12 Rec 12: WSS Processes; Predictor Design

Directions: Your instructor will spend the the first 40 minutes of the recitation period working some review problems and going over one or more Matlab experiments in the following. During the last 10 minutes of recitation, your proctor will give you a "Lab Form" that your recitation team completes, signs, and turns in. See the last page for an indication of what you will be asked to do on the Lab Form.

Due to time limitations, only a part of the following can be covered during the recitation period. However, you might want in the future to try some of the uncovered experiments on your own. They could give skills useful on some future homework problems and could lend insight into your understanding of the course from an experimental point of view.

This Week's Topics.

- Mean Function of a Process
- WSS Process Power
- WSS Autocorrelations Via Time Averaging
- WSS Autocorrelations Via Space Averaging
- Linear Predictor Design

12.1 Exp 1: Mean Function of a Process

Any process $(X(t))$ has a *mean function* $\mu_X(t)$, defined by

$$\mu_X(t) \triangleq E[X(t)], \quad \text{for all } t$$

This experiment consists of two parts: in Part One, we discuss the mean function of a nonstationary process; in Part Two, we discuss the mean of a wide-sense stationary process (WSS process), which is a constant.

12.1.1 Part One: $\mu_X(t)$ of Nonstationary Process $X(t)$

There are many applications in which one may want to estimate the mean function of a nonstationary process. Here are a couple of these applications:

Application 1: Suppose $x(t)$ is a deterministic input signal to an additive noise communication channel. The channel output signal $Y(t)$ is a random signal which can be modeled as

$$Y(t) = x(t) + Z(t)$$

where the random process $(Z(t))$ is the channel noise. If the channel noise has mean zero, then $x(t) = \mu_Y(t)$ and trying to estimate the input signal is equivalent to estimating the mean function of the Y process.

Application 2: Let (X_n) be a DTRP (discrete time random process) in which, for each time n, X_n models the Dow Jones final closing value on day n at the New York Stock exchange. You might find wild fluctuations in X_n from day to day. It might be more beneficial for stock market forecasting purposes to find an underlying "trend", a deterministic signal x_n less wild than X_n, which might be modeled as

$$X_n = x_n + Z_n$$

where Z_n is 0-mean noise. With this kind of modeling, x_n is the mean function of RP X_n.

Example 1. Run the following script:

```
t=1:100;
x=20*t + 10000 +200*randn(1,length(t));
plot(t,x)
```

What you see on your screen is an initial piece of a realization of a process $X(t)$. (Think of this plot as the plot of 100 consecutive daily Dow Jones closing figures, if you want.) What is the "trend"? That is, what is an estimate of the mean function $\mu_X(t)$, which is a deterministic signal? Run the following Matlab code, which "space-averages" across 1000 different realizations to estimate the trend:

```
clear;
t=1:100;
S=zeros(1,length(t));
for i=1:1000
realization=20*t + 10000 +200*randn(1,length(t));
S=S+realization;
end
mean_function_estimate=S/1000;
plot(t,mean_function_estimate)
```

Is the trend more apparent now?

Example 2. Suppose one wants to estimate the mean function but one has only one realization to work with. (This might be a more realistic assumption for the Dow Jones problem.) One can't just simply average across the whole realization to approximate the mean function, because then one would obtain a constant. Instead, one could subdivide the time axis into small subintervals and then average the samples over each subinterval. This is what we do below for the pseudorandomly generated realization from Example 1. One change though—this type of estimation requires many more samples along the realization than we were using before; instead of 100 samples, we use 99000, which we subdivide into 1000 intervals containing 99 samples each:

```
clear;
t=1.001:.001:100;
x=20*t + 10000 + 200*randn(1,length(t));
```

```
for i=1:1000
subinterval=(i-1)*99+1:i*99;
tt(i)=mean(t(subinterval));
xx(i)=mean(x(subinterval));
end
time_axis=tt;
mean_function_estimate=xx;
plot(time_axis,mean_function_estimate)
```

12.1.2 Part Two: Mean of WSS Process

The mean function of a WSS process is a constant. Let us take the WSS process as a discrete time process X_n. Then its mean is the number μ_X such that

$$E[X_n] = \mu_X$$

for every n. In other words, every 1-D cross-section random variable of the process has the same mean, which we denote by μ_X. There are two methods for estimating μ_X, namely, "space averaging" and "time averaging," described as follows:

Space Averaging Method For Estimating μ_X: For a fixed time n, estimate μ_X via the formula

$$\hat{\mu}_X = J^{-1} \sum_{j=1}^{J} x_n^j,$$

for a large J, where, for each $j = 1, 2, \cdots, J$, x_n^j is a different realization of the process X_n.

