Module 11 - Homework 11

Colt Thomas

April 26, 2021

Problem 1

A discrete-time WSS random process X[n]is defined by the difference equation

$$X[n] = \frac{1}{2}X[n-1] + U[n] - \frac{1}{2}U[n-1]$$

where U[n]is a discrete-time white noise random process with variance $\sigma_U^2 = 1$. Find the auto correlation sequence(ACS) and power spectral density (PSD) of X[n]. Comment on the results.

We can find the power spectral density of X[n] by first finding the frequency response of the system:

$$X[n] - \frac{1}{2}X[n-1] = U[n] - \frac{1}{2}U[n-1]$$

$$X(f) - \frac{1}{2}X(f)e^{-j2\pi f} = U(f) - \frac{1}{2}U(f)e^{-j2\pi f}$$

$$(1 - \frac{1}{2}e^{-j2\pi f})X(f) = (1 - \frac{1}{2}e^{-j2\pi f})U(f)$$

Remember that the frequency response is just output over input:

$$H(f) = \frac{X(f)}{U(f)} = \frac{(1 - \frac{1}{2}e^{-j2\pi f})}{(1 - \frac{1}{2}e^{-j2\pi f})} = 1$$

The PSD for a LTI WSS random process can be given by solving for:

$$S_X(f) = |H(f)|^2 S_U(f)$$

By definition, the power spectral density for white noise is

$$S_U(f) = \frac{N_o}{2} = \sigma_U^2$$

We then get the following for the power spectral density of X[n]:

$$\boxed{S_X(f) = |(1)|^2(\sigma_U^2) = 1}$$

The autocorrelation sequence can be found by taking the inverse fourier transform of the power spectral density:

$$\boxed{r_X(\tau) = FT^{-1}\{S_X(f)\} = \delta(\tau)}$$

I noticed that the frequency response is the same as the power spectral density, and the autocorrelation sequence also happens to be the same as our impulse response $h[n] = \delta[n]$ but in continuous time.

Problem 2

Consider the prediction of a randomly phased sinusoid whose ACS is $r_X[k] = cos(2\pi f_o k)$. For $M = 2$, solve the Wiener-Hopf equations to determine the optimal linear predictor and also the minimum mean square error (MSE).Hint:The minimum MSE is zero. Use the trigonometric identity $cos(2\theta) = 2cos^2(\theta) - 1$ to establish this.

$$\begin{bmatrix} r_X[0] & r_X[1] \\ r_X[1] & r_X[0] \end{bmatrix} \begin{bmatrix} h[0] \\ h[1] \end{bmatrix} = \begin{bmatrix} r_X[1] \\ r_X[2] \end{bmatrix}$$

$$\begin{bmatrix} cos(0) & cos(2\pi f_o) \\ cos(2\pi f_o) & cos(0) \end{bmatrix} \begin{bmatrix} h[0] \\ h[1] \end{bmatrix} = \begin{bmatrix} cos(2\pi f_o) \\ cos(4\pi f_o) \end{bmatrix}$$

$$\begin{bmatrix} h[0] & cos(2\pi f_o)h[1] \\ cos(2\pi f_o)h[0] & h[1] \end{bmatrix} = \begin{bmatrix} cos(2\pi f_o) \\ cos(4\pi f_o) \end{bmatrix}$$

I found it easier to just solve the system of equations rather than find that inverse matrix. I got:

$$h[0] = cos(2\pi f_o) - cos(2\pi f_o)h[1]$$

$$cos(2\pi f_o)(cos(2\pi f_o) - cos(2\pi f_o)h[1]) + h[1] = cos(4\pi f_o)$$

$$h[1](1 - cos^2(2\pi f_o)) = 2cos^2(2\pi f_o) - 1$$

$$h[1] = -1$$

$$h[0] = 2cos(2\pi f_o)$$

$$\boxed{\begin{bmatrix} h[0] \\ h[1] \end{bmatrix} = \begin{bmatrix} -1 \\ 2cos(2\pi f_o) \end{bmatrix}}$$

We still need to find the MSE:

$$MSE_{min} = r_X[0] - \sum_{k=0}^{M-1} h_{opt}[k] r_X[k+1]$$

$$MSE_{min} = r_X[0] - h_{opt}[0] r_X[1] - h_{opt} r_X[2]$$

$$MSE_{min} = cos(0) - h_{opt}[0]cos(2\pi f_o) - h_{opt}[1]cos(4\pi f_o)$$
$$MSE_{min} = cos(0) - (2cos(2\pi f_o))cos(2\pi f_o) - (-1)cos(4\pi f_o)$$
$$MSE_{min} = 1 - (2cos^2(2\pi f_o)) + cos(4\pi f_o)$$
$$MSE_{min} = 1 - (2cos^2(2\pi f_o)) + (2cos^2(2\pi f_o) - 1)$$
$$MSE_{min} = 2cos^2(2\pi f_o) + 2cos^2(2\pi f_o)$$
$$\boxed{MSE_{min} = 0}$$

Problem 3

An LTI system has the impulse response $h(\tau) = e^{-2\tau}$ for $\tau \geq 0$ and zero for $\tau < 0$. If continuous white noise with autocorrelation function $r_U(\tau) = \frac{N_o}{2}\delta(\tau)$ is input to the system, what is the PSD of the output random process?

