14 Recitation 14

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.

- Periodogram Method to Estimate Power Spectrum
- Bartlett's Method to Estimate Power Spectrum
- Application to Stock Market Investment
- More on Single Server Queue
- Review of Bayes Method

14.1 Exp 1: Periodogram Method to Estimate Power Spectrum

Let (X_n) be a discrete-time ergodic WSS process whose power spectral density $S_X(f)$ is not known. In order to estimate $S_X(f)$, one can use samples of the process $x[1], x[2], \ldots, x[N]$ measured at times $n = 1$ through $n = N$ along a realization $x[n]$ of the X process, where N is large. There are quite a number of effective spectrum estimation procedures that can be based upon these N samples. We discuss the periodogram estimate in this first experiment. In Experiment 2, you will look at Bartlett's estimate of the power spectrum (which typically gives a better estimate than the periodogram does).

The periodogram estimate $\hat{S}_X(f)$ of $S_X(f)$ is given by the formula:

$$\hat{S}_X(f) \triangleq \frac{1}{N}\left|\sum_{k=1}^{N} x[k]e^{-jk2\pi f}\right|^2, \ -\infty < f < \infty \tag{1}$$

The periodogram estimate can be easily found using the MATLAB function "`fft`". Just form a vector $x = [x[1], x[2], \ldots, x[N]]$ consisting of the N samples of the process. Then the MATLAB operation

```
abs(fft(x)).^2/N
```

computes the right hand side of (1).

Example 1. Let (Z_n) be Gaussian white noise with unit variance. Let (X_n) be the process defined by filtering the white noise as follows:

$$X_n = (0.5)X_{n-1} + (0.5)Z_n \tag{2}$$

The power spectral density $S_X(f)$ of the X process was derived in Experiment 3 of Recitation 13. It is

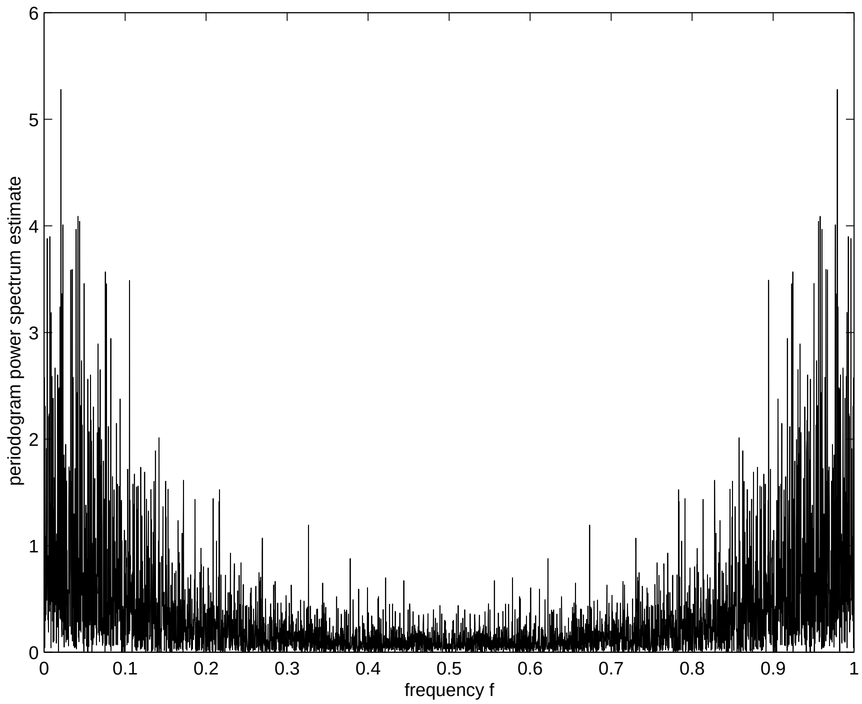
$$S_X(f) = \frac{1}{5 - 4\cos(2\pi f)}. \tag{3}$$

Suppose we do not know the filtering mechanism given by (2), and therefore we do not know the expression for $S_X(f)$ given in (3). Instead, we are simply handed a series of consecutive samples of process X_n, and must then estimate $S_X(f)$ based on these samples.