Time Averaging Method For Estimating μ_X: For a realization x_n of the process X_n, compute the following estimate of μ_X via time averaging:

$$\hat{\mu}_X = N^{-1} \sum_{n=1}^{N} x_n \tag{1}$$

In formula (1), $[x_1, x_2, \cdots, x_N]$ represents a large segment of the realization. We are assuming here that the estimate (1) does not vary appreciably from realization to realization. (This means we have an *ergodic process*. We will cover ergodic WSS processes in a future lecture.) If the right side of (1) varies from realization to realization, then we would have to average up the right hand sides of (1) for many different realizations in order to obtain our estimate of μ_X. (This is the nonergodic case.)

Example 3. Let (X_n) be the IID process in which all the X_n's have the same exponential distribution with mean 1/4. We let (Y_n) be the process

$$Y_n = 2X_n + 3X_{n-1} \tag{2}$$

The Y process is WSS (because the X process is, and anytime you pass a WSS process through a time-invariant stable linear filter, you get a WSS output process; we will prove

this in Chapter 11). In this example, we estimate the mean μ_Y of the Y process both by space-averaging and time-averaging. Run the following code, which estimates μ_Y by space-averaging:

```
clear;
for i=1:10000
X1=-(0.25)*log(rand(1,1)); X2=-(0.25)*log(rand(1,1));
Y2space(i)=2*X2+3*X1;
end
estimate_muY=mean(Y2space)
```

Can you determine what the exact value of μ_Y is using the equation (2)? Is your estimate generated by the preceding Matlab code very far off from that? Now, run the following Matlab code, which estimates μ_Y via time-averaging:

```
x=-(0.25)*log(rand(1,50000));
n=2:50000;
y=2*x(n)+3*x(n-1);
estimate_muY=mean(y)
```

Re-run these lines several times to make sure you are getting about the same estimate. (This ensures that we have an ergodic process—the estimate is not changing appreciably if we re-compute the estimate using different realizations.)

Example 4. Let (X_n) be the same process as used in Example 3. If we pass (X_n) through a stable linear time-invariant filter with impulse response function (h_n), resulting in output process Y, then Chapter 11 will tell us that μ_X and μ_Y are related by the formula

$$\mu_Y = \mu_X[\sum_n h_n] \tag{3}$$

Let (Y_n) be the process

$$Y_n = X_n + (0.75)Y_{n-1} \tag{4}$$

Run the following Matlab code, which will estimate μ_Y via time-averaging:

```
clear;
x=-(0.25)*log(rand(1,10000));
y(1)=0;
for i=2:10000
y(i) = x(i) + (0.75)*y(i-1);
end
mean(y)
```

As in Example 3, re-run the code more than once to make sure the ergodic assumption is valid (that is, you should get about the same estimate of μ_Y each run). Recall from EE 3015 that the impulse response h_n of the filter used to obtain (Y_n) from (X_n) is the response of the filter to the delta function input. With this in mind, you can now run the following code which will obtain an approximation to $\sum_n h_n$ in equation (3):

```
delta=[0 1 zeros(1,10000)];
h(1)=0;
for i=2:10000
h(i) = delta(i) + (0.75)*h(i-1);
end
sum(h)
```

Does formula (3) seem to be true? (Recall that the X process has mean 1/4.)

