Exam 2016

Table of Contents

Problem 1

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

$\lvert l \rvert$	$r_x(l)$
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:

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

This is useful because we can use it to find the coefficients $\{a_k\}$ of an AR(q) model using the autocorrelation $r_{\mathrm{yy}}(\ell)$ 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:

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

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

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

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

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*:

$$\boldsymbol{R}_y \boldsymbol{a} = -\boldsymbol{r}_y. \qquad (13.143)$$

The input noise variance can be compute as follows:

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

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 = 2×1
   -1.5604
    0.8114
v = 0.9922
```

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

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

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

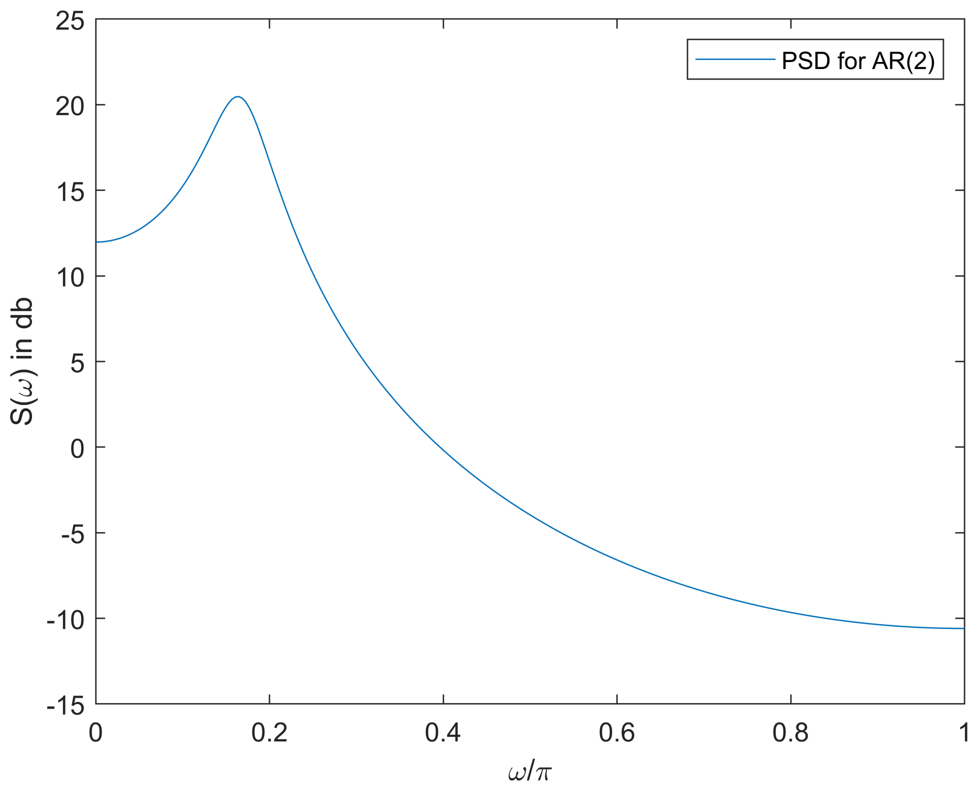
$$S_{yy}(\omega) = \sigma_x^2 \left| \frac{1}{1 + \sum_{k=1}^{p} a_k e^{-j\omega k}} \right|^2$$

To solve this problem, the method is as follows:

1. Find the coefficents $\{a_1, a_2, \cdots, a_p\}$ for the $\mathrm{AR}(p)$ model
2. Compute the transfer function for the $\mathrm{AR}(p)$ 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 variane 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)
```

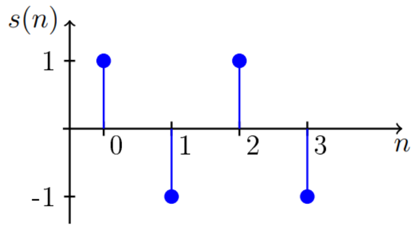
```
MSE_ar4 = 0.9919
```

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

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

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:

$$\boldsymbol{h} = \kappa \boldsymbol{R}_v^{-1}\boldsymbol{s}. \qquad (14.97)$$

where R_v is autocorrelation matrix of noise and κ is the normalisation factor.

