) = -\sin(\Omega n)$ and $-\sin(\Omega n - \pi) = \sin(\Omega n)$. Substituting these identities we get:

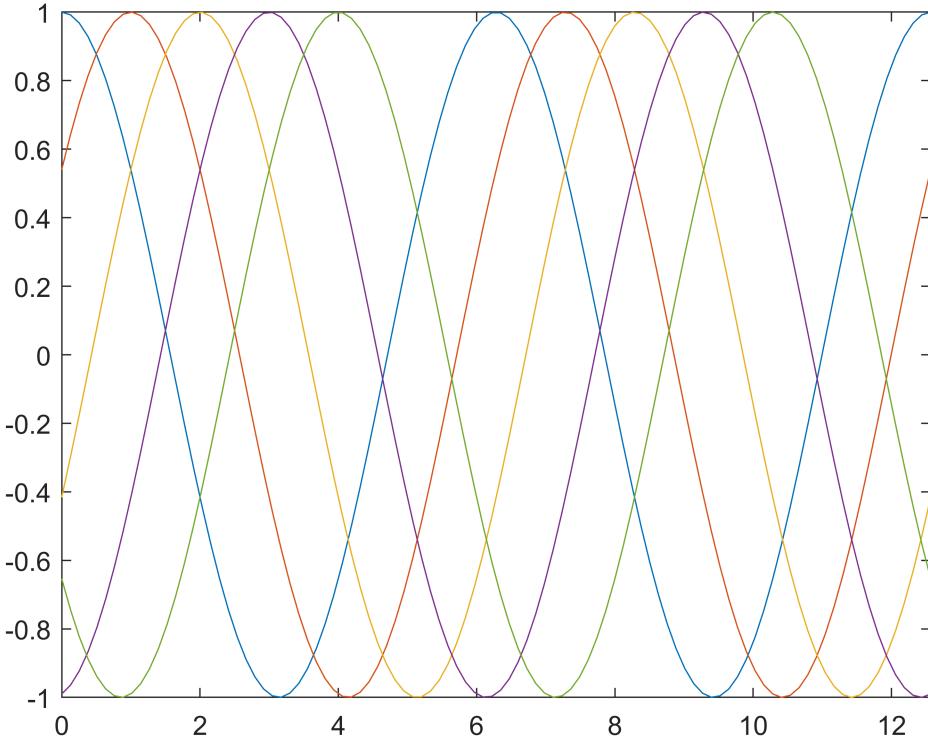
$$E[\cos(\Omega n + \Theta)] = \frac{1}{2\pi} [-\sin(\Omega n) + \sin(\Omega n)] = 0$$

Therefore: $E[\cos(\Omega n + \Theta)] = 0$

We can visualise why the expected value of $\cos(\Omega n + \Theta)$ is zero.

Suppose we want to compute $E[\cos(\Omega n + \Theta)]$ where $\Theta \sim U(-\pi, \pi)$. Let us pick one frequency ω (realise one value of Ω). Then let us pick a lot of realisations of Θ . Now if we plot the function $\cos(\omega n + \theta)$ for different values of θ then we will see something like this:

```
n = linspace(0, 4*pi);
plot(n, cos(n), n, cos(n-1), n, cos(n-2), n, cos(n-3), n, cos(n-4));
xlim([0, 4*pi]);
```



If we plot hundreds of cosine functions shifted slightly, we get a large blob of points from -1 to 1. For this reason, the quantity $E[\cos(\Omega n + \Theta)]$ will be zero because the mean value is 0.

In signal processing, we like to add random shifts ala $\Theta \sim U(0, 2\pi)$ to avoid that the expected value or the mean value becomes dependent on time.

b) Determine the autocovariance sequence ACVS $c_X[m, n]$

The ACVS provides a measure of the linear dependence between the values of a random process at two different times. In this sense, it determines how quickly the signal amplitude changes from sample to sample (time variation).

To compute the autocovariance sequence, we use the following equation:

$$r_{xy}[n, m] \triangleq E[x[n]y[m]] = c_{xy}[n, m] + E(x[n])E(y[m]). \quad (13.75b)$$

The formula relates the cross-correlation $r_{xy}[n, m]$ and cross-covariance $c_{xy}[n, m]$. It states that the cross-correlation is the cross-covariance added with the product of the means of the two signals.

From a) we know that the expected value (mean) of the given random process is zero. This essentially means that the cross-correlation and the cross-covariance are the same.

This is also clear from the definition of the cross-covariance and cross-correlation.

The cross-covariance for a random process defined as:

$$c_{xy}[n, m] = E\{(x[n] - E\{x[n]\})(y[m] - E\{y[m]\})\}$$

The cross-correlation for a random process is defined as:

$$r_{xy}[n, m] = E\{x[n]y[m]\}$$

The term “auto” comes about because the two random variables are taken from the same process.

Note that when $E\{x[n]\} = E\{y[m]\} = 0$ then $c_{xy}[n, m] = r_{xy}[n, m]$.

```
syms A
f = A^2;
int(f, A, 0, 1)
```

ans =

$$\frac{1}{3}$$

The result from the solution set:

$$\begin{aligned} c_X[m, n] &= r_X[m, n] = E[A \cos(\Omega m + \Theta) A \cos(\Omega n + \Theta)] \\ &= E\left[\frac{A^2}{2} \{\cos[\Omega(m+n) + 2\Theta] + \cos[\Omega(m-n)]\}\right] \text{ [a]} \\ &= \frac{1}{2} E[A^2] \{E[\cos \Omega(m+n)]E[\cos 2\Theta] - E[\sin \Omega(m+n)]E[\sin 2\Theta]\} \text{ [b]} \\ &\quad + E[\cos \Omega(m-n)] \\ &= \frac{1}{2} E[A^2] E[\cos \Omega(m-n)] \end{aligned}$$

$$E[A^2] = \int_0^1 a^2 da = \frac{1}{3} \text{ [c]}$$

$$E[\cos(m-n)\Omega] = \frac{1}{2} \cos 10(m-n) + \frac{1}{2} \cos 20(m-n)$$

Hence, the ACVS $c_X[m, n]$ is

$$c_X[m, n] = \frac{1}{12} \cos 10(m-n) + \frac{1}{12} \cos 20(m-n)$$

where $\ell = m - n$

Following identities were used in the solution:

$$\text{[a]} \cos(\alpha)\cos(\beta) = \frac{1}{2}(\cos(\alpha + \beta) + \cos(\alpha - \beta))$$

[b] $\cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$

[c] $E[X^n] = \int_{-\infty}^{\infty} x^n \cdot f_X(x) dx$

c) Comment on the stationarity of the random process

First, let us defined what the stationarity of a random process is.

A random process is said to be stationary if its characteristics does not change over time. This means that:

- i) The mean and the variance of the process is constant:

$$E(x[n]) = m_x \quad \text{and} \quad \text{var}(x[n]) = \sigma_x^2, \text{ for all } n. \quad (13.72)$$

- ii) The auto-correlation and auto-covariance only depend on the lag ℓ :

$$c_{xx}[n, m] \triangleq \text{cov}(x[n], x[m]) = c_{xx}[\ell], \text{ for all } m, n \quad (13.73)$$

If a random process fulfills both requirements, then it is said to be wide-sense stationary.

In a) we showed the mean of the process is zero.

In b) we showed that the autocovariance only depends on the lag $\ell = m - n$.

Therefore, we can conclude that the random process $x[n]$ is wide-sense stationary.

Notice this:

Example 13.2 Sinusoidal random process

We can easily create a sinusoidal random process by making the amplitude, frequency, or phase of a sinusoidal sequence as random variables

$$x[n, \zeta_k] = A(\zeta_k) \cos[\omega(\zeta_k)n + \phi(\zeta_k)]. \quad (13.77)$$

This is a fixed-form random process because each realization has the same shape, which is controlled by a finite number of parameters. Clearly, the properties of the entire process cannot be obtained from a single realization $x[n, \zeta_k]$. Finally, since the values of any realization $x[n, \zeta_k]$ for $n > n_0$ can be determined from the values for $n \leq n_0$ the process is called predictable. The essential randomness in (13.77) is not in the character of individual realizations, but in the particular choice of the set of values (A, ω, ϕ) . **This process, in general, is not stationary** (see Tutorial Problem 9). ■

The result of tutorial problem 9 is obtained for one particular way of generating the randomness of the amplitude, phase and frequencies. You can generate other random processes where the stationarity breaks down. One good example I often use it to constrain the randomness of the phase. In tutorial problem 9 (and for that matter any other time we have used sinusoidal signals in the course) we have chosen the phase as being uniformly distributed between 0

and 2π . If you instead constrain the phase to be, say a uniformly random number between 0 and 0.1 , the process is no longer stationary. You can convince yourself of this result by drawing two figures with many sinusoids with the same amplitude and frequency, but with either choice of phase.

In all the problems, we have looked at in the course, we always use the 0 to 2π choice and we consider sinusoids as stationary.

Signal Modelling

Table of Contents

Signal Modelling.....	1
Regular processes.....	1
Paley-Wiener condition.....	2
Modelling signals with line spectra.....	3
Modelling signals with bandwidth limited spectra.....	4
Wold decomposition.....	4

Signal Modelling

Signal modelling is about gaining some understanding of certain signals so we can model them. We do not need to have a full suite of samples but we want to understand from a statistical perspective.

White noise a building blocks that we can use.

Signal modelling is about taking some unknown signal $y(n)$ and coming up with an LTI system that generates $y(n)$ given some white noise input.

All the correlation that was between samples before are stored in the filter coefficients. This becomes a model of our signal. If we know the filter, we know the signal. The samples of the signal will not be the same but the signal will have the same statistical properties.

We cannot generate the exact same $y(n)$ using different white noise inputs. Instead we generate signals with the **same statistical properties**.

Regular processes

A regular process is just a minimum-phase LTI system given by the difference equation:

$$y[n] = \sum_{k=0}^{\infty} h[k]x[n-k] \quad (13.129)$$

This is just a FIR filter with infinitely many coefficients.

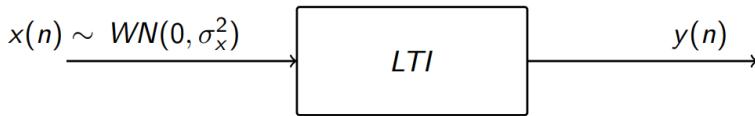
The filter $h[n]$ is known as a **synthesis** or **colouring** filter.

Since the LTI system is minimum-phase, it is causal, stable and therefore also invertible.

$$x[n] = \sum_{k=0}^{\infty} h_{\text{inv}}[k]y[n-k] \quad (13.130)$$

This system is known as an **analysis** or **whitening** filter.

If feed this minimum-phase LTI system a white noise signal, we can compute the autocorrelation and PSD of the resulting process:



The autocorrelation of the input signal (white noise) is:

$$r_{xx}(\ell) = \sigma_x^2 \delta(\ell), \quad \text{where } \sigma_x^2 = E[x^2(n)] \text{ is the variance}$$

The power spectral density of the input signal is:

$$S_{xx}(\omega) = \sigma_x^2$$

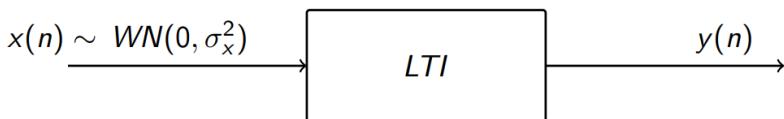
The autocorrelation of the output signal is:

$$r_{yy}(\ell) = \sigma_x^2 r_{hh}(\ell)$$

The power spectral density of the output signal is power transfer function of the LTI system:

$$S_{yy}(\omega) = \sigma_x^2 |H(e^{j\omega})|^2$$

Finding the synthesis filter $h[m]$ from the autocorrelation or PSD is known as **spectral factorization**. This is essentially what signal modelling is about; taking some unknown signal $y[n]$ and coming up with a minimum-phase LTI system that generates $y(n)$ given some white noise input.



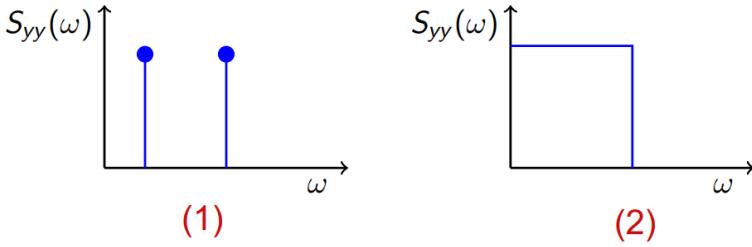
Paley-Wiener condition

A process is regular if its PSD satisfies the Paley–Wiener condition:

$$\int_{-\pi}^{\pi} |\ln S_{yy}(\omega)| d\omega < \infty. \quad (13.131)$$

There are two kind of signals that cannot be modelled by an LTI system driven by white noise:

1. signal with line spectra e.g. sinusoids $\cos(\omega)$
2. bandwidth limited spectra



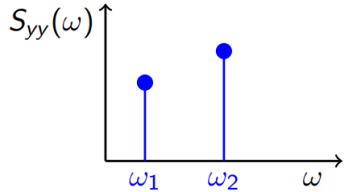
We cannot model these types of signals because it does not fulfill the Paley-Wiener condition:

$$\int_{-\pi}^{\pi} |\ln S_{yy}(\omega)| d\omega < \infty$$

Modelling signals with line spectra

Signals with line spectra are easy to model.

Suppose we want to model following signal:



We can just use a **harmonic process** to model the signal:

$$x[n] = \sum_{k=1}^p A_k \cos(\omega_k n + \phi_k)$$

where

- $\omega_k \neq 0$
- p is a constant denoting the number of frequency components
- A_1, \dots, A_p are constants denoting amplitudes
- $\omega_1, \dots, \omega_p$ are constants denoting frequencies
- ϕ_1, \dots, ϕ_p are pairwise independent random variables uniformly distributed in the interval $(0, 2\pi)$

This model has $3p$ parameters A_k , ω_k and ϕ_k that we need to estimate. We can find these parameters using the least-squares.