12.2 Exp 2: WSS Process Power

Let

$$X_n, \quad n = 1, 2, 3, \cdots$$

be a DT WSS process. The power figure P_X for the process X_n is defined by the second moment

$$P_X \triangleq E[X_n^2],$$

where, in this calculation, the discrete time variable n is held fixed (by the WSS assumption, you will obtain the same second moment for all n). If you have already been given the autocorrelation function $R_X(\tau)$ of the process, then P_X can be conveniently obtained via the formula

$$P_X \triangleq R_X(0).$$

Similarly to what we did in Experiment 1 in estimating μ_X, you can also estimate P_X by either space averaging or time averaging:

Space Averaging Method For Estimating P_X: For a fixed time n, approximate $E[X_n^2] = P_X$ as a space-average

$$\hat{P}_X \approx J^{-1} \sum_{j=1}^{J} (x_n^j)^2,$$

for a large J, where, for each $j = 1, 2, \cdots, J$, x_n^j is a different realization of the process X_n.

Time Averaging Method For Estimating P_X: For a realization x_n of the process X_n, evaluate the time-average

$$N^{-1} \sum_{n=1}^{N} x_n^2 \tag{5}$$

over a large segment $[x_1, x_2, \cdots, x_N]$ of the realization. See if this quantity varies appreciably from realization to realization. If it does not, then this average can be taken as an estimate of P_X. (That is, you have an ergodic process.) If the time average (5) varies from realization to realization, then you have a nonergodic process and then you must average up these time averages for many different realizations in order to get a true estimate of P_X.

Example 5. We consider the sum of two random sinusoids

$$X_n = A_1 \cos(n + \Theta_1) + A_2 \cos(2n + \Theta_2), \quad n = 1, 2, \cdots$$

in which the amplitude A_1 is Gaussian with mean 0 variance 1, the amplitude A_2 is Gaussian with mean 0 variance 4, and the two phases Θ_1 and Θ_2 are each uniformly distributed in $[0, 2\pi]$. (The amplitudes and phases are independent.) It can be shown that this process is WSS, so we can use both the space and time averaging techniques for estimating P_X. Run the following Matlab code, which estimates P_X using the space averaging method, where the space average is computed at time $n = 1$:

```
clear;
n=1;
for i=1:5000
a1=randn(1,1); a2=2*randn(1,1);
theta1=2*pi*rand(1,1); theta2=2*pi*rand(1,1);
x1=a1*cos(n+theta1)+a2*cos(2*n+theta2);
powerterm(i)=x1^2;
end
PX = mean(powerterm)
```

Just to double-check that we can do the space-averaging at any time, run the following Matlab code where the space-averaging power estimate is computed at time $n = 2$:

```
n=2;
for i=1:5000
a1=randn(1,1); a2=2*randn(1,1);
theta1=2*pi*rand(1,1); theta2=2*pi*rand(1,1);
x2=a1*cos(n+theta1)+a2*cos(2*n+theta2);
powerterm(i)=x2^2;
end
PX = mean(powerterm)
```

Now, run the following code, to see what time-average power a realization of this process gives:

```
n=1:100;
a1=randn(1,1); a2=2*randn(1,1);
theta1=2*pi*rand(1,1); theta2=2*pi*rand(1,1);
x=a1*cos(n+theta1)+a2*cos(2*n+theta2);
power=mean(x.^2)
```

Run this code several times to see if the power varies (it should, because we have a nonergodic process). Now run the following Matlab code, in which you average up 500 of these power figures:

```
for j=1:500
n=1:100;
```

```
a1=randn(1,1); a2=2*randn(1,1);
theta1=2*pi*rand(1,1); theta2=2*pi*rand(1,1);
x=a1*cos(n+theta1)+a2*cos(2*n+theta2);
power(j)=mean(x.^2);
end
mean(power)
```

Run this code again to see if the result changes appreciably (it should not). Does this time-averaging power figure approximately agree with the space-averaging power figure that you computed at the beginning of this example?