Although the maximum SNR can be obtained by any choise of constant κ, we choose the constant by requiring that:

(a) $\boldsymbol{h}^T\boldsymbol{s} = 1$, which yields $\kappa = 1/\boldsymbol{s}^T\boldsymbol{R}_v^{-1}\boldsymbol{s}$

(b) $\mathrm{E}(v_o^2[n]) = \boldsymbol{h}^T\boldsymbol{R}_v\boldsymbol{h} = 1$, which yields $\kappa = 1/\sqrt{\boldsymbol{s}^T\boldsymbol{R}_v^{-1}\boldsymbol{s}}$.

```
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 = 4×1
    0.2174
   -0.2826
    0.2826
   -0.2174
```

The optimum SNR at the output is given by:

$$\mathrm{SNR_o} = a^2\tilde{\boldsymbol{s}}^{\mathrm{T}}\tilde{\boldsymbol{s}} = a^2\boldsymbol{s}^{\mathrm{T}}\boldsymbol{R}_v^{-1}\boldsymbol{s}. \qquad (14.98)$$

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:

$$\mathrm{SNR}_i = \frac{(\text{Value of signal at } n = n_0)^2}{\text{power of noise}} = \frac{s^2(n = n_0)}{r_v(0)}$$

```
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.

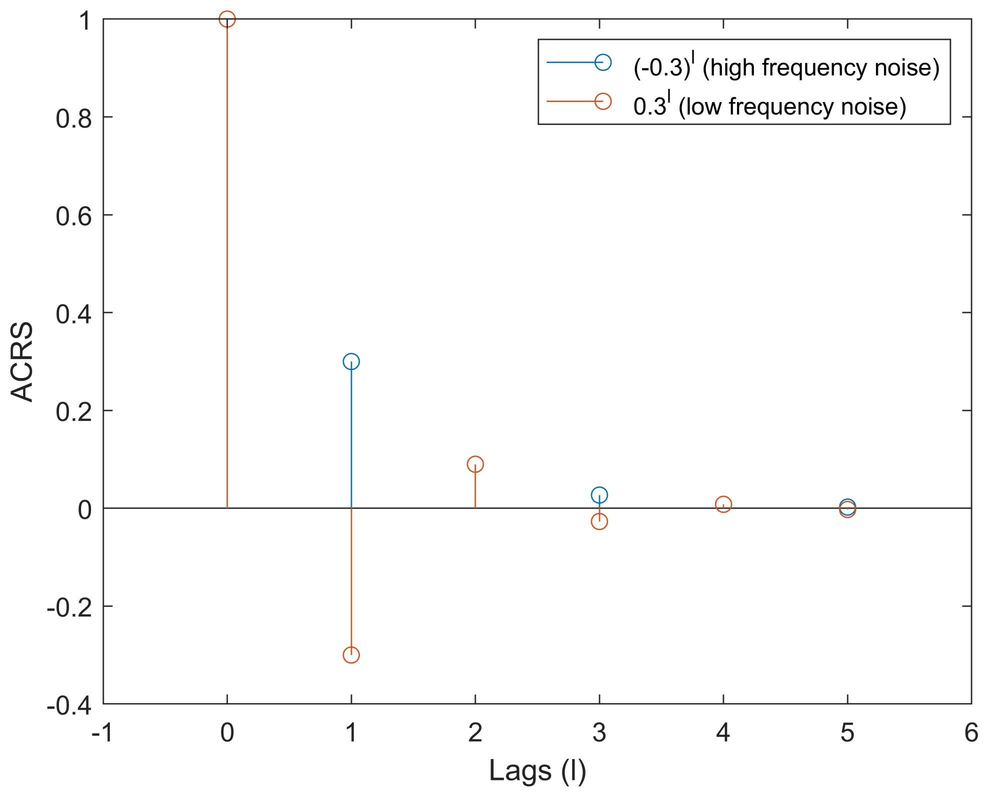
We confirm this by calculating the SNR:

```
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
```

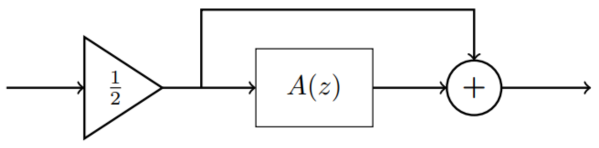
Characteristics of high-frequency noise ACRS vs low-frequency noise.

```
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')
```



Problem 3: Lattice Filters

Consider the digital signal processing system shown below where $A(z)$ is a 2^{nd} order all-pass filter.



1) Explain why a system is a digital notch filter

Since *A*(*z*) is an all-pass �lter 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 difference equations for an all-pole system and an all-zero system with

$$H_{\text{ap}}(z) = \frac{Y(z)}{X(z)} = \frac{1}{A_2(z)}, \quad H_{\text{az}}(z) = \frac{G_2(z)}{Y(z)} = z^{-2}A_2(1/z), \qquad (9.84)$$

where

$$A_2(z) \triangleq 1 + a_1^{(2)}z^{-1} + a_2^{(2)}z^{-2}. \qquad (9.85)$$

From the system function of the all-pass filter, we know that:

$A_2(z) = 1 + 0.91z^{-1} + 0.3z^{-2}$ and $a_2^{(2)} = 0.3$

... should we continue with this?

[?] 3) Determine the group delay in the digital notch filter

....

Problem 4: Random variables

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 probablity $\frac{1}{6}$
2. If the signal is +2 at t_0 then it will remain -2 at $t_0 + 1$ with probablity $\frac{2}{6}$
3. If the signal is -1 at t_0 then it will remain -1 at $t_0 + 1$ with probablity $\frac{2}{6}$
4. If the signal is -1 at t_0 then it will go to +2 at $t_0 + 1$ with probablity $\frac{1}{6}$

If the signal has the value +2 it has probability $\frac{2}{6}$ for staying at +2 than switching to -1.

If the signal has the value -1 it has probability for staying at -1 than switching to +2.

A realisation coded in MATLAB:

```
p_pos = 2/6; % The probability to remain when previous value is positive
```

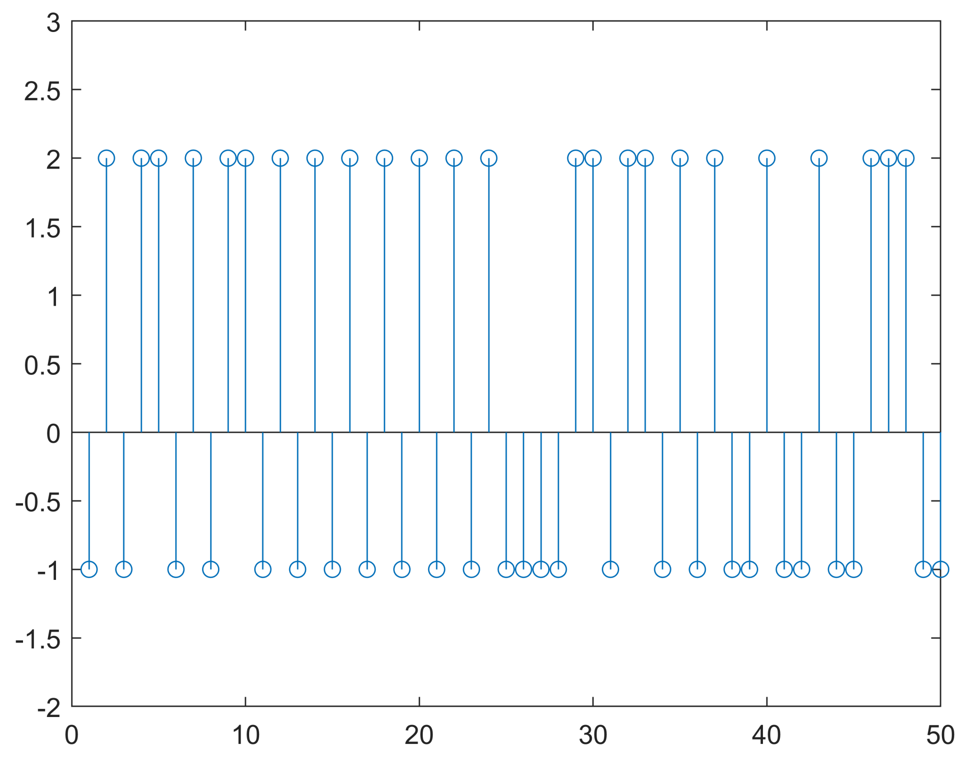
```
p_neg = 2/6; % The probability to remain when previous value is negative