The harmonic process is wide-sense stationary with mean value zero:

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

The autocorrelation of the harmonic process is:

$$r_{xx}(\ell) = \frac{1}{2} \sum_{k=1}^p A_k^2 \cos(\omega_k \ell)$$

Take the Fourier Transform, we get the power spectral density:

$$S_{xx}(\omega) = \sum_{k=1}^p 2\pi \frac{A_k^2}{4} [\delta(\omega - \omega_k) + \delta(\omega + \omega_k)]$$

Modelling signals with bandwidth limited spectra

If we want to compute a filter with the brick-wall shape (2), we need an infinitely long non-causal filter. However, we said that we only wanted minimum-phase filters i.e. causal and stable filter. Therefore, this is not going to work.

Wold decomposition

The Wold decomposition says: *any stationary discrete-time process can be expressed as the sum of two **uncorrelated** processes, one regular process and one harmonic process.*

Wiener filters

Table of Contents

Filters that minimize the output mean square error.....	1
Wiener filter.....	2
Applications of Wiener filtering.....	3
True/False Problems.....	3
Does the Least Mean Square adaptive filter converge to Wiener-Hopf Solution?.....	4
Exam 2015 Problem 3: Recover signal by a Wiener filter.....	4
1) Design a 4 tap Wiener filter to recover $s(n)$	5
2) Calculate the minimum mean square error (MSE).....	6
3) Can the MSE be lowered using a longer Wiener filter?.....	7
4) Calculate SNR before and after the Wiener filter.....	7
5) What happens when the noise amplitude is twice as large?.....	8
Exam 2017 Problem 4: Recover signal using a Wiener filter.....	9
1) Compute the autocorrelation function of the noise.....	9
2) Compute the optimum filter coefficients.....	10
3) Calculate SNR before and after the Wiener filter.....	11
Exam 2018 Problem 4: Recover signal using a Wiener filter.....	12
1) Determine the autocorrelation function,.....	12
2) Solve the Wiener-Hopf Equation.....	13
3) Discuss whether 3 taps is an optimum choice for this problem?.....	14
Exam 2018 Problem 1: Model a system using a Wiener filter.....	15
1) Calculate the autocorrelation matrix and cross-correlation vector.....	16
2) Calculate the expected energy.....	17
3) Solve for $H(z)$ and comment on the influence of $v(n)$	17
Exam 2018 Problem 4: Recover signal using a Wiener filter.....	18
[✓] 1) Determine the autocorrelation function,.....	18
[✓] 2) Solve the Wiener-Hopf Equation.....	19
[✓] 3) Discuss whether 3 taps is an optimum choice for this problem?.....	21
ADSI Problem 6.3: Recover AR(1) signal using Wiener filter.....	22
1) Determine the autocorrelation sequence for a signal with two noise processes.....	22
2) Design a Wiener filter of length $M=2$ to estimate an AR(1) process.....	23
3) Determine the minimum mean square error for $M=2$	24
Exam 2017 Problem 4: Recover signal using a Wiener filter.....	25
1) Compute the autocorrelation function of the noise.....	25
2) Compute the optimum filter coefficients.....	26
3) Calculate SNR before and after the Wiener filter.....	27

Filters that minimize the output mean square error

Problem: using a linear estimator, we want to guess the value of random variable y from a set of observations of a related set of p random variables x_1, x_2, \dots, x_p :

$$\hat{y} = \sum_{k=1}^p h_k x_k = \mathbf{h}^T \mathbf{x}. \quad (14.100)$$

This is basically a multivariate linear regression problem where \mathbf{h} is the coefficients of the model. In the multivariate regression problem, we hypothesise that the response random variable Y can be computed as a linear combination of the coefficients of the model and the predictor variables X_1, X_2, \dots, X_p

$$\hat{y}_h(\mathbf{x}) = \mathbf{h}^T \mathbf{x}$$

The mean squared error loss function is:

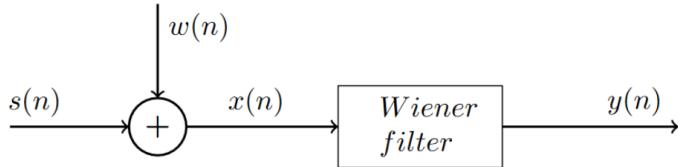
$$J_h = \frac{1}{2N} \sum_{i=1}^N (\hat{y}_h(\mathbf{x}_i) - y_i)^2$$

We can minimise the loss function by taking the derivative, setting it zero and solving the equation.

Wiener filter

A special case of the multivariate linear regression model is the Wiener filter, which is used to estimate the original signal $s[n]$ given p observations of a noise distorted signal $x[n] = s[n] + w[n]$.

We assume that the noise process $w[n]$ is uncorrelated with the desired process that generated the original signal $s[n]$. We also assume that $x[n]$ and $w[n]$ are wide-sense stationary.



An p 'order Wiener filter for estimating the original signal $s(n)$ is given by Eq. 14.112:

$$\hat{y}[n] = \sum_{k=1}^p h_k x[n+1-k] \quad (14.112)$$

The normal equation for the Wiener filter is given by Eq. 14.113

$$\begin{bmatrix} r_x[0] & r_x[1] & \dots & r_x[p-1] \\ r_x[1] & r_x[0] & \dots & r_x[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_x[p-1] & r_x[p-2] & \dots & r_x[0] \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_p \end{bmatrix} = \begin{bmatrix} r_{yx}[0] \\ r_{yx}[1] \\ \vdots \\ r_{yx}[p-1] \end{bmatrix}, \quad (14.113)$$

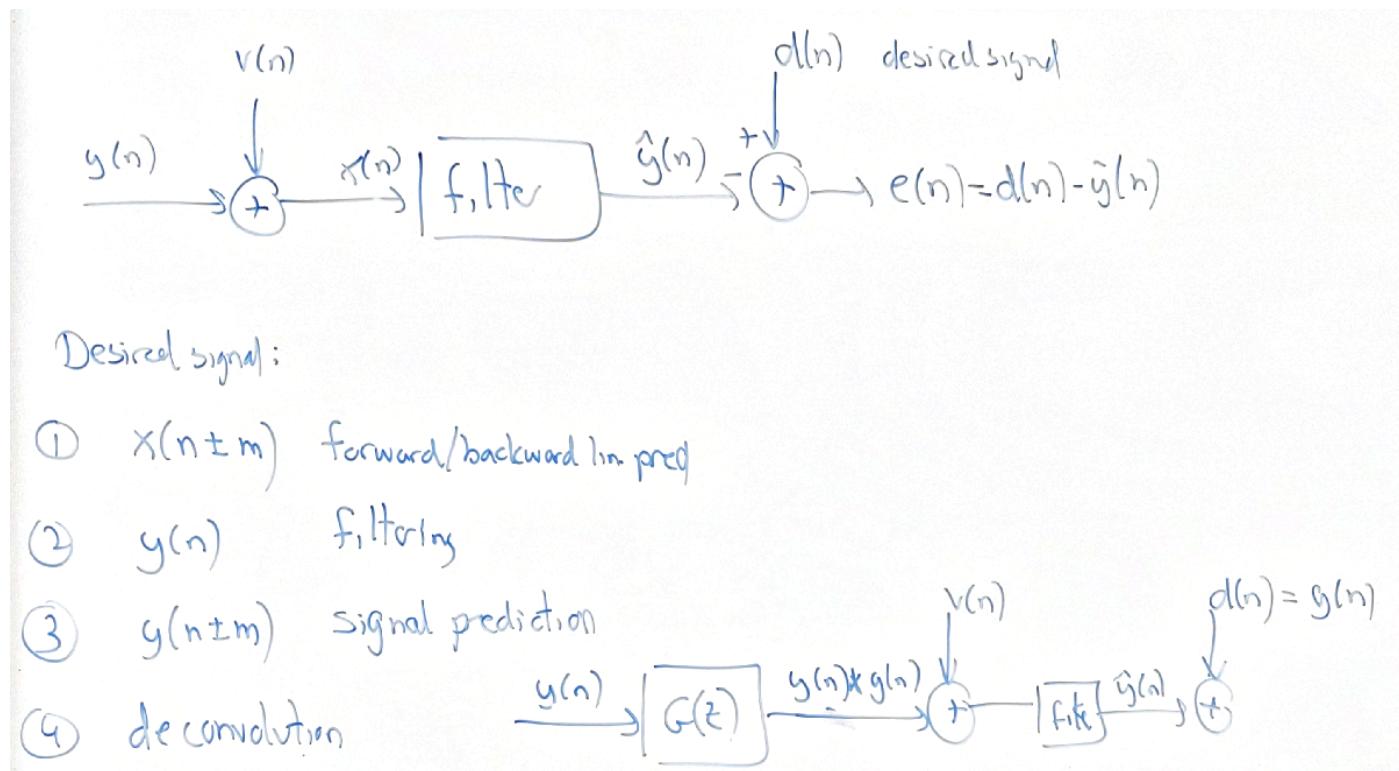
where $h_k = h_o[k-1]$

The minimum square error for a p th Wiener filter is given by Eq. 14.115:

$$J_o = r_y[0] - \mathbf{h}_o^T \mathbf{g} = r_y[0] - \sum_{k=0}^{p-1} h_o[k] r_{yx}[k]. \quad (14.115)$$

The optimum Wiener filter "passes" the input at bands with high SNR and "blocks" the input at bands with low SNR (see Eq. 14.120)

Applications of Wiener filtering



True/False Problems

Does the Least Mean Square adaptive filter converge to Wiener-Hopf Solution?

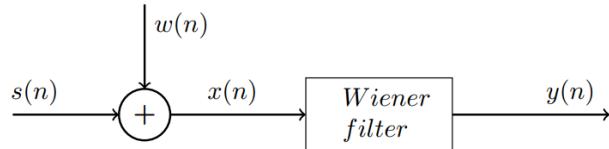
Under stationary conditions, a least mean square (LMS) adaptive filter with proper choice of the step-size will converge to the Wiener-Hopf solution.

The third statement is **true**. The Wiener-Hopf solution minimizes $E[e^2(n)]$. In the LMS adaptive filter the instantaneous error $e^2(n)$ is minimized. Under stationary conditions the adaptive filter effectively averages the instantaneous error and the filter will converge towards the Wiener-Hopf solution.

Exam 2015 Problem 3: Recover signal by a Wiener filter

```
clear variables;
```

A signal $s(n)$ is created as an AR(2) process. It is corrupted by uncorrelated, additive noise $w(n)$ as shown in the figure.



It is desired to recover the signal by a Wiener filter. The AR(2) process has two complex poles at $re^{\pm j\theta}$. For this kind of process the autocorrelation of the signal can be calculated analytically and is given by

$$r_s(l) = \frac{r^l (\sin((l+1)\theta) - r^2 \sin((l-1)\theta))}{((1-r^2) \sin \theta)(1-2r^2 \cos 2\theta + r^4)}, \quad l \geq 0$$

For simplicity the first few values of the autocorrelation function for this specific process is given by

$ l $	$r_s(l)$
0	7.89
1	5.55
2	0.67
3	-3.64
4	-5.18
5	-3.64

```
r_ss = [7.89, 5.55, 0.67, -3.64, -5.18, -3.64];
```

Less information is known about the noise. The autocorrelation for the first few lags are given by:

$ l $	$r_w(l)$
0	5
1	3
2	1

For all other lags, the autocorrelation, $r_w(l)$ is assumed to be zero.

```
r_ww = [5, 3, 1, 0, 0, 0];
```

1) Design a 4 tap Wiener filter to recover $s(n)$.

The p -order Wiener filter for estimating the signal $s(n)$ is given by:

$$\hat{y}[n] = \sum_{k=1}^p h_k x[n+1-k] \quad (14.112)$$

The optimum Wiener filter coefficients is given by the Wiener-Hopf equation:

$$\mathbf{h}_o = \mathbf{R}_x^{-1} \mathbf{g}, \quad (14.109)$$

where \mathbf{R}_x is the autocorrelation matrix of the corrupted signal $s(n)$ and \mathbf{g} is the cross-correlation between the desired signal $s(n)$ and the corrupted signal $x(n)$.

Designing a Wiener filter to recover a corrupted signal involves 3 steps:

1. Compute the autocorrelation sequence $r_x(\ell)$ and matrix R_x
2. Compute the cross-correlation $r_{sx}(\ell)$
3. Solve the Wiener-Hopf equation to find the optimum Wiener filter coefficients

Step 1: Compute the autocorrelation $r_x(\ell)$:

Since the signal $s(n)$ and the noise $w(n)$ are uncorrelated the autocorrelation function of $s(n)$ is just the sum of the individual autocorrelation functions:

$$\begin{aligned} rx(\ell) &= E[x(n)x(n-\ell)] \\ &= E[(s(n)+w(n))(s(n-\ell)+w(n-\ell))] \\ &= E[s(n)s(n-\ell) + s(n)w(n-\ell) + w(n)s(n-\ell) + w(n)w(n-\ell)] \\ &= r_s(\ell) + r_{sw}(\ell) + r_{ws}(\ell) + r_w(\ell) \\ &= r_s(\ell) + r_w(\ell) \text{ (assuming } s(n) \text{ and } w(n) \text{ are uncorrelated)} \end{aligned}$$

```
p = 4;
```

```
r_xx = r_ss + r_ww;
R_xx = toeplitz(r_xx(1:p))
```

```
R_xx = 4x4
12.8900    8.5500    1.6700   -3.6400
 8.5500   12.8900    8.5500    1.6700
 1.6700    8.5500   12.8900    8.5500
 -3.6400    1.6700    8.5500   12.8900
```

Step 2: Compute the cross-correlation $r_{sx}(\ell)$:

Since the desired signal is $s(n)$, the cross-correlation between $s(n)$ and $x(n)$ simplifies to the autocorrelation of the signal $r_s(\ell)$:

$$\begin{aligned} r_{sx}(\ell) &= E[s(n)x(n-\ell)] \\ &= E[s(n)(s(n-\ell) + w(n-\ell))] \\ &= E[s(n)s(n-\ell) + s(n)w(n-\ell)] \\ &= r_s(\ell) + r_{sw}(\ell) \\ &= r_s(\ell) \quad (\text{assuming } s(n) \text{ and } w(n) \text{ are uncorrelated}) \end{aligned}$$

```
g = r_ss(1:p)'
```

```
g = 4x1
 7.8900
 5.5500
 0.6700
 -3.6400
```

Step 3: Compute the optimum filter coefficients:

```
h_opt = R_xx\g
```

```
h_opt = 4x1
 0.4841
 0.1026
 0.0477
 -0.1906
```

2) Calculate the minimum mean square error (MSE)

The minimum value of the mean square error $E[e^2(n)]E[(y(n) - \hat{y}(n))^2]$ is given by

$$J_o = r_y[0] - \mathbf{h}_o^T \mathbf{g} = r_y[0] - \sum_{k=0}^{p-1} h_o[k] r_{yx}[k]. \quad (14.115)$$

```
mse = r_ss(1) - h_opt'*g
```

```
mse = 2.7757
```

3) Can the MSE be lowered using a longer Wiener filter?

