Exam August 2018

Table of Contents

Problem 1: Compute PSD from data using Pisarenko (sinusoid with additive white noise)

Here's a sequence of data from a measurement.

$$\{-11,\ 21,\ 18,\ 62,\ 34,\ -5,\ -14,\ -64,\ -49,\ -35,\ -1,\ 29,\ 37,\ 49,\ 32,\ 12\}$$

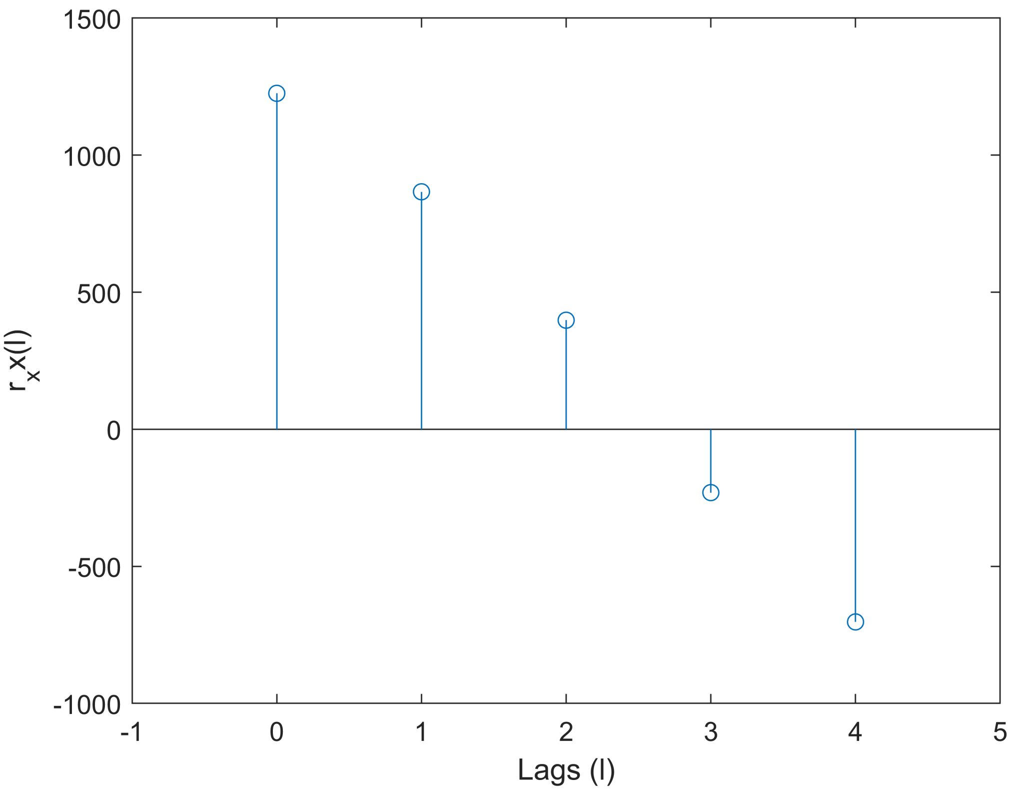
[✔] 1) Compute and plot the autocorrelation for lags 0 to 4

```
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 = 1×5
10^3 ×
    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 (l)')
ylabel('r_xx(l)')
```



[✔] 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 $\mathrm{ARMA}(2p, 2p)$ 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.