The PSD for a LTI WSS random process can be given by solving for:

$$S_X(f) = |H(f)|^2 S_U(f)$$

To find the frequency response given the impulse response, we can solve for:

$$H(f_o) = \int_{-\infty}^{\infty} h(\tau)e^{-j2\pi f_o \tau} d\tau$$
$$H(f_o) = \int_{0}^{\infty} e^{-2\tau} e^{-j2\pi f_o \tau} d\tau$$

Notice that this is just a Fourier transform. There is an identity for doing the transform of an exponential signal, which yields:

$$H(f_o) = \frac{1}{2 + j2\pi f_o}$$

For the input signal, given the autocorrelation function, we can find $S_U(\tau)$ by taking the Fourier transform of $r_U(\tau)$. Luckily the transform of a delta is straightforward:

$$S_U(\tau) = FT\{r_U(\tau)\} = \frac{N_o}{2}$$

Combine the results to get the PSD of our random process:

$$S_X(f) = \left|\frac{1}{2 + j2\pi f}\right|^2 \frac{N_o}{2}$$
$$S_X(f) = \left(\frac{1}{2\sqrt{\pi^2 f^2 + 1}}\right)^2 \frac{N_o}{2}$$
$$\boxed{S_X(f) = \frac{N_o}{8(\pi^2 f^2 + 1)}}$$

Problem 4

Write a Matlab program to simulate a three-state Markov with transition probability matrix

$$P = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/3 & 1/3 & 1/3 \\ 1/6 & 1/2 & 1/3 \end{bmatrix}$$

Part A

Assuming that the process starts in the third state, generate a sequence of 1000 states. Estimate the steady state probability distribution, π, using the sequence you generated.

I attached my code in the appendix, and was able to produce the following results for a steady state probability distribution:

$$\boxed{\begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}}$$

Part B

Does it agree with the theoretical answer? Let's calculate the theoretical answer:

$$\pi P = \pi$$

$$\begin{bmatrix} \pi_1 & \pi_2 & \pi_3 \end{bmatrix} \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/3 & 1/3 & 1/3 \\ 1/6 & 1/2 & 1/3 \end{bmatrix} = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 \end{bmatrix}$$

Solve for this system of equations:

$$\pi_1 + \pi_2 + \pi_3 = 1$$

$$1/2\pi_1 + 1/2\pi_2 = \pi_1$$

$$1/3\pi_1 + 1/3\pi_2 + 1/3\pi_3 = \pi_2$$

We get:

$$\boxed{\begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}}$$

Part C

Does the steady state distribution depend on the starting state of the process?

No. I tried changing the starting state called currState in the Matlab code, and would still get the same steady state distribution regardless.

Matlab Code

```
%% %%%%%%%%%%%%%%
% Problem 4
%%%%%%%%%%%%%%%%
P = [1/2,1/2,0;1/3,1/3,1/3;1/6,1/2,1/3];

% Experiment setup
N = 1000;
currState = 3; % also considered the starting state
stateSequence = zeros(1,N+1);
stateSequence(1) = currState; % make the first index  be our starting state
steadyStateResults = zeros(3,3); % this should represent P as N->infinity

% Experiment execution
for i=2:N
    % generate a uniform random number
    r = rand;

    % convert our probability density function (for the current state) to
    % an indexed CDF
    CDF = cumsum([0,P(currState,:)]);

    % create a logical array indicating whether the rand is higher than the
    % value at respective indices, then sum the results to give the output
    % state
    stateSequence(i) = sum(rand >= CDF);

    % Update steady state results matrix
    steadyStateResults(currState,stateSequence(i)) = steadyStateResults(currState,stateSequence(i)) + 1;
    % Update next state
    currState = stateSequence(i);
end

% normalize the steady state results. It should resemble the transition
% probability matrix
steadyStateResults = [steadyStateResults(1,:)./sum(
```

```
    steadyStateResults(1,:)); ...
                        steadyStateResults(2,:)./sum(steadyStateResults(2,:)); ...
                        steadyStateResults(3,:)./sum(steadyStateResults(3,:))];

% Find the steady state probability distribution pi*P=pi
sspd = [1,1,1,1;...
     steadyStateResults(1,1)-1,steadyStateResults(1,2),steadyStateResults(1,3),0;...
     steadyStateResults(2,1),steadyStateResults(2,2)-1,steadyStateResults(2,3),0]
sspd = rref(sspd) % solve the system of equations by reducing the matrix

% steady state probability distribution results
pi_1 = sspd(1,4);
pi_2 = sspd(2,4);
pi_3 = sspd(3,4);
```