12.3 Exp 3: WSS Autocorrelations Via Time Averaging

Let

$$X_n, \quad n = 0, \pm 1, \pm 2, \pm 3, \cdots$$

be a bilateral DT WSS process. (You can always consider a WSS process to have started at time $-\infty$.) Then, for each integer "lag" τ, the autocorrelation $R_X(\tau)$ at lag τ is defined by

$$R_X(\tau) \triangleq E[X_n X_{n-\tau}]$$

(This computation can be done for any fixed n and will yield the same answer.) In Experiments 1 and 2, we used the time averaging method to estimate the process mean μ_X and the process power P_X. In this experiment, we use the time averaging method to estimate values of the autocorrelation function $R_X(\tau)$. (In the next experiment, we use the space averaging method to estimate $R_X(\tau)$ values.) Here is how the time averaging method works:

Time Averaging Method For Estimating $R_X(\tau)$: Observe a sufficiently long piece

$$[x_1, x_2, x_3, \cdots, x_N]$$

of a realization x_n, and then estimate $R_X(\tau)$ for a fixed τ by:

$$\hat{R}_X(\tau) = N^{-1} \sum_{n=1}^{N} x_n x_{n-\tau} \tag{6}$$

If we assume that the process X_n is ergodic, then looking at this one realization is enough. (Subsequent realizations of the process will yield roughly the same $\hat{R}_X(\tau)$ value via time averaging.) It should be pointed out that the size of N that you need in (6) is dependent upon the fixed value of τ. If you make τ somewhat larger, then you would probably also have to make N somewhat larger in order for the expression (6) to give a good estimate.

Example 6. We can obtain a WSS process by putting a white noise process through an FIR filter. In this example, we filter the Gaussian white noise process (Z_n) in which each Z_n is a standard Gaussian RV as follows:

$$X_n = 2Z_n - (3.5)Z_{n-1} + 4Z_{n-2} + (1.5)Z_{n-4}$$

Run the following code, which will find an approximation $\hat{R}_X(\tau)$ to $R_X(\tau)$ for $\tau = 0, 1, 2, 3, 4$ via the time averaging method:

```
clear;
z=randn(1,100000);
n=5:100000;
x=2*z(n)-(3.5)*z(n-1) + 4*z(n-2) + (1.5)*z(n-4);
Rxhat0 = mean(x.*x)
Rxhat1 = mean(x(2:length(x)).*x(1:length(x)-1))
Rxhat2 = mean(x(3:length(x)).*x(1:length(x)-2))
Rxhat3 = mean(x(4:length(x)).*x(1:length(x)-3))
Rxhat4 = mean(x(5:length(x)).*x(1:length(x)-4))
```

You will see the autocorrelation estimates printed out on your screen. Let me guide you in calculating the exact value of $R_X(2)$ so that you can compare to estimate $\hat{R}_X(2)$ on your screen. Since $R_X(2) = E[X_n X_{n-2}]$, we have to multiply

$$2Z_n - (3.5)Z_{n-1} + 4Z_{n-2} + (1.5)Z_{n-4}$$

by

$$2Z_{n-2} - (3.5)Z_{n-3} + 4Z_{n-4} + (1.5)Z_{n-6}$$

You only have to worry about a product of a term from the 1st sum times a term from the second sum if both terms are of the form Z_i for the same i (why?). This gives us the products

$$8Z_{n-2}^2 + 6Z_{n-4}^2$$

Taking the expected value, you get $8 + 6 = 14 = R_X(2)$. Compare this to the estimate $\hat{R}_X(2)$. Compute one or more other values $R_X(\tau)$ and compare to the corresponding $\hat{R}_X(\tau)$. Save your estimates $\hat{R}_X(\tau)$ (you will compare them to the space averaging estimates in Experiment 4).

Example 7. Let (Z_n) be the same process as in Example 6. We generate from (Z_n) the following process $(Y_n : n = 1, 2, \cdots)$, via second-order autoregresssive filtering:

$$Y_n = \begin{cases} 0, & n = 1, 2 \\ Z_n + (0.75)Y_{n-1} - (0.2)Y_{n-2}, & n > 2 \end{cases}$$

The process (Y_n) is not WSS (because of the initial conditions at times $n = 1, 2$), but it looks more and more like a WSS process as time $n \to \infty$. Therefore, the autocorrelations $R_Y(\tau)$ can be taken as

$$R_Y(\tau) = \lim_{n\to\infty} E[Y_n Y_{n-\tau}]$$

We will not have a general method for explicitly computing these autocorrelations until we get to Chapter 11. However, we can approximate these autocorrelations via time averaging right now. Run the following code to get approximations to $R_Y(0), R_Y(1), R_Y(2)$:

```
clear;
z=randn(1,10000);
y(1)=0; y(2)=0;
for n=3:10000
y(n)=z(n) + (0.75)*y(n-1) - (0.2)*y(n-2);
```

```
end
Ryhat0 = mean(y.*y)
Ryhat1 = mean(y(2:length(y)).*y(1:length(y)-1))
Ryhat2 = mean(y(3:length(y)).*y(1:length(y)-2))
```

Save the approximations $\hat{R}_Y(\tau)$ ($\tau = 0, 1, 2$) that you see on your screen (for Experiment 4).