pos_val = 2;
neg_val = -1;

N = 50;
x = zeros(1, N);

x(1) = randsrc(1,1, [pos_val, neg_val; 0.5, 0.5]);
for i = 2:N
    if x(i-1) == pos_val
        x(i) = randsrc(1,1, [pos_val, neg_val; p_pos, 1-p_pos]);
    elseif x(i-1) == neg_val
        x(i) = randsrc(1,1, [neg_val, pos_val; p_neg, 1-p_neg]);
    end
end

% stairs(x);
stem(x)
ylim([neg_val-1, pos_val+1]);
```



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:

$$f_{X_0}(x_0) = \int_{-\infty}^{\infty} f_{X_0,X_1}(x_0,x_1)\,\mathrm{dx}_1$$

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:

$$\begin{aligned}
f_{X_0}(x_0) &= \int_{-\infty}^{\infty} f_{X_0,X_1}(x_0,x_1)dx_1\\
&= \int_{-\infty}^{\infty} \Big(\frac{1}{6}\delta(x_0-2)\delta(x_1+1)+\frac{2}{6}\delta(x_0-2)\delta(x_1-2)\\
&\qquad +\frac{2}{6}\delta(x_0+1)\delta(x_1+1)+\frac{1}{6}\delta(x_0+1)\delta(x_1-2)\Big)dx_1\\
&= \frac{1}{6}\delta(x_0-2)+\frac{2}{6}\delta(x_0-2)+\frac{2}{6}\delta(x_0+1)+\frac{1}{6}\delta(x_0+1)\\
&= \frac{1}{2}\delta(x_0-2)+\frac{1}{2}\delta(x_0+1).
\end{aligned}$$

The expected value of a discrete random variable X with outcomes $x_1, x_2, \cdots, x_k$ 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:

$$\begin{aligned} E[X_0X_1] &= \int_{-\infty}^{\infty} x_0 x_1 f_{X_0,X_1}(x_0, x_1) dx_0 dx_1 \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_0 x_1 \Big(\frac{1}{6}\delta(x_0 - 2)\delta(x_1 + 1) + \frac{2}{6}\delta(x_0 - 2)\delta(x_1 - 2) \\ &\qquad + \frac{2}{6}\delta(x_0 + 1)\delta(x_1 + 1) + \frac{1}{6}\delta(x_0 + 1)\delta(x_1 - 2) \Big) dx_0 dx_1 \\ &= \frac{1}{6}(2)(-1) + \frac{2}{6}(2)(2) + \frac{2}{6}(-1)(-1) + \frac{1}{6}(-1)(2) \\ &= 1. \end{aligned}$$

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 classi�cation is thus **partly deterministic**.

Functions

```
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

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
```