Explain whether the minimum mean square error can be lowered by using a longer Wiener filter when we only have sparse knowledge of the noise.

The minimum mean square error can be reduced further by increasing the length of the Wiener filter. As the length of the filter is increased the frequency resolution increases and the spectral shape of the filter becomes better suited for passing the signal and rejecting the noise.

This is also evident if the minimum mean square error is calculated. For a 5 tap filter, the minimum mean square error decreases to 2.6617:

```
p = 5;
r_xx = r_ss + r_ww;
R_xx = toeplitz(r_xx(1:p));
g = r_ss(1:p)';
h_opt = R_xx\g;
mse = r_ss(1) - h_opt'*g
```

```
mse = 2.6617
```

4) Calculate SNR before and after the Wiener filter

The signal to noise ratio of the input signal is given by

$$\text{SNR}_i = \frac{\text{power of signal}}{\text{power of noise}} = \frac{r_s(0)}{r_w(0)}$$

```
SNR_i = r_ss(1) / r_ww(1)
```

```
SNR_i = 1.5780
```

The output SNR of the Wiener filter is given by:

$$\text{SNR}_o = \frac{h_{\text{opt}}^T R_s h_{\text{opt}}}{h_{\text{opt}}^T R_w h_{\text{opt}}}$$

```
R_ss = toeplitz(r_ss(1:p));
R_ww = toeplitz(r_ww(1:p));
SNR_o = (h_opt'*R_ss*h_opt) / (h_opt'*R_ww*h_opt)
```

```
SNR_o = 2.2673
```

The filter increases the signal to noise ratio but only slightly. This is expected and is an excellent way to check your results.

5) What happens when the noise amplitude is twice as large?

Consider a second, identical Wiener filtering problem, except that the noise amplitude is twice as big. Discuss whether this second system will have the same minimum mean square error as the first system.

The expected outcome of an increase of the noise amplitude is an increase of the minimum mean square error as there is more noise to remove.

If the noise $w(n)$ doubles in amplitude, then the values of the autocorrelation function must quadruple:

$$r_{2w}(\ell) = E[2w(n)2w(n-\ell)] = 4E[w(n)w(n-\ell)] = 4r_w(\ell)$$

```
r_xx = r_ss + r_ww;
R_xx = toeplitz(r_xx(1:p));
g = r_ss(1:p)';
h_opt = R_xx\g;
mse = r_ss(1) - h_opt'*g
```

`mse = 2.6617`

```
r_xx = r_ss + r_ww.*4;
R_xx = toeplitz(r_xx(1:p));
g = r_ss(1:p)';
h_opt = R_xx\g;
mse = r_ss(1) - h_opt'*g
```

`mse = 5.1096`

We see that the minimum MSE increases when the noise amplitude is increased.

Another example using following autocorrelation sequences:

l	$r_s(l)$	$r_w(l)$	$r_x(l)$
0	1	1	2
1	-0.4	0.8187	0.4187
2	0.2	0.7536	0.9536

```
p = 3;
r_ss = [1, -0.4, 0.2];
r_ww = [1, 0.8187, 0.7536];
r_xx = r_ss + r_ww;
R_xx = toeplitz(r_xx(1:p));
g = r_ss(1:p)';
h_opt = R_xx\g;
mse = r_ss(1) - h_opt'*g
```

`mse = 0.2777`

```

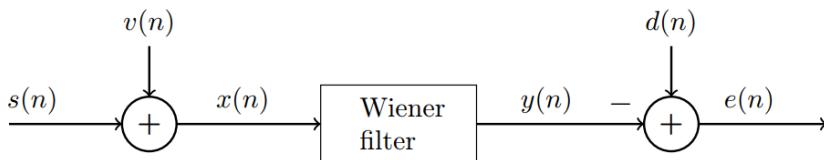
r_xx = r_ss + r_ww.*4;
R_xx = toeplitz(r_xx(1:p));
g = r_ss(1:p)';
h_opt = R_xx\g;
mse = r_ss(1) - h_opt'*g

```

`mse = 0.5081`

Exam 2017 Problem 4: Recover signal using a Wiener filter

Consider the Wiener filtering problem shown below. A signal $s(n)$ is corrupted by additive, uncorrelated noise $v(n)$ giving the signal $x(n) = s(n) + v(n)$. It is desired to use a 2-tap Wiener filter to recover the signal i.e. $d(n) = s(n)$.



The noise $v(n)$ is generated in an MA(1) process as

$$v(n) = w(n) + w(n-1)$$

where $w(n)$ is a zero-mean gaussian white noise sequence with $\sigma_w^2 = 1$. The autocorrelation function of the signal is

$$r_s(l) = 4 \cdot 0.25^{|l|}.$$

```

clear variables;

M = 2;
ell = 0:M-1;
delta = @(l) ell == 1;

r_ss = 4*(0.25).^(abs(ell));

```

1) Compute the autocorrelation function of the noise

1. Show that the autocorrelation function of the noise $v(n)$ is given by

$$r_v(l) = \delta(l-1) + 2\delta(l) + \delta(l+1).$$

In ADSI Problem 4.9, we found that the autocorrelation for an MA(1) process where the input signal is a white noise with unit variance can be described as:

$$r_v(\ell) = (b_0^2 + b_1^2)\delta(\ell) + b_0b_1(\delta(\ell-1) + \delta(\ell+1))$$

In this problem $b_0 = b_1 = 1$:

$$r_v(\ell) = \delta(\ell - 1) + 2\delta(\ell) + \delta(\ell + 1)$$

```
r_vv = delta(-1) + 2*delta(0) + delta(1)
```

```
r_vv = 1x2
      2      1
```

2) Compute the optimum filter coefficients

2. Calculate the crosscorrelation vector \mathbf{g} , the autocorrelation matrix \mathbf{R}_x and the optimum filter coefficients.

The p -order Wiener filter for estimating the signal $s(n)$ is given by:

$$\hat{y}[n] = \sum_{k=1}^p h_k x[n+1-k] \quad (14.112)$$

The optimum Wiener filter coefficients is given by the Wiener-Hopf equation:

$$\mathbf{h}_o = \mathbf{R}_x^{-1} \mathbf{g}, \quad (14.109)$$

where \mathbf{R}_x is the autocorrelation matrix of the corrupted signal $s(n)$ and \mathbf{g} is the cross-correlation between the desired signal $s(n)$ and the corrupted signal $x(n)$.

Designing a Wiener filter to recover a corrupted signal involves 3 steps:

1. Compute the autocorrelation sequence $r_x(\ell)$ and matrix R_x
2. Compute the cross-correlation $r_{sx}(\ell)$
3. Solve the Wiener-Hopf equation to find the optimum Wiener filter coefficients

Step 1: Compute the autocorrelation $r_x(\ell)$:

Since the signal $s(n)$ and the noise $v(n)$ are uncorrelated the autocorrelation function of $s(n)$ is just the sum of the individual autocorrelation functions:

$$\begin{aligned} r_x(\ell) &= E[x(n)x(n-\ell)] \\ &= E[(s(n) + v(n))(s(n-\ell) + v(n-\ell))] \\ &= E[s(n)s(n-\ell) + s(n)v(n-\ell) + v(n)s(n-\ell) + v(n)v(n-\ell)] \\ &= r_s(\ell) + r_{sv}(\ell) + r_{vs}(\ell) + r_v(\ell) \end{aligned}$$

Since $s(n)$ and $v(n)$ are uncorrelated $r_{sv}(\ell) = 0$. We computed $r_v(\ell) = \delta(\ell - 1) + 2\delta(\ell) + \delta(\ell + 1)$:

$$r_x(\ell) = r_s(\ell) + r_v(\ell)$$

```
r_xx = r_ss + r_vv;
R_xx = toeplitz(r_xx)
```

```
R_xx = 2x2
      6      2
      2      6
```

Step 2: Compute the cross-correlation $r_{sx}(\ell)$:

Since the desired signal is $s(n)$, the cross-correlation between $s(n)$ and $x(n)$ simplifies to the autocorrelation of the signal $r_s(\ell)$.

$$\begin{aligned} r_{yx}(\ell) &= E[s(n)x(n-\ell)] \\ &= E[s(n)(s(n-\ell) + v(n-\ell))] \\ &= E[s(n)s(n-\ell)] + E[s(n)v(n-\ell)] \\ &= r_s(\ell) + r_{sv}(\ell) \\ &= r_s(\ell) + 0 \quad (\text{since } s(n) \text{ and } v(n) \text{ are uncorrelated } r_{sv}(\ell) = 0) \end{aligned}$$

```
g = r_ss'
```

```
g = 2x1
  4
  1
```

Step 3: Compute the optimum filter coefficients:

```
h_opt = R_xx\g
```

```
h_opt = 2x1
  0.6875
 -0.0625
```

3) Calculate SNR before and after the Wiener filter

The signal to noise ratio of the input signal is given by

$$\text{SNR}_i = \frac{\text{(value of signal at } n = n_0)^2}{\text{power of noise}} = \frac{s^2(n = n_0)}{r_v(0)} = \frac{r_s(0)}{r_v(0)}$$

We know that:

$$r_s(\ell) = 4 \cdot 0.25^{|\ell|} \quad \text{and} \quad r_v(\ell) = \delta(\ell - 1) + 2\delta(\ell) + \delta(\ell + 1)$$

This means:

$$r_s(0) = 4 \quad \text{and} \quad r_v(0) = 2$$

Thus:

$$\text{SNR}_i = \frac{r_s(0)}{r_v(0)} = \frac{4}{2} = 2$$

The output SNR of the Wiener filter is given by:

$$\text{SNR}_o = \frac{h_{\text{opt}}^T R_s h_{\text{opt}}}{h_{\text{opt}}^T R_v h_{\text{opt}}}$$

```
R_ss = toeplitz(r_ss);
R_vv = toeplitz(r_vv);
SNR_o = (h_opt'*R_ss*h_opt) / (h_opt'*R_vv*h_opt)
```

```
SNR_o = 2.0991
```

The filter increases the SNR but only slightly. This is expected and is an excellent way to check your results.

Exam 2018 Problem 4: Recover signal using a Wiener filter

A system given by $H(z) = 4 + z^{-1}$ is excited by unit variance white Gaussian noise $w(n)$ to give the signal $s(n)$.

```
clear variables;
```

1) Determine the autocorrelation function, $r_s(l)$

The output signal of the system is given as:

$$s(n) = 4w(n) + w(n - 1)$$

The autocorrelation of this signal is:

$$\begin{aligned} r_s(\ell) &= E[s(n)s(n - \ell)] \\ &= E[(4w(n) + w(n - 1))(4w(n - \ell) + w(n - \ell - 1))] \\ &= E[4w(n)4w(n - \ell)] + E[4w(n)w(n - \ell - 1)] + E[w(n - 1)4w(n - \ell)] + E[w(n - 1)w(n - \ell - 1)] \\ &= 16E[w(n)w(n - \ell)] + 4E[w(n)w(n - \ell - 1)] + 4E[w(n - 1)w(n - \ell)] + E[w(n - 1)w(n - \ell - 1)] \end{aligned}$$

```
syms n l w(n)
expand((4*w(n) + w(n-1)) * (4*w(n-1) + w(n-1-1)))
```

$$\begin{aligned}
\text{ans} &= 16w(n-l)w(n) + 4w(n-l-1)w(n) + 4w(n-l)w(n-1) + w(n-l-1)w(n-1) \\
&= 16r_w(\ell) + 4r_w(\ell-1) + 4r_w(\ell+1) + r_w(\ell)
\end{aligned}$$

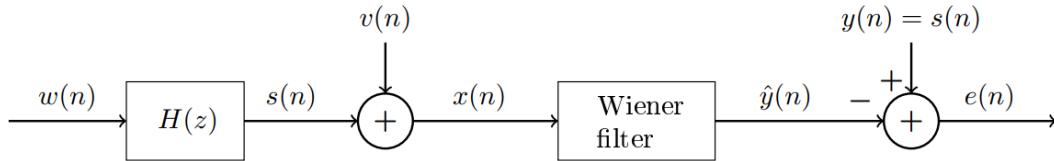
Since the autocorrelation of unit variance white noise is $r_{xx}(\ell) = \sigma_x^2 \delta(\ell) = \delta(\ell)$:

$$r_s(\ell) = 17\delta(\ell) + 4\delta(\ell-1) + 4\delta(\ell+1)$$

```
r_ss = [17, 4, 0];
```

2) Solve the Wiener-Hopf Equation

As shown in the block diagram below, the signal is corrupted by additive white Gaussian noise with $\sigma_v^2 = 3$ giving $x(n) = s(n) + v(n)$. It is desired to recover the signal with a 3-tap Wiener filter.



- Determine the 3×3 autocorrelation matrix R_x and the 3×1 crosscorrelation vector $\mathbf{g} = E[\mathbf{x}(n)\mathbf{y}(n)]$ and use these to solve the Wiener-Hopf equation.