Step 1: In this step, we ran the following MATLAB script to generate the periodogram estimate of $S_X(f)$ based on the samples $x[1], x[2], \ldots, x[4096]$ from a realization $x[n]$ of (X_n):

```
N=4096;
z=randn(1,N);
x(1)=0;
for i=2:N
x(i)=.5*x(i-1)+.5*z(i);
end
periodogram = abs(fft(x)).^2/N;
freq = (0:N-1)/N;
plot(freq,periodogram)
axis([ 0, 1, 0, 6])
xlabel('frequency f')
ylabel('periodogram power spectrum estimate')
```

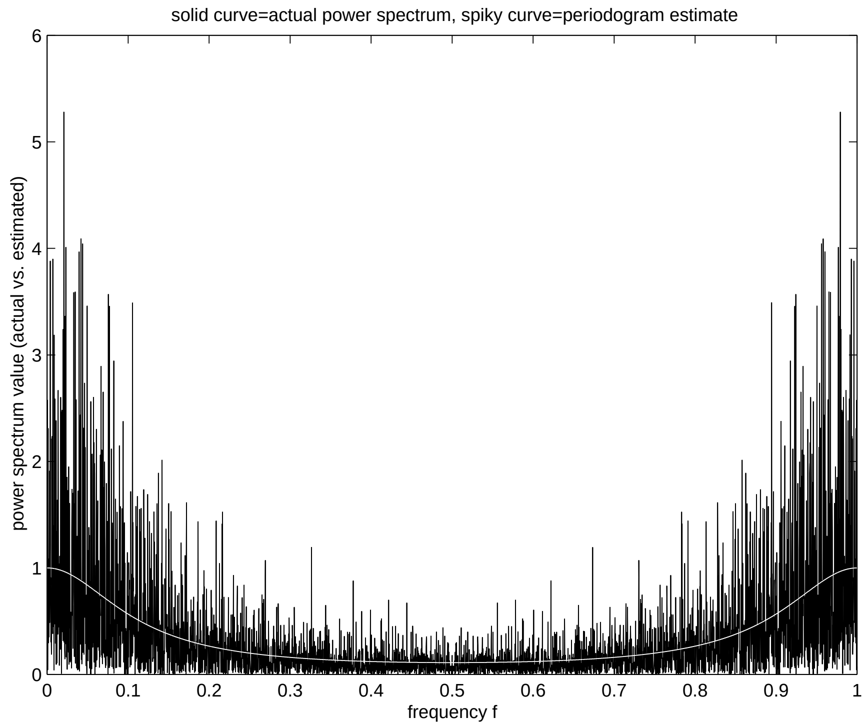
The following plot resulted:



Step 2: In this step, we ran the following script in order to obtain a plot of the actual $S_X(f)$ (3) and its periodogram estimate on the same set of coordinate axes:

```
PSD = (5-4*cos(2*pi*freq)).^(-1);
plot(freq,periodogram,freq,PSD)
axis([ 0, 1, 0, 6])
xlabel('frequency f')
ylabel('power spectrum value (actual vs. estimated)')
title('solid curve=actual power spectrum, spiky curve=periodogram estimate')
```

The following plot resulted:



The periodogram looks "spiky"; as a consequence, the periodogram provides a poor estimate of $S_X(f)$ in certain frequency ranges. The Bartlett estimate in Experiment 2 "smooths out" the spikiness in the periodogram estimate in a clever way, thereby providing better estimation of the power spectrum.

14.2 Exp 2: Bartlett's Method to Estimate Power Spectrum

Let (X_n) be a DT ergodic WSS process. In this experiment, you will try two different ways to get an estimate $\hat{S}_X(f)$ of the PSD $S_X(f)$ of the X process which will hopefully improve upon the periodogram estimate obtained in Experiment 1.

(i)Space-Averaging Method: For some large positive integer N, you average up N periodograms, each periodogram computed from a different realization of process X.

(ii)Bartlett's Method: For some large positive integer N, you average up N periodograms computed from disjoint segments of the same realization of process X.

Here are more details concerning Bartlett's method. Given `N` consecutive samples of a realization of the X process, Bartlett's method partitions these `N` samples into `N2` segments, each segment consisting of `N1` consecutive samples (of course, `N=N1*N2` must hold); a periodogram for each segment is computed, and then the `N2` periodograms are averaged to get Bartlett's PSD estimate.