Example 8. Let Θ be a RV uniformly distributed in the interval $[0, 2\pi]$. Let $(W_n : n = 1, 2, 3, \cdots)$ be the DT process

$$W_n = \cos(n + \Theta), \quad n = 1, 2, 3, \cdots$$

By running the following code, you approximate some autocorrelation values $R_W(\tau)$:

```
n=1:100000;
w=cos(n+2*pi*rand(1,1));
Rwhat0 = mean(w.*w)
Rwhat1 = mean(w(2:length(w)).*w(1:length(w)-1))
Rwhat2 = mean(w(3:length(w)).*w(1:length(w)-2))
Rwhat3 = mean(w(4:length(w)).*w(1:length(w)-3))
Rwhat4 = mean(w(5:length(w)).*w(1:length(w)-4))
```

12.4 Exp 4: WSS Autocorrelations Via Space Averaging

Let X_n be a WSS process. This experiment illustrates the use of the space averaging method for estimating values of the autocorrelation function $R_X(\tau)$.

Suppose for a fixed τ that you want to estimate the autocorrelation value $R_X(\tau)$. The space averaging method estimates $R_X(\tau)$ as follows:

$$\hat{R}_X(\tau) = \textit{ Average of } X_N X_{N-\tau} \textit{ across a large number of realizations} \tag{7}$$

In equation (7), the integer N is fixed. Let us discuss the choice of the integer N:

- If the integer time variable n in the WSS process X_n runs from $-\infty$ to ∞ (bilateral process), then the fixed integer N could be any integer.
- If the WSS process X_n starts at time $n = 0$, then N would be chosen to be $\geq \tau$.
- Suppose the process X_n arises from recursive stable linear filtering of a WSS process starting at time $n = 0$. Then, X_n will not be WSS but will instead become approximately WSS as n becomes larger and larger; this is because of the presence of a transient component of the random signal X_n which dies out as n becomes large. In this case, one would choose the fixed integer N in the space averaging method to be large enough so that the approximate WSS behavior is valid.

Example 9. Let (X_n) be the process from Example 6. Run the following Matlab script to obtain space averaging estimates for the same autocorrelation values used in Example 6. (The script uses $N = 9$ and 5000 realizations to implement the space averaging estimate given by (7).)

```
clear;
M=zeros(5000,5); %each row will store X5,X6,X7,X8,X9 from a realization
for i=1:5000
z=randn(1,9);
n=5:9;
x=2*z(n)-(3.5)*z(n-1) + 4*z(n-2) + (1.5)*z(n-4);
M(i,1:5)=x;
end
Rxhat0 = mean(M(:,5).*M(:,5))
Rxhat1 = mean(M(:,4).*M(:,5))
Rxhat2 = mean(M(:,3).*M(:,5))
Rxhat3 = mean(M(:,2).*M(:,5))
Rxhat4 = mean(M(:,1).*M(:,5))
```

Compare these estimates with the time averaging estimates found in Example 6.

Example 10. Let Y be the process in Example 7. Starting at $N = 20$, the process is almost WSS. Run the following code, which forms estimates of $R_Y(0), R_Y(1), R_Y(2)$ via (7) with $N = 20$ and 5000 realizations:

```
clear;
M=zeros(5000,3); %each row will store Y18,Y19,Y20 from a realization
for i=1:5000
z=randn(1,20);
y(1)=0; y(2)=0;
for n=3:20
y(n)=z(n) + (0.75)*y(n-1) - (0.2)*y(n-2);
end
M(i,1:3)=y(18:20);
end
Ryhat0 = mean(M(:,3).*M(:,3))
Ryhat1 = mean(M(:,2).*M(:,3))
Ryhat2 = mean(M(:,1).*M(:,3))
```

Compare these estimates with the time averaging estimates found in Example 7.