We know that $x(n) = s(n) + v(n)$ and $y(n) = s(n) = 4w(n) + w(n-1)$

As the signals $s(n)$ and $v(n)$ are uncorrelated the autocorrelation of $x(n)$ is just the sum of the individual autocorrelations

Assuming $w(n)$ and $v(n)$ are uncorrelated, the autocorrelation for $x(n)$:

$$\begin{aligned}
r_x(\ell) &= E[x(n)x(n-\ell)] \\
&= E[(s(n) + v(n))(s(n-\ell) + v(n-\ell))] \\
&= E[s(n)s(n-\ell) + s(n)v(n-\ell) + v(n)s(n-\ell) + v(n)v(n-\ell)] \\
&= E[s(n)s(n-\ell)] + E[s(n)v(n-\ell)] + E[v(n)s(n-\ell)] + E[v(n)v(n-\ell)] \\
&= r_s(\ell) + 2r_{sv}(\ell) + r_v(\ell) \\
&= r_s(\ell) + r_v(\ell) \quad (\text{because } w(n) \text{ and } v(n) \text{ are uncorrelated so } r_{sv}(\ell) = 0) \\
&= 17\delta(\ell) + 4\delta(\ell-1) + 4\delta(\ell+1) + 3\delta(\ell) \quad -- r_v(\ell) = 3\delta(\ell) \text{ since } \sigma_v^2 = 3 \\
&= 20\delta(\ell) + 4\delta(\ell-1) + 4\delta(\ell+1)
\end{aligned}$$

```
r_xx = [20, 4, 0];
R_xx = toeplitz(r_xx);
```

We know that $x(n) = s(n) + v(n)$ and $y(n) = s(n)$.

Compute the cross-correlation vector:

$$\begin{aligned} r_{xy}(\ell) &= E[x(n)y(n - \ell)] \\ &= E[(s(n) + v(n))(s(n - \ell))] \\ &= E[s(n)s(n - \ell) + v(n)s(n - \ell)] \\ &= r_s(\ell) + r_{vs}(\ell) \\ &= r_s(\ell) + 0 \quad \text{since } s(n) \text{ and } v(n) \text{ are uncorrelated.} \end{aligned}$$

```
g = r_ss';
```

The third order Wiener filter for estimating the signal $y(n)$ is given by:

$$\hat{y}(n) = h_1x(n) + h_2x(n - 1) + h_3x(n - 2)$$

The optimum Wiener filter to estimate a random process is given by Eq. 14.109:

$$\mathbf{h}_o = \mathbf{R}_x^{-1}\mathbf{g}, \quad (14.109)$$

where \mathbf{R}_x is the correlation matrix of a random vector x and \mathbf{g} is the cross-correlation vector between x and y

The Wiener-Hopf solution is:

```
h_opt = R_xx\g
```

```
h_opt = 3x1
0.8435
0.0326
-0.0065
```

3) Discuss whether 3 taps is an optimum choice for this problem?

We can check whether the minimum mean square error can be lowered by using a longer Wiener filter.

Typically, increasing the length of the Wiener filter lowers the MSE which means that the filter becomes better at separating signal from noise.

The minimum value of the mean square error $E[e^2(n)]E[(y(n) - \hat{y}(n))^2]$ is given by

$$J_o = r_y[0] - \mathbf{h}_o^T \mathbf{g} = r_y[0] - \sum_{k=0}^{p-1} h_o[k]r_{yx}[k]. \quad (14.115)$$

Let us compute it for a 3-tap Wiener filter:

```
mse_tap3 = r_ss(1) - h_opt'*g
```

```
mse_tap3 = 2.5304
```

Let us compute it for a 4-tap Wiener filter:

```
R_xx4 = toeplitz([r_xx, 0]);
g4 = [g', 0]';
h_opt4 = R_xx4\g4;
mse_tap4 = r_ss(1) - h_opt4'*g4
```

```
mse_tap4 = 2.5304
```

Let us compute it for a 5-tap Wiener filter:

```
R_xx5 = toeplitz([r_xx, 0, 0]);
g5 = [g', 0, 0]';
h_opt5 = R_xx5\g5;
mse_tap5 = r_ss(1) - h_opt5'*g5
```

```
mse_tap5 = 2.5304
```

We observe that the minimum MSE can be lowered further by increasing the filter length. However, the decrease in minimum MSE is relatively small. If we want a faster filter, we would stick with 3-tap or even a 2-tap.

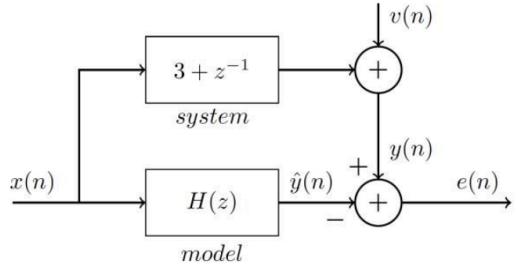
Let us compute it for a 2-tap Wiener filter:

```
R_xx2 = toeplitz(r_xx(1:2));
g2 = g(1:2);
h_opt2 = R_xx2\g2;
mse_tap2 = r_ss(1) - h_opt2'*g2
```

```
mse_tap2 = 2.5313
```

Exam 2018 Problem 1: Model a system using a Wiener filter

Consider the following setup, where we aim to model the system $3+z^{-1}$ using a Wiener filter. The input signal $x(n)$ is unit variance white noise. The output of the system is corrupted by additive noise $v(n)$, which is also white with $\sigma_v^2 = 0.2$. It can be assumed that $x(n)$ and $v(n)$ are uncorrelated.



$x(n) \sim WN(0, 1)$, $v(n) \sim WN(0, 0.2)$ and $x(n)$ and $v(n)$ are uncorrelated

```
clear variables;
```

1) Calculate the autocorrelation matrix and cross-correlation vector

- Calculate the 2×2 autocorrelation matrix \mathbf{R}_x and the 2×1 cross-correlation vector \mathbf{g} .

Since $x(n) \sim WN(0, 1)$, its autocorrelation function is:

$$r_{xx}(\ell) = \sigma_x^2 \delta(\ell) = \delta(\ell)$$

The autocorrelation matrix \mathbf{R}_x is therefore

```
M = 2;
ell = 0:M-1;
r_xx = [1, 0];
R_xx = toeplitz(r_xx)
```

$$\begin{matrix} R_{xx} &= 2 \times 2 \\ &\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \end{matrix}$$

From the drawing, we know that the output of the system is:

$$y(n) = 3x(n) + x(n-1) + v(n)$$

Now, we can compute the cross-correlation vector g :

$$\begin{aligned} r_{xy}(\ell) &= E[x(n)y(n-\ell)] \\ &= E[x(n)(3x(n-\ell) + x(n-\ell-1) + v(n-\ell))] \\ &= E[3x(n)x(n-\ell) + x(n)x(n-\ell-1) + x(n)v(n-\ell)] \\ &= 3E[x(n)x(n-\ell)] + E[x(n)x(n-\ell-1)] + E[x(n)v(n-\ell)] \\ &= 3r_{xx}(\ell) + r_{xx}(\ell-1) + E[x(n)v(n-\ell)] \end{aligned}$$

If we assume that $x(n)$ and $v(n)$ are uncorrelated then:

$$= 3r_{xx}(\ell) + r_{xx}(\ell-1)$$

Since $r_{xx}(\ell) = \delta(\ell)$ then:

$$r_{xy}(\ell) = 3\delta(\ell) + \delta(\ell - 1)$$

Thus, the cross-correlation vector g is:

$$g = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

2) Calculate the expected energy

2. Calculate $E[y^2(n)]$.

From the drawing, we know that the output of the system is:

$$y(n) = 3x(n) + x(n - 1) + v(n)$$

Let us first expand $y^2(n)$:

```
syms n x(n) v(n)
expand((3*x(n) + x(n-1) + v(n))^2)
```

$$\text{ans} = x(n-1)^2 + 2x(n-1)v(n) + 6x(n-1)x(n) + v(n)^2 + 6v(n)x(n) + 9x(n)^2$$

Now, compute the expectation of each term:

$$E[x^2(n-1)] = E[x(n)x(n)] = r_{xx}(0) = 1 \cdot \delta(0) = 1 \cdot 1 = 1$$

$$E[2x(n-1)v(n)] = 2E[x(n-1)v(n)] = 0 \text{ (because } x(n) \text{ and } v(n) \text{ are uncorrelated)}$$

$$E[6x(n-1)x(n)] = 6r_{xx}(1) = 6\delta(1) = 6 \cdot 0 = 0$$

$$E[v^2(n)] = r_{vv}(0) = 0.2 \cdot \delta(0) = 0.2 \cdot 1 = 0.2$$

$$E[6v(n)x(n)] = 6r_{xv}(0) = 6 \cdot 0 = 0 \text{ (because } x(n) \text{ and } v(n) \text{ are uncorrelated)}$$

$$E[9x(n)x(n)] = 9r_{xx}(0) = 9\delta(0) = 9 \cdot 1 = 9$$

The expected energy is:

$$E[y^2(n)] = 1 + 0.2 + 9 = 10.2$$

3) Solve for $H(z)$ and comment on the influence of $v(n)$

3. Solve for $H(z)$ and comment on the result, in particular what is the influence of $v(n)$.

The p -order Wiener filter for estimating the signal $x(n)$ is given by:

$$\hat{y}[n] = \sum_{k=1}^p h_k x[n+1-k] \quad (14.112)$$

The optimum Wiener filter coefficients is given by the Wiener-Hopf equation:

$$\mathbf{h}_o = \mathbf{R}_x^{-1} \mathbf{g}, \quad (14.109)$$

where \mathbf{R}_x is the autocorrelation matrix of the input signal $x(n)$ and \mathbf{g} is the cross-correlation between the input signal $x(n)$ and the desired signal $y(n)$.

```
g = [3, 1]';
R_xx = [[1, 0];
          [0, 1]];
h_opt = R_xx\g
```

```
h_opt = 2x1
      3
      1
```

The optimal Wiener filter is:

$$H(z) = 3 + z^{-1}$$

This is exactly the system that we are trying to model.

Exam 2018 Problem 4: Recover signal using a Wiener filter

A system given by $H(z) = 4 + z^{-1}$ is excited by unit variance white Gaussian noise $w(n)$ to give the signal $s(n)$.

```
clear variables;
```

[✓] 1) Determine the autocorrelation function, $r_s(l)$

The output signal of the system is given as:

$$s(n) = 4w(n) + w(n - 1)$$

The autocorrelation of this signal is:

$$\begin{aligned} r_s(\ell) &= E[s(n)s(n - \ell)] \\ &= E[(4w(n) + w(n - 1))(4w(n - \ell) + w(n - \ell - 1))] \\ &= E[4w(n)4w(n - \ell)] + E[4w(n)w(n - \ell - 1)] + E[w(n - 1)4w(n - \ell)] + E[w(n - 1)w(n - \ell - 1)] \\ &= 16E[w(n)w(n - \ell)] + 4E[w(n)w(n - \ell - 1)] + 4E[w(n - 1)w(n - \ell)] + E[w(n - 1)w(n - \ell - 1)] \end{aligned}$$

```

syms n l w(n)
expand((4*w(n) + w(n-1)) * (4*w(n-1) + w(n-1-1)))

```

$$\text{ans} = 16w(n-l)w(n) + 4w(n-l-1)w(n) + 4w(n-l)w(n-1) + w(n-l-1)w(n-1)$$

$$= 16r_w(\ell) + 4r_w(\ell-1) + 4r_w(\ell+1) + r_w(\ell)$$

Since $r_w(\ell-1) = r_w(\ell+1)$

$$r_s(\ell) = 17r_w(\ell) + 8r_w(\ell-1)$$

Since the autocorrelation of unit variance white noise is $r_{xx}(\ell) = \sigma_x^2\delta(\ell) = \delta(\ell)$:

$$r_s(\ell) = 17\delta(\ell) + 8\delta(\ell-1)$$

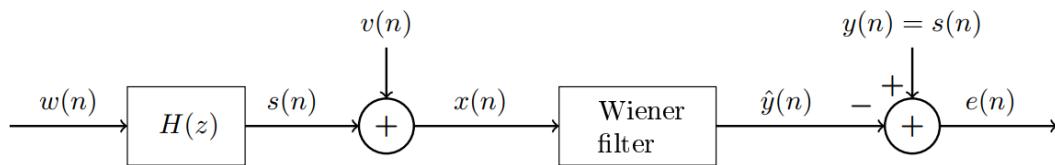
```

r_ss = [17, 8, 0];

```

[✓] 2) Solve the Wiener-Hopf Equation

As shown in the block diagram below, the signal is corrupted by additive white Gaussian noise with $\sigma_v^2 = 3$ giving $x(n) = s(n) + v(n)$. It is desired to recover the signal with a 3-tap Wiener filter.



- Determine the 3×3 autocorrelation matrix R_x and the 3×1 crosscorrelation vector $\mathbf{g} = E[\mathbf{x}(n)\mathbf{y}(n)]$ and use these to solve the Wiener-Hopf equation.

We know that $x(n) = s(n) + v(n)$ and $y(n) = s(n) = 4w(n) + w(n-1)$

Assuming $w(n)$ and $v(n)$ are uncorrelated, the autocorrelation for $x(n)$:

$$\begin{aligned}
r_x(\ell) &= E[x(n)x(n-\ell)] \\
&= E[(s(n) + v(n))(s(n-\ell) + v(n-\ell))] \\
&= E[s(n)s(n-\ell) + s(n)v(n-\ell) + v(n)s(n-\ell) + v(n)v(n-\ell)] \\
&= E[s(n)s(n-\ell)] + E[s(n)v(n-\ell)] + E[v(n)s(n-\ell)] + E[v(n)v(n-\ell)] \\
&= r_s(\ell) + 2r_{sv}(\ell) + r_v(\ell) \\
&= r_s(\ell) + r_v(\ell) \quad (\text{because } w(n) \text{ and } v(n) \text{ are uncorrelated so } r_{sv}(\ell) = 0)
\end{aligned}$$

$$= 17\delta(\ell) + 8\delta(\ell - 1) + 3\delta(\ell) \quad \text{-- } r_v(\ell) = 3\delta(\ell) \text{ since } \sigma_v^2 = 3$$

$$= 20\delta(\ell) + 8\delta(\ell - 1)$$

We know that $x(n) = s(n) + v(n)$ and $y(n) = s(n) = 4w(n) + w(n - 1)$.