In the examples which follow, to see how good the power spectrum estimates are, you will do *scatter plots* of them versus the actual PSD plot.

- *Example 2.* Let (Z_n) be Gaussian white noise with unit variance. Let (X_n) be the process obtained by filtering the white noise as follows:

$$X_n = (0.5)X_{n-1} + (0.5)Z_n$$

In this example, you use the space-averaging method. The estimate of $S_X(f)$ will be obtained by averaging up 32 periodograms from 32 different realizations of the X process. (Each periodogram is computed from 256 samples.) Run the following Matlab code, which plots the resulting PSD estimate as a scatter plot on the same set of axes as the actual PSD $S_X(f)$:

```
clear
N=256;
s=zeros(1,N);
for j=1:32
z=randn(1,N);
x(1)=0;
for i=2:N
x(i)=.5*x(i-1)+.5*z(i);
end
periodogram=abs(fft(x)).^2/N;
s=s+periodogram;
end
SXhat=s/32;
freq=(0:N-1)/N;
SX = (5-4*cos(2*pi*freq)).^(-1);
subplot(2,1,1)
plot(freq,SXhat,'+',freq,SX,'*')
title('Plot of SX(f) and its space-averaging estimate')
```

Examine your plot. Does the space-averaging estimate given by the scatter plot seem to be fairly close to the actual $S_X(f)$?

- *Example 3.* Let X be the same process used in Example 2. In this example, you use Bartlett's method. You average up 32 periodograms, each periodogram computed from 256 points on the *same* realization. Your goal is to see whether you get comparable (or better) performance than in Example 2. Run the code:

```
clear
N1=256;
N2=32;
N=N1*N2;
z=randn(1,N);
x(1)=0;
for i=2:N
x(i)=.5*x(i-1)+.5*z(i);
```

```
end
s=zeros(1,N1);
for j=1:N2
segment=x((j-1)*N1+1:j*N1);
periodogram=abs(fft(segment)).^2/N1;
s=s+periodogram;
end
SXhat=s/N2;
t=0:N1-1;
freq=t/N1;
SX = (5-4*cos(2*pi*freq)).^(-1);
subplot(2,1,2)
plot(freq,SXhat,'+',freq,SX,'*')
title('Plot of SX(f) and its Bartlett estimate')
```

Compare your plot with the plot in Example 2. Do the two scatter plots seem to give comparable estimates? The two estimation methods use the same number of points to form their estimates, and give comparable performance. The difference between the two methods resides in the fact that Bartlett's method uses just one realization. Therefore, Bartlett's method is the superior of the two methods.

14.3 Exp 3: Application to Stock Market Investment

Let X_n be the price of a stock (in dollars per share) on day n. We suppose that the X_n's are independent, identically distributed random variables. At the beginning of each day, the investor invests in this stock and in this stock only as follows:

(i) The investor sells all of his shares of the stock and adds the proceeds to his capital.

(ii) The investor invests $100p\%$ of his capital in the stock. (p is a fixed parameter that is kept fixed from day to day.)

If the investor's initial capital is one dollar, then his/her capital C_n after n days of investment (i.e., at the beginning of day $n+1$) is given by the formula:

$$C_n = \prod_{i=1}^{n} \left(\frac{pX_{i+1}}{X_i} + 1 - p \right)$$

If n is large, then with probability close to 1,

$$C_n \approx \exp(n\phi(p)),$$

where

$$\phi(p) = E[\log_e \left(\frac{pX_2}{X_1} + 1 - p \right)] \tag{4}$$

The best choice of p is the one for which $\phi(p)$ is a maximum, which, setting equal to zero the derivative of the right side of (4), yields:

$$E\left[\frac{X_2 - X_1}{pX_2 + (1-p)X_1}\right] = 0 \tag{5}$$

Let the "Louis Rukeyser strategy" be the best investment strategy which uses the choice of p satisfying equation (5). In this experiment, you simulate the return on your capital from investment using the Louis Rukeyser strategy as compared to the return obtained from more simple-minded strategies. For simplicity, we take the stock price X_n on day n to be either $1, 2$, or 3 dollars (equidistributed).