$$A(z) = 1 + \sum_{m=1}^{2p} a_m z^{-m} \qquad (14.5.4)$$

Step 4: Solve Eq. (14.5.11) for the signal powers $\{P_i\}$

$$\begin{bmatrix} \cos 2\pi f_1 & \cos 2\pi f_2 & \cdots & \cos 2\pi f_p \\ \cos 4\pi f_1 & \cos 4\pi f_2 & \cdots & \cos 4\pi f_p \\ \vdots & \vdots & & \vdots \\ \cos 2\pi p f_1 & \cos 2\pi p f_2 & \cdots & \cos 2\pi p f_p \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_p \end{bmatrix} = \begin{bmatrix} \gamma_{yy}(1) \\ \gamma_{yy}(2) \\ \vdots \\ \gamma_{yy}(p) \end{bmatrix} \qquad (14.5.11)$$

where

- $\gamma_{\mathrm{yy}}(1), \gamma_{\mathrm{yy}}(2), \cdots, \gamma_{\mathrm{yy}}(p)$ are the estimated autocorrelation values
- $P_i = \frac{A_i^2}{2}$ is the average power of the ith 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:

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

The autocorrelation function of white noise with variance σ_w^2 is given by:

$$r_{\mathrm{ww}}(\ell) = \sigma_w^2 \delta(\ell)$$

Combining these results, the general autocorrelation function of our signal is:

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

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:

$$S(\omega) = \frac{(45.6585)^2}{2}\pi[\delta(\omega - 2\pi \cdot 0.0938) + \delta(\omega + 2\pi \cdot 0.0938)] + 183.2154$$

[✔] 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 $\sigma_w^2 = 183.2154$

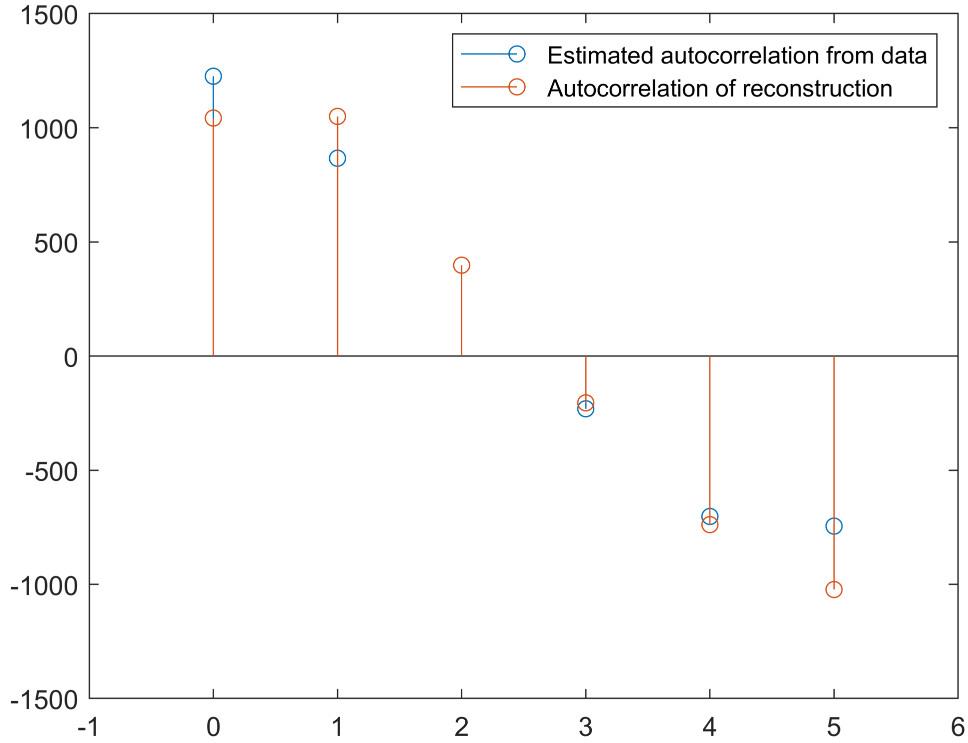
The general autocorrelation function of that signal is:

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

Considering the estimated autocorrelation of the data and the autocorrelation of the reconstruction, we can say that the Pisarenko model is appropriate.

```
% 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 sinusiodal shape. Since Pisarenko model assumes sinuisoidal signal that has been corrupted by white noise then it can be appropriate model.

Problem 2: True/False Questions

For the statements given below, state whether they are true or false and justify your answer for each statement.

[✔] 1) 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 inversable filter to be minimum-phase which means that the filter *and* its inverse must causal and stable.

[✔] 2) 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:

$$r_y(\ell) = E[y(n)y(n-\ell)]$$

$$r_y(\ell) = E[2\,x(n)2x(n-\ell)]$$

$$r_y(\ell) = 4 \cdot E[x(n)x(n-\ell)]$$

$$r_y(\ell) = 4r_x(\ell)$$

[?] 3) Under stationary conditions, a least mean square (LMS) adaptive filter with proper choice of the step-size will converge to the Wiener-Hopf solution.

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:

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

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

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

```
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 = 0.8.^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 = 4×1
    0.9449
```

```
-1.7008
 1.7008
-0.9449
```

The optimum SNR is given by:

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

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
```

Problem 4: Wiener Filtering

A system given by $H(z) = 4 + z^{-1}$ is excited by unit variance white Gaussian noise $w(n)$ to give the signal $s(n)$.

[✔] 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:

$$r_s(\ell) = E[s(n)s(n-\ell)]$$

$$= E[(4x(n) + x(n-1))(4x(n-\ell) + x(n-\ell-1))]$$

$$= E[4x(n)4x(n-\ell)] + E[4x(n)x(n-\ell-1)] + E[x(n-1)4x(n-\ell)] + E[x(n-1)x(n-\ell-1)]$$

$$= 16E[x(n)x(n-\ell)] + 4E[x(n)x(n-\ell-1)] + 4E[x(n-1)x(n-\ell)] + E[x(n-1)x(n-\ell-1)]$$

```
syms n l w(n)
expand((4*w(n) + w(n-1)) * (4*w(n-l) + w(n-1-l)))
```

ans = $16\,w(n-l)\,w(n) + 4\,w(n-l-1)\,w(n) + 4\,w(n-l)\,w(n-1) + w(n-l-1)\,w(n-1)$

$$= 16r_w(\ell) + 4r_w(\ell-1) + 4r_w(\ell+1) + r_w(\ell)$$

Since $r_w(\ell-1) = r_w(\ell+1)$

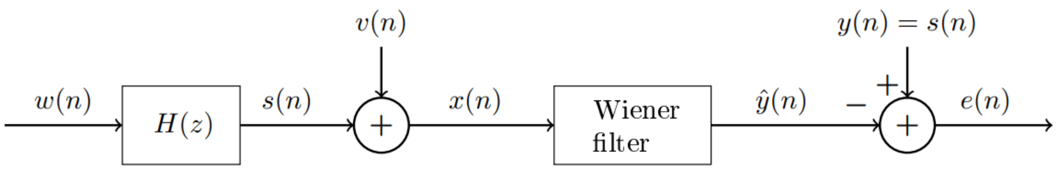
$$r_s(\ell) = 17r_w(\ell) + 8r_w(\ell-1)$$

Since the autocorrelation of unit variance white noise is $r_{xx}(\ell) = \sigma_x^2\delta(\ell) = \delta(\ell)$:

$$r_s(\ell) = 17\delta(\ell) + 8\delta(\ell-1)$$

[✔] 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.



2. Determine the 3×3 autocorrelation matrix R_x and the 3×1 crosscorrelation vector $\mathbf{g} = E[\mathbf{x}(n)y(n)]$ 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 $x(n)$:

$$r_x(\ell) = E[x(n)x(n-\ell)]$$

$$= E[(s(n)+v(n))(s(n-\ell)+v(n-\ell))]$$

$$= E[s(n)s(n-\ell)+s(n)v(n-\ell)+v(n)s(n-\ell)+v(n)v(n-\ell)]$$

$$= E[s(n)s(n-\ell)]+E[s(n)v(n-\ell)]+E[v(n)s(n-\ell)]+E[v(n)v(n-\ell)]$$

$$= r_s(\ell)+2r_{\mathrm{sv}}(\ell)+r_v(\ell)$$

$= r_s(\ell)+r_v(\ell)$ (because $w(n)$ and $v(n)$ are uncorrelated so $r_{\mathrm{sv}}(\ell)=0$)