Compute the cross-correlation vector:

$$\begin{aligned} r_{xy}(\ell) &= E[x(n)y(n - \ell)] \\ &= E[(s(n) + v(n))(4w(n - \ell) + w(n - \ell - 1))] \\ &= 4E[s(n)w(n - \ell)] + E[s(n)w(n - \ell - 1)] + 4E[v(n)w(n - \ell)] + E[v(n)w(n - \ell - 1)] \\ &= 4r_{sw}(\ell) + r_{sw}(\ell - 1) + 4r_{vw}(\ell) + r_{vw}(\ell - 1) \end{aligned}$$

We assume that $w(n)$ and $v(n)$ are uncorrelated so $r_{vw}(\ell) = 0$:

$$= 4r_{sw}(\ell) + r_{sw}(\ell - 1) + 0 + 0$$

According to Eq. 13.100, the cross-correlation $r_{sw}(\ell) = \sigma_w^2 h(\ell) = h(\ell)$ since $\sigma_w^2 = 1$. We know that the impulse response $h(\ell) = [4, 1]$:

$$= 4h(\ell) + h(\ell - 1)$$

The third order Wiener filter for estimating the signal $y(n)$ is given by:

$$\hat{y}(n) = h_1 x(n) + h_2 x(n - 1) + h_3 x(n - 2)$$

The optimum Wiener filter to estimate a random process is given by Eq. 14.109:

$$\mathbf{h}_o = \mathbf{R}_x^{-1} \mathbf{g}, \quad (14.109)$$

where \mathbf{R}_x is the correlation matrix of a random vector x and \mathbf{g} is the cross-correlation vector between x and y

```
r_xx = [20, 8, 0];
R_xx = toeplitz(r_xx)
```

```
R_xx = 3x3
20     8      0
 8     20      8
 0     8     20
```

```
g = [4, 1, 0]'
```

```
g = 3x1
 4
 1
 0
```

```
h_opt = R_xx\g
```

```

h_opt = 3x1
 0.2176
 -0.0441
 0.0176

```

The optimal Wiener filter is given by:

$$H(z) = 0.2471 - 0.1176z^{-1} + 0.0471z^{-2}$$

[✓] 3) Discuss whether 3 taps is an optimum choice for this problem?

We can check whether the minimum mean square error can be lowered by using a longer Wiener filter.

Typically, increasing the length of the Wiener filter lowers the MSE which means that the filter becomes better

The minimum value of the mean square error $E[e^2(n)]E[(y(n) - \hat{y}(n))^2]$ is given by

$$J_0 = r_y[0] - h_o^T g = r_y[0] - \sum_{k=0}^{p-1} h_o[k] r_{yx}[k]. \quad (14.115)$$

Let us compute it for a 3-tap Wiener filter:

```

mse_tap3 = r_ss(1) - h_opt'*g
mse_tap3 = 16.1735

```

Let us compute it for a 4-tap Wiener filter:

```

R_xx4 = toeplitz([r_xx, 0]);
g4 = [g', 0]';
h_opt4 = R_xx4\g4;
mse_tap4 = r_ss(1) - h_opt4'*g4

```

`mse_tap4 = 16.1723`

Let us compute it for a 5-tap Wiener filter:

```

R_xx5 = toeplitz([r_xx, 0, 0]);
g5 = [g', 0, 0]';
h_opt5 = R_xx5\g5;
mse_tap5 = r_ss(1) - h_opt5'*g5

```

`mse_tap5 = 16.1720`

We observe that the minimum MSE can be lowered further by increasing the filter length. However, the decrease in minimum MSE is relatively small. If we want a faster filter, we would stick with 3-tap or even a 2-tap.

Let us compute it for a 2-tap Wiener filter:

```

R_xx2 = toeplitz(r_xx(1:2));
g2 = g(1:2);
h_opt2 = R_xx2\g2;

```

```
mse_tap2 = r_ss(1) - h_opt2'*g2
```

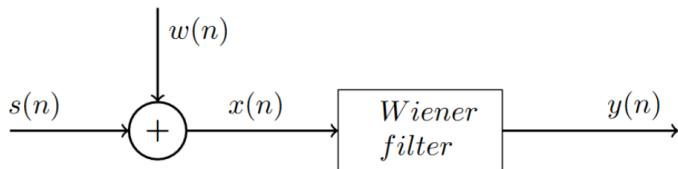
```
mse_tap2 = 16.1786
```

ADSI Problem 6.3: Recover AR(1) signal using Wiener filter

Consider a signal $x(n) = s(n) + w(n)$ where $s(n)$ is an AR(1) process that satisfies the difference equation

$$s(n) = 0.8s(n-1) + v(n)$$

where $\{v(n)\}$ is a white noise sequence with variance $\sigma_v^2 = 0.49$ and $\{w(n)\}$ is a white noise sequence with variance $\sigma_w^2 = 1$. The processes $\{v(n)\}$ and $\{w(n)\}$ are uncorrelated.



```
clear variables;
```

1) Determine the autocorrelation sequence for a signal with two noise processes

Determine the autocorrelation sequences $\{r_s(l)\}$ and $\{r_x(l)\}$.

In Problem 6.2, we found that the autocorrelation function of an AR(1) is:

$$r_{yy}^{\text{AR}(1)}(\ell) = (-a_1)^{|\ell|} \frac{\sigma_x^2}{1 - a_1^2}$$

For the AR(1) process, we are given $a_1 = -0.8$ and $\sigma_v^2 = 0.49$. So the autocorrelation sequence for $s(n)$ is:

$$r_{ss}(\ell) = 0.8^{|\ell|} \frac{0.49}{1 - (-0.8)^2} = 0.8^{|\ell|} \frac{0.49}{0.36}$$

Next, we need to find the autocorrelation sequence for $w(n)$. The autocorrelation of white Gaussian noise is $\sigma_w^2 \delta(\ell)$ so since white noise has unit variance i.e. $\sigma_w^2 = 1$ we have:

$$r_{ww}(\ell) = \delta(\ell)$$

As the two white noise processes are uncorrelated, the signal $s(n)$ and $w(n)$ are also uncorrelated. The autocorrelation of the noisy signal is therefore just the sum of the individual autocorrelations:

$$r_{xx}(\ell) = r_{ss}(\ell) + r_{ww}(\ell)$$

Therefore, the ACRS of $\{x(n)\}$ process is:

$$r_{xx}(\ell) = 0.8^{|\ell|} \frac{0.49}{0.36} + \delta(\ell)$$

2) Design a Wiener filter of length M=2 to estimate an AR(1) process

Design a Wiener filter of length $M = 2$ to estimate $\{s(n)\}$.

An M 'order Wiener filter for estimating the original signal $s(n)$ is given by Eq. 14.112:

$$\hat{y}[n] = \sum_{k=1}^p h_k x[n+1-k] \quad (14.112)$$

Note: In this problem, we must use $s(n)$ instead of $y(n)$ which is used in the book.

The second order Wiener filter for estimating the original signal $s(n)$ is given by:

$$\hat{s}(n) = h_1 x(n) + h_2 x(n-1)$$

The optimum Wiener filter to estimate a random process is given by Eq. 14.109:

$$\mathbf{h}_o = \mathbf{R}_x^{-1} \mathbf{g}, \quad (14.109)$$

where \mathbf{R}_x is the correlation matrix of a random vector x and \mathbf{g} is the cross-correlation vector between x and y (the signal that we want to recover which in this problem is $s[n]$).

Basically, to design a p th order Wiener filter, we have to solve following equation with respect to \mathbf{h} :

$$\begin{bmatrix} r_x[0] & r_x[1] & \dots & r_x[p-1] \\ r_x[1] & r_x[0] & \dots & r_x[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_x[p-1] & r_x[p-2] & \dots & r_x[0] \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_p \end{bmatrix} = \begin{bmatrix} r_{yx}[0] \\ r_{yx}[1] \\ \vdots \\ r_{yx}[p-1] \end{bmatrix}, \quad (14.113)$$

Eq. 14.113 is the normal equation for the Wiener filter.

Computing the autocorrelation matrix is straightforward since we have computed the autocorrelation $r_{xx}(\ell)$ in part 1).

```
M = 2;
ell = 0:M-1;

r_ss = 0.8.^abs(ell) * (0.49/0.36);
r_ww = (ell == 0); % Simulate the delta function
r_xx = r_ss + r_ww;
```

```
R_xx = toeplitz(r_xx)
```

```
R_xx = 2x2
 2.3611  1.0889
 1.0889  2.3611
```

We have to come up with an expression for the cross-correlation $r_{sx}(\ell)$:

$$r_{sx}(\ell) = E[s(n)x(n - \ell)]$$

$$r_{sx}(\ell) = E[s(n)(s(n - \ell) + w(n - \ell))]$$

$$r_{sx}(\ell) = E[s(n)s(n - \ell) + s(n)w(n - \ell)]$$

$$r_{sx}(\ell) = E[s(n)s(n - \ell)] + E[s(n)w(n - \ell)]$$

$$r_{sx}(\ell) = r_{ss}(\ell) + r_{sw}(\ell)$$

We already have an expression for $r_{ss}(\ell)$. Since the processes $s(n)$ and $w(n)$ are uncorrelated, we know that the cross-correlation between the two processes is $r_{sw}(\ell) = 0$ so we are left with:

$$r_{sx}(\ell) = r_{ss}(\ell) + 0$$

From 1), we know the that:

$$r_{ss}(\ell) = 0.8^{|\ell|} \frac{0.49}{0.36}$$

```
g = r_ss';
h_opt = R_xx\g % Same as `inv(R_xx)*g` but better
```

```
h_opt = 2x1
 0.4621
 0.2481
```

The second order Wiener filter for estimating the original signal $s(n)$ is therefore:

$$\hat{s}(n) = 0.4621x(n) + 0.2481x(n - 1)$$

3) Determine the minimum mean square error for M=2

The minimum square error for an optimum p th Wiener (FIR) filter is given by Eq. 14.115:

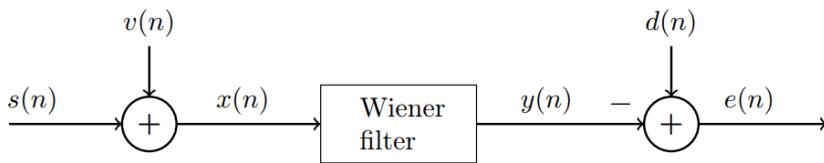
$$J_o = r_y[0] - \mathbf{h}_o^T \mathbf{g} = r_y[0] - \sum_{k=0}^{p-1} h_o[k] r_{yx}[k]. \quad (14.115)$$

```
mse = r_ss(1) - h_opt'*g
```

```
mse = 0.4621
```

Exam 2017 Problem 4: Recover signal using a Wiener filter

Consider the Wiener filtering problem shown below. A signal $s(n)$ is corrupted by additive, uncorrelated noise $v(n)$ giving the signal $x(n) = s(n) + v(n)$. It is desired to use a 2-tap Wiener filter to recover the signal i.e. $d(n) = s(n)$.



The noise $v(n)$ is generated in an MA(1) process as

$$v(n) = w(n) + w(n - 1)$$

where $w(n)$ is a zero-mean gaussian white noise sequence with $\sigma_w^2 = 1$. The autocorrelation function of the signal is

$$r_s(l) = 4 \cdot 0.25^{|l|}.$$

```

clear variables;
M = 2;
ell = 0:M-1;
delta = @(l) ell == 1;
r_ss = 4*(0.25).^(abs(ell));
  
```

1) Compute the autocorrelation function of the noise

1. Show that the autocorrelation function of the noise $v(n)$ is given by

$$r_v(l) = \delta(l - 1) + 2\delta(l) + \delta(l + 1).$$

In ADSI Problem 4.9, we found that the autocorrelation for an MA(1) process where the input signal is a white noise with unit variance can be described as:

$$r_v(\ell) = (b_0^2 + b_1^2)\delta(\ell) + b_0b_1(\delta(\ell - 1) + \delta(\ell + 1))$$

In this problem $b_0 = b_1 = 1$:

$$r_v(\ell) = \delta(\ell - 1) + 2\delta(\ell) + \delta(\ell + 1)$$

```
r_vv = delta(-1) + 2*delta(0) + delta(1)
```

```
r_vv = 1x2
      2      1
```

2) Compute the optimum filter coefficients

2. Calculate the crosscorrelation vector \mathbf{g} , the autocorrelation matrix \mathbf{R}_x and the optimum filter coefficients.

The p -order Wiener filter for estimating the signal $s(n)$ is given by:

$$\hat{y}[n] = \sum_{k=1}^p h_k x[n+1-k] \quad (14.112)$$

The optimum Wiener filter coefficients is given by the Wiener-Hopf equation:

$$\mathbf{h}_o = \mathbf{R}_x^{-1} \mathbf{g}, \quad (14.109)$$

where \mathbf{R}_x is the autocorrelation matrix of the corrupted signal $s(n)$ and \mathbf{g} is the cross-correlation between the desired signal $s(n)$ and the corrupted signal $x(n)$.