Example 4. You will model stock prices for n consecutive days as:

```
x = ceil(3*rand(1,n));
```

Let the initial capital be 1 dollar. You will see what your final return will be over 100 consecutive days. In this example, you test the "let it ride" strategy in which the re-invested fraction of day-to-day capital is close to one. Run the script:

```
p=.99; %Re-investment fraction of daily capital
C(1)=1; % initial capital
for j=1:100
x = ceil(3*rand(1,101));
for i=1:100
C(i+1)=C(i)*(p*(x(i+1)/x(i))+1-p);
end
capital(j)=C(101);
end
mean(capital)
```

You have estimated the return on your capital over a 100 day period, averaged over 100 runs. Do you get something on the order of \$1.50 or \$1.60 for the return on your investment? If so, you have earned about $50-60$ cents over the 100 days. (Remember: you only invested one dollar!)

Example 5. You now test the "play it safe" strategy in which the re-investment fraction is taken to be close to zero:

```
p=.01; %Re-investment fraction of daily capital
C(1)=1; % initial capital
for j=1:100
x = ceil(3*rand(1,101));
for i=1:100
C(i+1)=C(i)*(p*(x(i+1)/x(i))+1-p);
end
capital(j)=C(101);
end
mean(capital)
```

Is your return on the order of \$1.25? If so, you have earned about 25 cents over the 100 days.

Example 6. In this example, you test the effect of using the Louis Rukeyser investment strategy. First, you verify that $p = 1/2$ is the best re-investment fraction of capital, by verifying that it satisfies equation (5):

```
p=1/2;
n=10001;
x=ceil(3*rand(1,n));
y=x(2:length(x))-x(1:length(x)-1);
x=x(1:length(x)-1);
mean(y./(p*y+x))
```

Did you get nearly zero?

Example 7. Run the following code, to test the return you get from the Louis Rukeyser investment strategy:

```
p=.5; %Re-investment fraction of daily capital
C(1)=1; % initial capital
for j=1:100
x = ceil(3*rand(1,101));
for i=1:100
C(i+1)=C(i)*(p*(x(i+1)/x(i))+1-p);
end
capital(j)=C(101);
end
mean(capital)
```

Are you surprised by your result? This just goes to show you what an accurate model of the stock market could do for investors, potentially.[1]

14.4 Exp 4: More on Single Server Queue

In Recitation 13, we showed you how to simulate a single server queue with arrival rate λ and service rate μ. You learned that such a queue is stable if and only if $\mu > \lambda$. For a stable queue, you did a simulation to verify that the length of the queue does not blow up with time. For an unstable queue, you did a simulation to verify that the length of the queue does blow up with time. Instead of looking at the behavior of the length of the queue as time goes to infinity, the present experiment examines the behavior of the waiting time of the i-th arriving packet as $i \to \infty$. Specifically, you will do the following:

- For a stable queue ($\mu > \lambda$), you investigate the behavior of the waiting time of the i-th arriving packet as $i \to \infty$. In this case, the expected waiting time of the i-th packet converges to a finite limit as $i \to \infty$, and you do simulation to verify a theoretical formula that tells us what this limit is.

[1] Of course, the IID pricing model we used is unrealistic. A more complicated pricing model would be used in practice.

- For a unstable queue ($\mu \leq \lambda$), you investigate the behavior of the waiting time of the i-th arriving packet as $i \to \infty$. In this case, the expected waiting time of the i-th packet blows up as $i \to \infty$, and you do a simulation to verify this.

The purpose of the following Matlab examples is to provide elucidation of asymptotic properties of single server queues discussed in Section 42.2 of Lecture Notes 42.

Example 8. In this example, we let the arrival rate be $\lambda = 1$ and the service rate be $\mu = 2$. This will be a stable queue. Let W_i be the waiting time of the i-th arriving packet. We expect to see $E[W_i]$ leveling off as $i \to \infty$. Run the following Matlab script, which simulates the waiting times of each of the first 100 arriving packets:

```
lambda=1;
mu=2;
w(1)=0;
for i=2:100;
w(i)=max(0,log(rand(1,1))/lambda-log(rand(1,1))/mu+w(i-1));
end
```

Execute the line of code `w(1:15)`. You will see the waiting times for each of the first 15 packets printed out on your computer screen.