$= 17\delta(\ell)+8\delta(\ell-1)+3\delta(\ell)$ -- $r_v(\ell)=3\delta(\ell)$ 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:

$$r_{\mathrm{xy}}(\ell) = E[x(n)y(n-\ell)]$$

$$= E[(s(n)+v(n))(4w(n-\ell)+w(n-\ell-1))]$$

$$= 4E[s(n)w(n-\ell)]+E[s(n)w(n-\ell-1)]+4E[v(n)w(n-\ell)]+E[v(n)w(n-\ell-1)]$$

$$= 4r_{\mathrm{sw}}(\ell)+r_{\mathrm{sw}}(\ell-1)+4r_{\mathrm{vw}}(\ell)+r_{\mathrm{vw}}(\ell-1)$$

We assume that $w(n)$ and $v(n)$ are uncorrelated so $r_{\mathrm{vw}}(\ell)=0$:

$$= 4r_{\mathrm{sw}}(\ell)+r_{\mathrm{sw}}(\ell-1)+0+0$$

According to Eq. 13.100, the cross-correlation $r_{\mathrm{sw}}(\ell)=\sigma_w^2 h(\ell)=h(\ell)$ 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:

$$\hat{y}(n) = h_1x(n)+h_2x(n-1)+h_3x(n-2)$$

The optimum Wiener filter to estimate a random process is given by Eq. 14.109:

$$\boldsymbol{h}_{\mathrm{o}} = \boldsymbol{R}_x^{-1}\boldsymbol{g}, \qquad (14.109)$$

where $\boldsymbol{R}_x$ is the correlation matrix of a random vector $\boldsymbol{x}$ and $\boldsymbol{g}$ is the cross-correlation vector between $\boldsymbol{x}$ and y

```
r_xx = [20, 8, 0];
R_xx = toeplitz(r_xx)
```

```
R_xx = 3×3
    20     8     0
     8    20     8
     0     8    20
```

```
g = [4, 1, 0]'
```

```
g = 3×1
     4
     1
     0
```

```
h_opt = R_xx\g
```

```
h_opt = 3×1
    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?

```
r_xx = [20, 8];
R_xx = toeplitz(r_xx);
g = [4, 1]';
h_opt = R_xx\g
```

```
h_opt = 2×1
    0.2143
   -0.0357
```

Problem 5: 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 is given by:

$$\int_{-\infty}^{\infty} f_X(x)\,\mathrm{dx} = 1$$

We need to compute:

$$\int_{-1}^{1} \alpha x^2 + \frac{1}{4}\,\mathrm{dx}$$

For convenience and to avoid silly mistakes, use MATLAB:

```
syms x a
int(a*x^2 + 1/4, x, -1, 1)
```

```
ans =
```

$$\frac{2\,a}{3} + \frac{1}{2}$$

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:

$$a = \frac{3}{4}$$

[✔] 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:

$$E[g(x)] = \int_{-\infty}^{\infty} g(x) f_X(x)\,\mathrm{dx} = \int_{-1}^{1} e^{-|x|} \cdot \frac{3}{4} x^2 + \frac{1}{4}\,\mathrm{dx}$$

```
syms x
f = exp(-abs(x)) * 3/4 * x^2 + 1/4
```

f =

$$\frac{3\,x^2\,\mathrm{e}^{-|x|}}{4} + \frac{1}{4}$$

```
int(f , x, -1, 1 )
```

ans =

$$\frac{7}{2} - \frac{15\,\mathrm{e}^{-1}}{2}$$

```
vpa(int(f , x, -1, 1 ))
```

ans = $0.74090419121418258803357172378904$

The answer is:

$$E[g(x)] \approx 0.74$$

Functions

```
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
```