Example 11. Let W be the process in Example 8. The following code uses $N = 5$ and 10000 realizations to obtain space averaging estimates (7) of $R_W(\tau)$ for $\tau = 0, 1, 2, 3, 4$:

```
clear;
M=zeros(10000,5); %each row will store W1,W2,W3,W4,W5 from a realization
for i=1:10000
n=1:5;
w=cos(n+2*pi*rand(1,1));
M(i,1:5)=w; %the realization has been stored in M
end
Rwhat0 = mean(M(:,5).*M(:,5))
```

```
Rwhat1 = mean(M(:,4).*M(:,5))
Rwhat2 = mean(M(:,3).*M(:,5))
Rwhat3 = mean(M(:,2).*M(:,5))
Rwhat4 = mean(M(:,1).*M(:,5))
```

Compare these answers to the time averaging estimates in Example 8.

12.5 Exp 5: Linear Predictor Design

Let (X_n) be a WSS process. Then, the minimum mean-square first, second, and third order linear predictors for X_n take the form:

$$\begin{aligned}\hat{X}_n &= AX_{n-1} \\ \hat{X}_n &= BX_{n-1} + CX_{n-2} \\ \hat{X}_n &= DX_{n-1} + EX_{n-2} + FX_{n-3}\end{aligned}$$

The first order predictor coefficient A is given by the equation

$$A = R_X(1)/R_X(0). \tag{8}$$

The second order predictor coefficients B, C are obtained by solving the matrix equation:

$$\begin{bmatrix} R_X(0) & R_X(1) \\ R_X(1) & R_X(0) \end{bmatrix} \begin{bmatrix} B \\ C \end{bmatrix} = \begin{bmatrix} R_X(1) \\ R_X(2) \end{bmatrix} \tag{9}$$

The third order predictor coefficients D, E, F are obtained by solving the matrix equation:

$$\begin{bmatrix} R_X(0) & R_X(1) & R_X(2) \\ R_X(1) & R_X(0) & R_X(1) \\ R_X(2) & R_X(1) & R_X(0) \end{bmatrix} \begin{bmatrix} D \\ E \\ F \end{bmatrix} = \begin{bmatrix} R_X(1) \\ R_X(2) \\ R_X(3) \end{bmatrix} \tag{10}$$

Example 12. Let the WSS process X_n have autocorrelation function

$$R_X(\tau) = \begin{cases} 5, & \tau = 0 \\ -1, & \tau = \pm 1 \\ 2, & \tau = \pm 2 \\ 0, & \text{elsewhere} \end{cases}$$

Run the following Matlab script, which separately computes the predictor coefficients for the first, second, and third order predictor:

```
R=[5 -1 2 0];
%find first order predictor coefficient A
A = R(2)/R(1);
%find second order predictor coefficients B,C
Q2=inv(toeplitz(R(1:2)))*R(2:3)';
B=Q2(1), C=Q2(2);
%find third order predictor coefficients D,E,F
Q3=inv(toeplitz(R(1:3)))*R(2:4)';
D=Q3(1), E=Q3(2), F=Q3(3);
[A 0 0; B C 0; D E F]
```

Observe what you get on your computer screen from running the preceding script. You should see a 3×3 matrix. The first entry in the first row gives the first order predictor coefficient A. The first two entries in the second row give the second order predictor coefficients B, C. The third row gives the third order predictor coefficients D, E, F. This matrix gives a convenient way to express all three of these predictors. Next week, we will push this topic further by presenting the *Levinson recursion method*, which generates this same matrix all at once (therefore all three linear predictors are generated at the same time).

EE 3025 S2007 Recitation 12 Lab Form

Name and Student Number of Team Member 1:

Name and Student Number of Team Member 2:

Name and Student Number of Team Member 3:

****

Experiment 2 discusses both a "time averaging" method and a "space averaging" method for estimating the power generated by the realizations of a random process. Your lab report this week will involve the use of one or both of these methods.