Designing a Wiener filter to recover a corrupted signal involves 3 steps:

1. Compute the autocorrelation sequence $r_x(\ell)$ and matrix R_x
2. Compute the cross-correlation $r_{sx}(\ell)$
3. Solve the Wiener-Hopf equation to find the optimum Wiener filter coefficients

Step 1: Compute the autocorrelation $r_x(\ell)$:

Since the signal $s(n)$ and the noise $v(n)$ are uncorrelated the autocorrelation function of $s(n)$ is just the sum of the individual autocorrelation functions:

$$\begin{aligned} r_x(\ell) &= E[x(n)x(n-\ell)] \\ &= E[(s(n)+v(n))(s(n-\ell)+v(n-\ell))] \\ &= E[s(n)s(n-\ell) + s(n)v(n-\ell) + v(n)s(n-\ell) + v(n)v(n-\ell)] \\ &= r_s(\ell) + r_{sv}(\ell) + r_{vs}(\ell) + r_v(\ell) \end{aligned}$$

Since $s(n)$ and $v(n)$ are uncorrelated $r_{sv}(\ell) = 0$. We computed $r_v(\ell) = \delta(\ell-1) + 2\delta(\ell) + \delta(\ell+1)$:

$$r_x(\ell) = r_s(\ell) + r_v(\ell)$$

```
r_xx = r_ss + r_vv;
R_xx = toeplitz(r_xx)
```

```
R_xx = 2x2
 6   2
 2   6
```

Step 2: Compute the cross-correlation $r_{sx}(\ell)$:

Since the desired signal is $s(n)$, the cross-correlation between $s(n)$ and $s(n)$ simplifies to the autocorrelation of the signal $r_s(\ell)$.

$$\begin{aligned} r_{yx}(\ell) &= E[s(n)x(n-\ell)] \\ &= E[s(n)(s(n-\ell) + v(n-\ell))] \\ &= E[s(n)s(n-\ell)] + E[s(n)v(n-\ell)] \\ &= r_s(\ell) + r_{sv}(\ell) \\ &= r_s(\ell) + 0 \quad (\text{since } s(n) \text{ and } v(n) \text{ are uncorrelated } r_{sv}(\ell) = 0) \end{aligned}$$

```
g = r_ss'
```

```
g = 2x1
 4
 1
```

Step 3: Compute the optimum filter coefficients:

```
h_opt = R_xx\g
```

```
h_opt = 2x1
 0.6875
 -0.0625
```

3) Calculate SNR before and after the Wiener filter

The signal to noise ratio of the input signal is given by

$$\text{SNR}_i = \frac{\text{(value of signal at } n = n_0)^2}{\text{power of noise}} = \frac{s^2(n = n_0)}{r_v(0)} = \frac{r_s(0)}{r_v(0)}$$

We know that:

$$r_s(\ell) = 4 \cdot 0.25^{|\ell|} \quad \text{and} \quad r_v(\ell) = \delta(\ell - 1) + 2\delta(\ell) + \delta(\ell + 1)$$

This means:

$$r_s(0) = 4 \quad \text{and} \quad r_v(0) = 2$$

Thus:

$$\text{SNR}_i = \frac{r_s(0)}{r_v(0)} = \frac{4}{2} = 2$$

The output SNR of the Wiener filter is given by:

$$\text{SNR}_o = \frac{h_{\text{opt}}^T R_s h_{\text{opt}}}{h_{\text{opt}}^T R_v h_{\text{opt}}}$$

```
R_ss = toeplitz(r_ss);
R_vv = toeplitz(r_vv);
SNR_o = (h_opt'*R_ss*h_opt) / (h_opt'*R_vv*h_opt)
```

```
SNR_o = 2.0991
```

The filter increases the SNR but only slightly. This is expected and is an excellent way to check your results.

Table of Contents

Matlab Cheatsheet.....	1
How to represent polynomials in Matlab?.....	1
How to extract coefficients from polynomials?.....	1
How to compute the roots of a polynomial?.....	2
How to compute zero-pole representation of a transfer function?.....	2
How to compute partial fraction expansion?.....	3
How to perform polynomial multiplication?.....	4
How to convert a transfer function to its frequency response ?.....	4
How to plot the impulse reponse from a transfer function?.....	5
How to compute the group delay?.....	6
How to plot the zeros and poles?.....	7
How to plot phase response (principle value) and continuos phase function?.....	8
How to plot the magnitude response?.....	9
How to generate white Gaussian noise?.....	10
How to generate random value from uniform distribution?.....	14
How to delay a signal?.....	15
Functions.....	15

Matlab Cheatsheet

How to represent polynomials in Matlab?

Representation of polynomials in MATLAB Since most practical z -transforms are a ratio of polynomials, we start by explaining how MATLAB handles polynomials. In MATLAB polynomials are represented by *row* vectors containing the coefficients of the polynomial in decreasing order. For example, the polynomial

$$B(z) = 1 + 2z^{-1} + 3z^{-3}$$

is entered as `b=[1,2,0,3]`. We stress that even though the coefficient of the z^{-2} term

```
b = [1 2 0 3]
```

```
b = 1x4  
     1      2      0      3
```

How to extract coefficients from polynomials?

Suppose we want to automatically get the coefficients of the following polynomial:

$$H(z) = -8 + 10z^{-1} - 3z^{-2} - \frac{1}{2}z^{-3} + \frac{1}{4}z^{-4}$$

```
syms z;
H = -8 + 10*z^-1 - 3*z^-2 - 1/2*z^-3 + 1/4*z^-4;
H_b = coeffs(expand(H * z^4), 'all')
```

```
H_b =
(-8 10 -3 -1/2 1/4)
```

Representation of polynomials in MATLAB Since most practical z -transforms are a ratio of polynomials, we start by explaining how MATLAB handles polynomials. In MATLAB polynomials are represented by *row* vectors containing the coefficients of the polynomial in decreasing order. For example, the polynomial

$$B(z) = 1 + 2z^{-1} + 3z^{-3}$$

is entered as `b=[1,2,0,3]`. We stress that even though the coefficient of the z^{-2} term

```
B = 1 + 2*z^-1 + 3*z^-3;
B_b = coeffs(expand(B * z^3), 'all')
```

```
B_b = (1 2 0 3)
```

How to compute the roots of a polynomial?

```
b = [1 1.5 2];
z = roots(b)
```

```
z = 2x1 complex
-0.7500 + 1.1990i
-0.7500 - 1.1990i
```

How to compute zero-pole representation of a transfer function?

Suppose we have the transfer function of a FIR filter:

$$H(z) = 1 + 4.5z^{-1} + 2z^{-2}$$

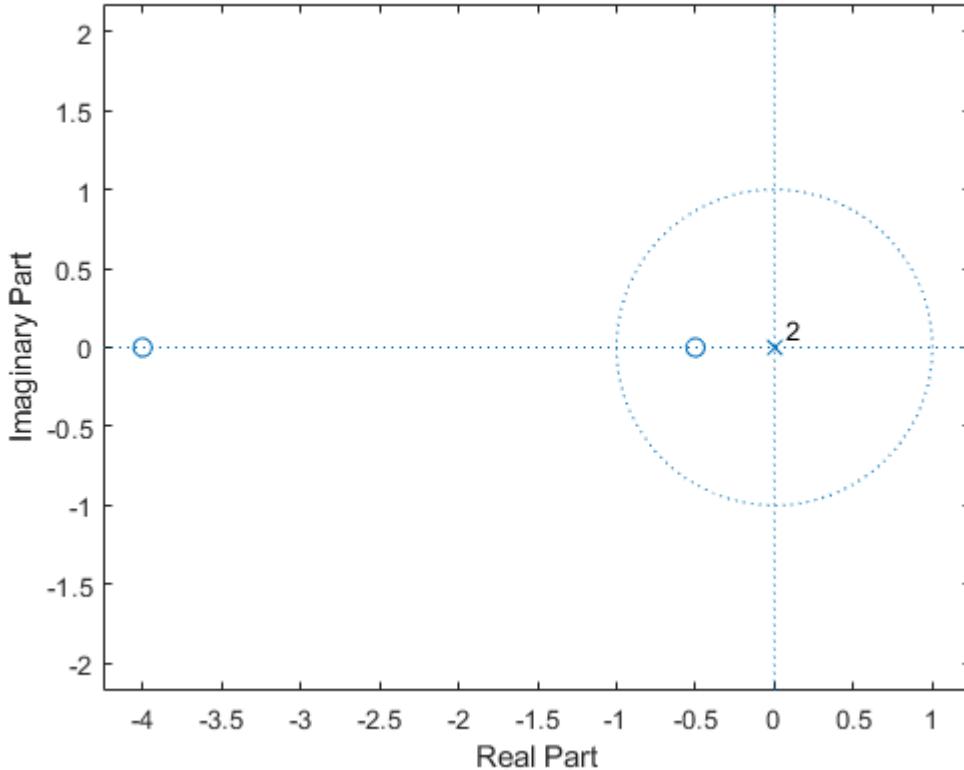
Step 1: find the zeros:

```
zeros = roots([1, 4.5, 2])
```

```
zeros = 2x1
-4.0000
-0.5000
```

We can see that there are two zeros at (-4, 0) and (-0.5, 0). Let us visualise it:

```
zplane([1, 4.5, 2]);
```



Step 2: use the formula $H(z) = b_0 \prod_k (1 - z_k z^{-1})$

$$H(z) = 1(1 + 4z^{-1})(1 + 0.5z^{-1})$$

How to compute partial fraction expansion?

Example 3.10 Partial fraction expansion using residuez

The following expansion:

$$X(z) = \frac{6 - 10z^{-1} + 2z^{-2}}{1 - 3z^{-1} + 2z^{-2}} = 1 + \frac{2}{1 - z^{-1}} + \frac{3}{1 - 2z^{-1}}, \quad (3.45)$$

is obtained by calling `residuez` with `b=[6,-10,2]` and `a=[1,-3,2]`. The reverse operation can be done using the same function as: `[b,a]=residuez(A,p,C)`.

```
b = [ 6 -10  2];
a = [ 1   -3  2];
```

```
% Partial fraction expansion using residuez
[A,p,C] = residuez(b, a)
```

```
A = 2x1
 3
 2
p = 2x1
 2
 1
C = 1
```

```
% Reverse operation
[b, a] = residuez(A, p, C)
```

```
b = 1x3
 6   -10      2
a = 1x3
 1    -3      2
```

How to perform polynomial multiplication?

Polynomial multiplication in MATLAB The convolution theorem (3.52) shows that polynomial multiplication is equivalent to convolution. Therefore, to compute the product

$$\begin{aligned}B(z) &= (1 + 2z^{-2})(1 + 4z^{-1} + 2z^{-2} + 3z^{-3}) \\&= 1 + 4z^{-1} + 4z^{-2} + 11z^{-3} + 4z^{-4} + 6z^{-5},\end{aligned}$$

we use the function

```
>> b=conv([1 0 2],[1 4 2 3])
b =
 1      4      4     11      4      6
```

to find the coefficients of $B(z)$.

How to convert a transfer function $H(z)$ to its frequency response $H(e^{j\omega})$?

Suppose we have the following transfer function:

$$H(z) = \frac{0.05634(1 + z^{-1})(1 - 1.0166z^{-1} + z^{-2})}{(1 - 0.683z^{-1})(1 - 1.4461z^{-1} + 0.7957z^{-2})}$$

We can express the numerator and denominator as polynomial convolutions:

```
b0 = 0.05634;
b1 = [1 1];
b2 = [1 -1.0166 1];
```

```

a1 = [1 -0.683];
a2 = [1 -1.4461 0.7957];

b = b0*conv(b1,b2);
a = conv(a1,a2);

```

We can use the freqz function to get the frequency response as a vector. The second output variable w is the angular frequencies.

```
[H,w] = freqz(b,a);
```

How to plot the impulse reponse from a transfer function?

Suppose we have the following transfer function:

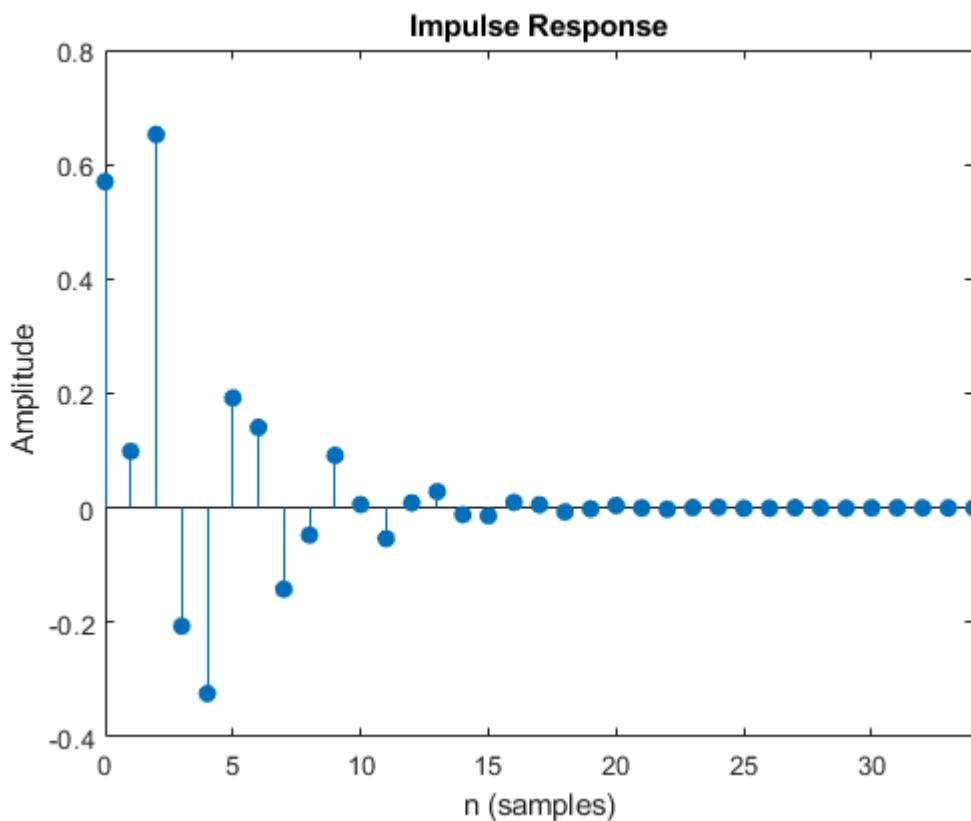
$$H(z) = \frac{0.57 + 0.23z^{-1} + z^{-2}}{1 + 0.23z^{-1} + 0.57z^{-2}}$$

We can use the impz function:

```

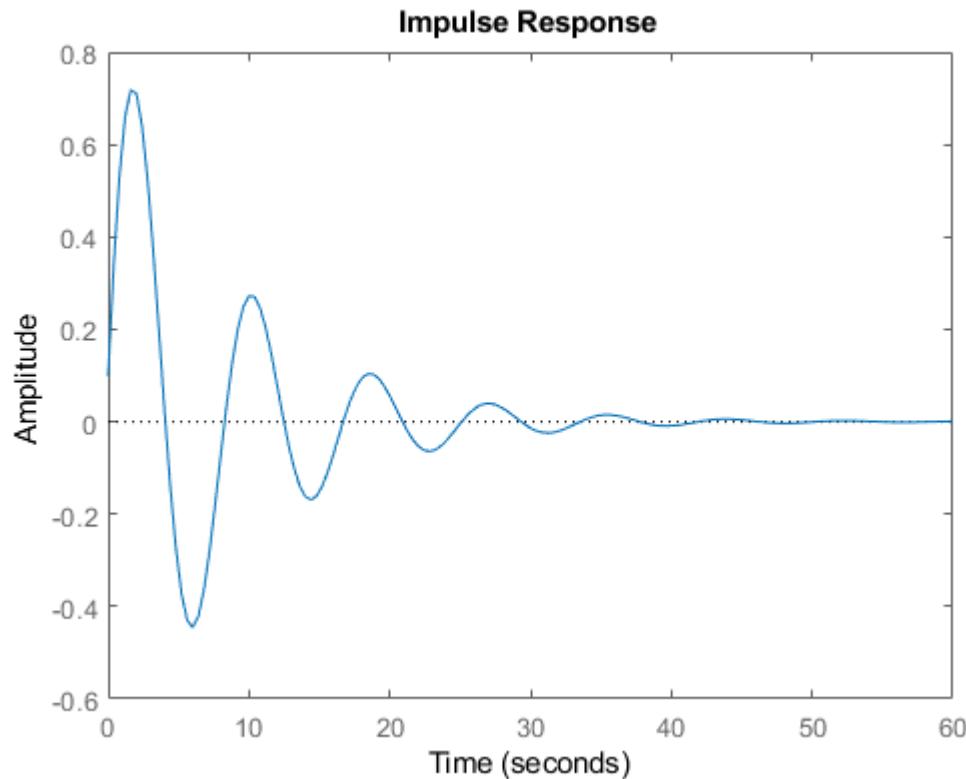
b = [0.57, 0.23, 1];
a = [1, 0.23, 0.57];
impz(b, a);

```



Alternatively,

```
b = [0.57, 0.23, 1];
a = [1, 0.23, 0.57];
h = tf(b, a);
impulseplot(h);
```



How to compute the group delay?

Suppose we have the following transfer function:

$$H(z) = \frac{1 + 1.655z^{-1} + 1.655z^{-2} + z^{-3}}{1 - 1.57z^{-1} + 1.264z^{-2} - 0.4z^{-3}},$$

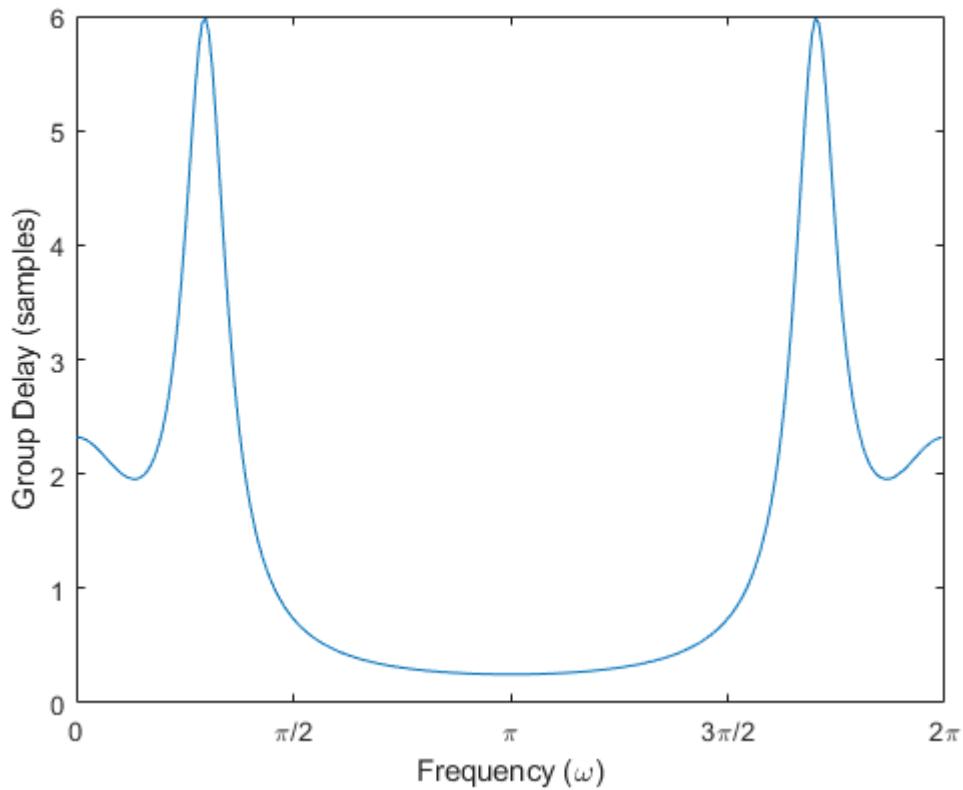
```
b = [1, 1.655, 1.655, 1];
a = [1, -1.57, 1.264, -0.4];
[gd, w] = grpdelay(b, a, 255, 'whole');

plot(w, gd);
set(gca, 'XTick', 0:pi/2:2*pi)
set(gca, 'XTickLabel', {'0', '\pi/2', '\pi', '3\pi/2', '2\pi'})
xlabel('Frequency (\omega)')
```

```

ylabel('Group Delay (samples)')
xlim([0, 2*pi]);

```



How to plot the zeros and poles?

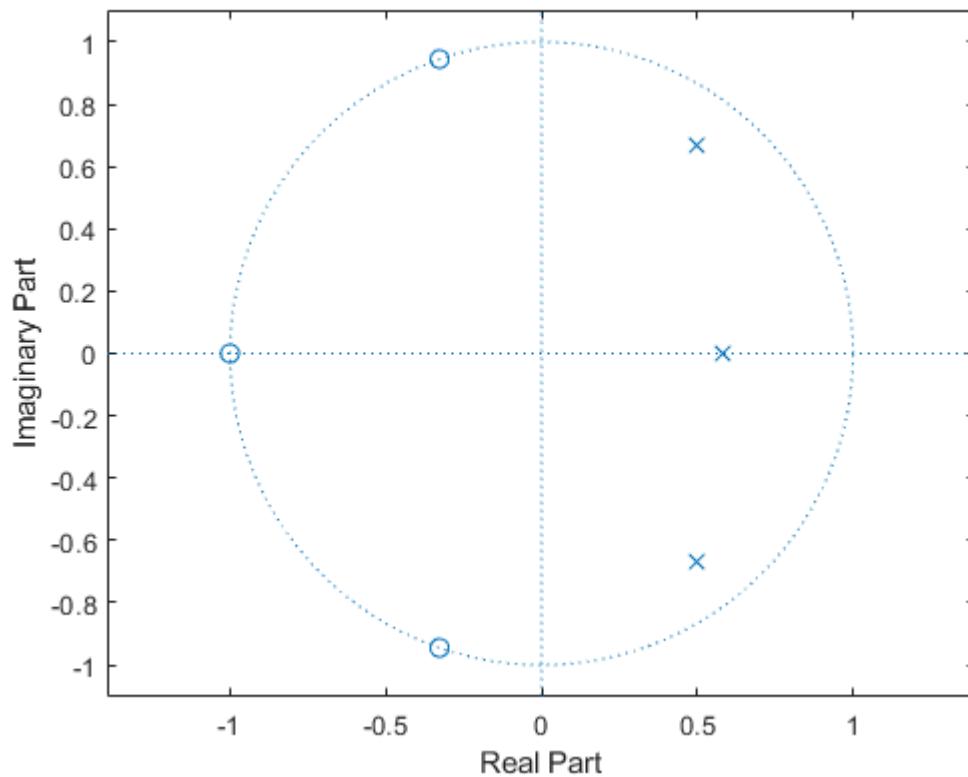
Suppose we have the following transfer function:

$$H(z) = \frac{1 + 1.655z^{-1} + 1.655z^{-2} + z^{-3}}{1 - 1.57z^{-1} + 1.264z^{-2} - 0.4z^{-3}},$$

```

b = [1, 1.655, 1.655, 1];
a = [1, -1.57, 1.264, -0.4];
zplane(b, a);

```



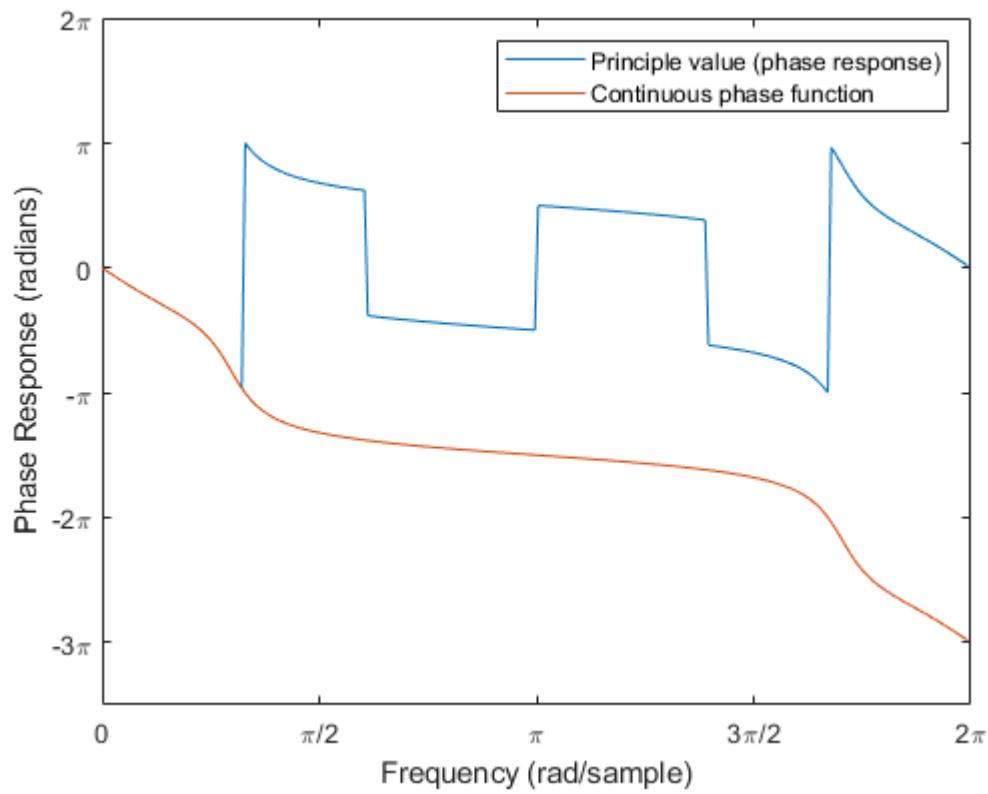
How to plot phase response (principle value) and continuos phase function?

Suppose we have the following transfer function:

$$H(z) = \frac{1 + 1.655z^{-1} + 1.655z^{-2} + z^{-3}}{1 - 1.57z^{-1} + 1.264z^{-2} - 0.4z^{-3}},$$

```
b = [1, 1.655, 1.655, 1];
a = [1, -1.57, 1.264, -0.4];

[gd, w] = grpdelay(b, a, 255, 'whole');
[H, w] = freqz(b, a, 255, 'whole');
plot(w, angle(H), w, contphase(gd, w));
legend('Principle value (phase response)', 'Continuous phase function')
set(gca, 'XTick', 0:pi/2:2*pi)
set(gca, 'XTickLabel', {'0', '\pi/2', '\pi', '3\pi/2', '2\pi'})
set(gca, 'YTick', -3*pi:pi:3*pi)
set(gca, 'YTickLabel', {'-3\pi', '-2\pi', '-\pi', '0', '\pi', '2\pi', '3\pi'})
xlabel('Frequency (rad/sample)')
ylabel('Phase Response (radians)')
xlim([0, 2*pi]);
ylim([-3.5*pi, 2*pi]);
```

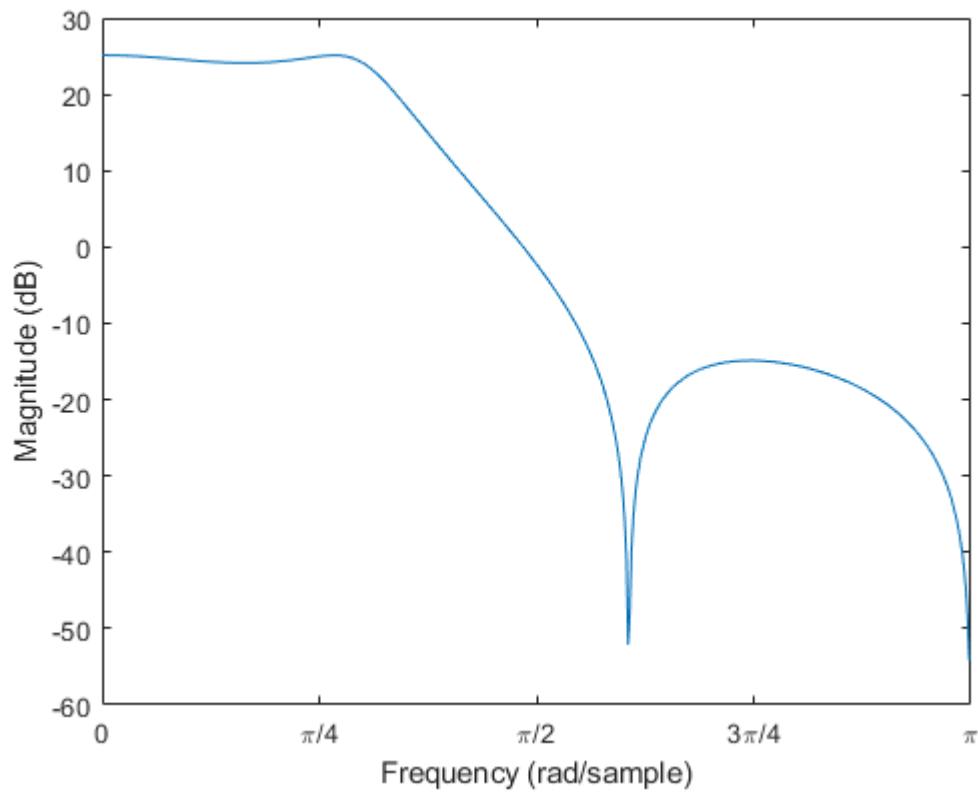


How to plot the magnitude response?

Suppose we have the following transfer function:

$$H(z) = \frac{1 + 1.655z^{-1} + 1.655z^{-2} + z^{-3}}{1 - 1.57z^{-1} + 1.264z^{-2} - 0.4z^{-3}},$$

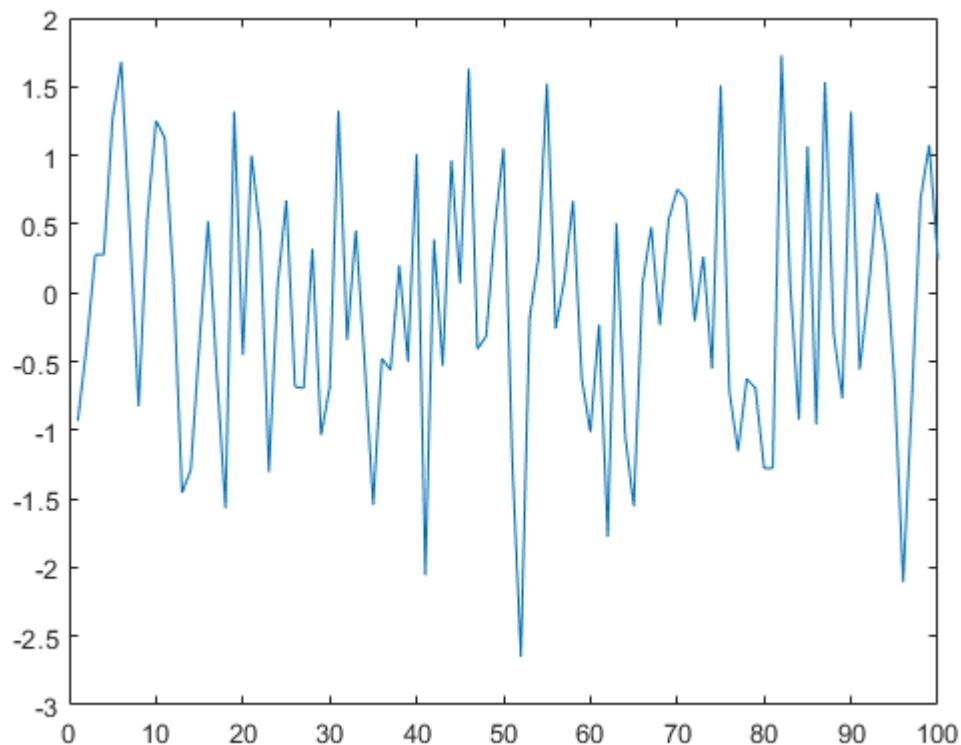
```
b = [1, 1.655, 1.655, 1];
a = [1, -1.57, 1.264, -0.4];
[H, w] = freqz(b, a);
plot(w, pow2db(H.*conj(H)));
set(gca,'XTick', 0:pi/4:pi)
set(gca,'XTickLabel', {'0','\pi/4','\pi/2','3\pi/4','\pi'})
xlabel('Frequency (rad/sample)')
ylabel('Magnitude (dB)')
xlim([0, pi]);
```



How to generate white Gaussian noise?

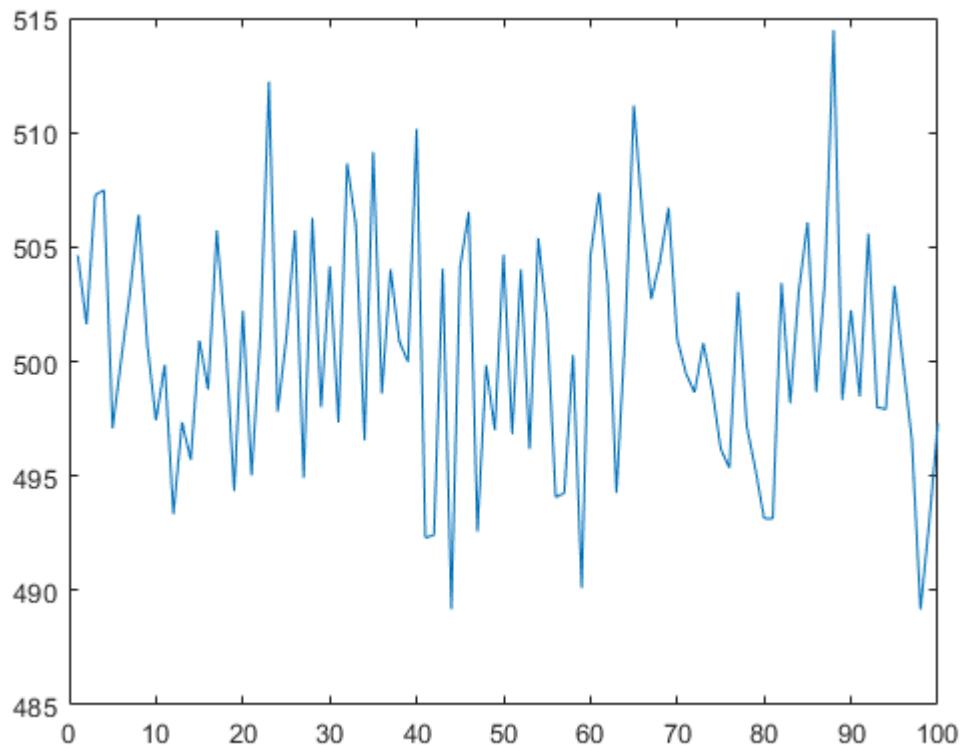
Use the function wgn.

```
N = 100;
n = [1:N];
x = wgn(N,1,0);
plot(n, x)
```



Create a vector of 1000 random values drawn from a normal distribution with a mean of 500 and a standard deviation of 5.

```
std_dev = 5;
mu = 500;
w = std_dev.*randn(N,1) + mu;
plot(n, w)
```



```
% Calculate the sample mean, standard deviation, and variance.
stats = [mean(w) std(w) var(w)]
```

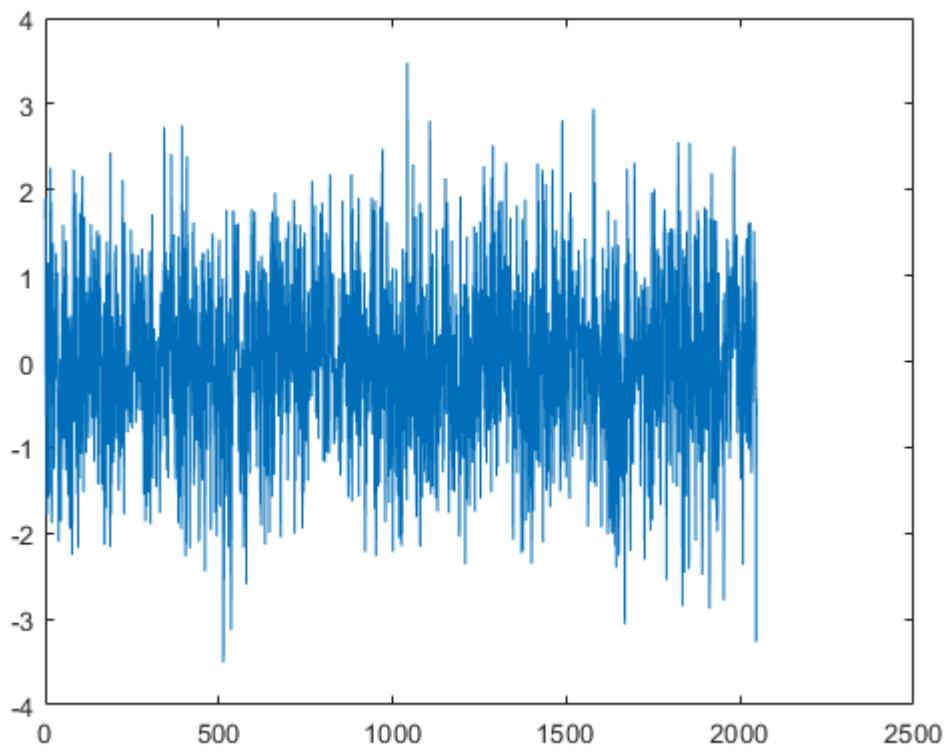
```
stats = 1x3
500.4948    5.1767    26.7987
```

The mean and variance are not 500 and 25 exactly because they are calculated from a sampling of the distribution.

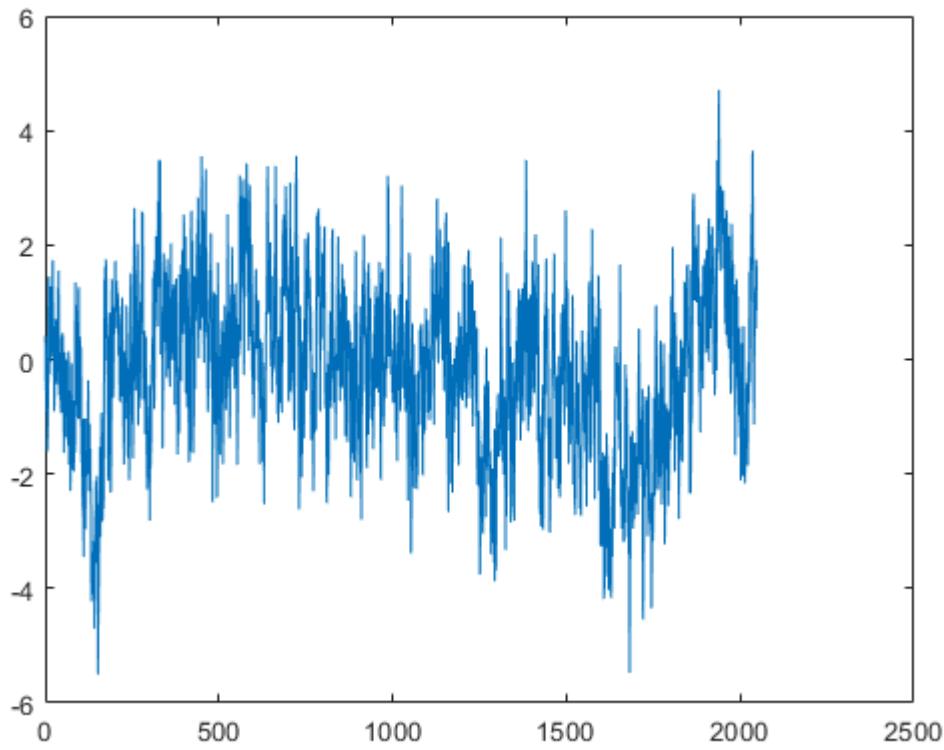
How to generate coloured noise?

```
N = 2048;

% Slight coloured noise
x1 = step(dsp.ColoredNoise('InverseFrequencyPower', 0.1, 'SamplesPerFrame', N));
plot(x1);
```



```
% Very coloured noise (Pink noise)
x1 = step(dsp.ColoredNoise('InverseFrequencyPower', 1, 'SamplesPerFrame', N));
plot(x1);
```



How to generate random value from uniform distribution?

By default, `rand` returns normalized values (between 0 and 1) that are drawn from a uniform distribution. To change the range of the distribution to a new range, (a, b) , multiply each value by the width of the new range, $(b - a)$ and then shift every value by a .

```
a = 50;
b = 100;
r = (b-a).*rand(4, 1) + a
```

```
r = 4x1
52.2526
86.1587
67.3719
83.0308
```

Generate a random value for the uniform distribution $\phi \sim U(0, 2\pi)$:

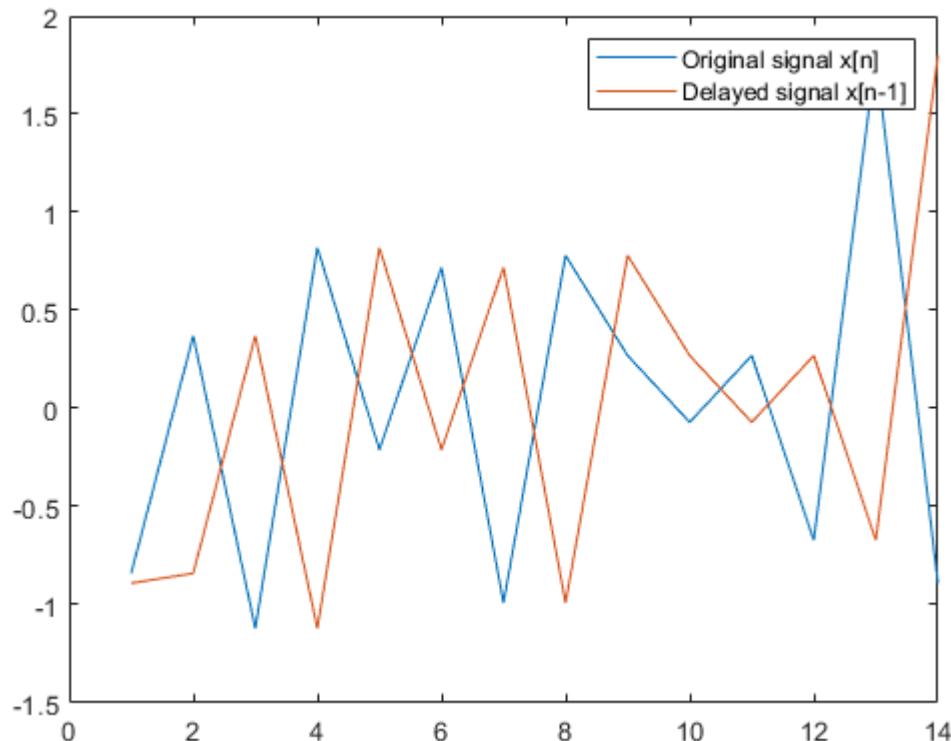
```
phi = 2*pi*rand()
```

```
phi = 2.4119
```

How to delay a signal?

Use the circshift function.

```
delay = 1;
x = [-0.84, 0.37, -1.12, 0.82, -0.21, 0.72, -0.99, 0.78, 0.27, -0.07, 0.27, -0.67, 1.80, -0.89];
N = numel(x);
n = [1:N];
plot(n, x, n, circshift(x, delay))
legend('Original signal x[n]', strcat('Delayed signal x[n-', num2str(delay), ']'))
```



Functions

```
function cph=contphase(grd,om)
% Computation of continuous phase function
% from equidistant values of group delay
N=length(om);
dom=om(2)-om(1);
p(1)=0;
for k=2:N
```

```
p(k)=p(k-1)+dom*(grd(k-1)+grd(k))/2;  
end  
cph=-p;  
end
```