Now run the following Matlab script to generate the waiting times of the first 20000 packets:

```
n=20000;
lambda=1;
mu=2;
w(1)=0;
for i=2:n
w(i)=max(0,log(rand(1,1))/lambda-log(rand(1,1))/mu+w(i-1));
end
t=1:n;
plot(t,cumsum(w)./t)
xlabel('number of packets')
ylabel('average waiting time')
```

What you see is the plot of each i versus the average waiting time for packets 1 through i, for $i = 1, 2, \cdots, 20000$. Do these average waiting times appear to be "settling down" as the number of packets gets large? In the optional section of notes to be posted next week, it will be shown that

$$E[W_i] \approx \frac{\lambda}{\mu(\mu - \lambda)}, \quad i \text{ large.} \tag{6}$$

For $\lambda = 1$ and $\mu = 2$, compute

$$\frac{\lambda}{\mu(\mu - \lambda)} \tag{7}$$

and compare this theoretical value with the asymptotic average waiting time you see at the right end of your plot. Are these about the same? Re-run the preceding script a few times to see if the resulting plot's asymptotic average waiting time fluctuates closely about the value (7).

Example 9. In this example, you simulate the waiting times of packets for another stable single server queue, this time with $\mu = 2$ and $\lambda = 1.5$. Run the Matlab script:

```
n=20000;
lambda=1.5;
mu=2;
w(1)=0;
for i=2:n;
w(i)=max(0,log(rand(1,1))/lambda-log(rand(1,1))/mu+w(i-1));
end
t=1:n;
plot(t,cumsum(w)./t)
lambda/(mu*(mu-lambda))
```

Compare the asymptotic average waiting time you see at the right end of your plot with the number (7) computed for $\mu = 2$ and $\lambda = 1.5$. Do you get close agreement? Run your Matlab script again to be sure.

Example 10. Now we simulate what happens to the waiting times for an unstable queue. We will take $\mu = 1$ and $\lambda = 2$. Run the Matlab script

```
n=20000;
lambda=2;
mu=1;
w(1)=0;
for i=2:n;
w(i)=max(0,log(rand(1,1))/lambda-log(rand(1,1))/mu+w(i-1));
end
t=1:n;
plot(t,cumsum(w)./t)
```

Do the average waiting times appear to be growing linearly as the number of packets gets large? If so, this is the earmark of an unstable system. Run the script at least one more time to be sure that this behavior keeps occuring. (There is a theory giving the slope of this asymptotic straight line curve as a function of μ and λ, which one can read about in any good textbook on queueing systems.)

14.5 Exp 5: Review of Bayes Method

Bayes Method will be one of the review topics for the final exam. The purpose of this experiment is to remind you how to implement the different steps of Bayes Method in Matlab.

Let X, Y be discrete random variables. It is helpful to view X as the input to a channel and to view Y as the corresponding output from the channel. We suppose that there are N_x values of X and N_y values of Y, that the values of X have been ordered in some way, and that the values of Y have been ordered in some way. We let PX, PY, PXY, PY_X, PX_Y denote the matrices given below.

- PX = the vector of channel input probabilities. This means that PX is the N_x-dimensional row vector whose i-th component is $P\{X = x_i\}$, where x_i is the i-th value of X in the ordering of the values of X.
- PY = the vector of channel output probabilities. This means that PY is the N_y-dimensional row vector whose j-th component is $P\{Y = y_j\}$, where y_j is the j-th value of Y.
- PXY = the matrix of joint input-output probabilities. This means that PXY is the $N_x \times N_y$ matrix such that the element in row i and column j is $P\{X = x_i, Y = y_j\}$.
- PY_X = the channel matrix. This means that PY_X is the $N_x \times N_y$ matrix such that the element in row i and column j is $P\{Y = y_j | X = x_i\}$.
- PX_Y = the matrix of posterior probabilities. This means that PX_Y is the $N_x \times N_y$ matrix such that element in row i and column j is $P\{X = x_i | Y = y_j\}$.

14.5.1 Computing PY From PX and PY_X

The following MATLAB command will do this:

```
PY = PX*PY_X
```

Example 11. Let the vector of input probabilities and the channel matrix be given by:

$$\texttt{PX} = [1/3, 1/3, 1/3]$$

$$\texttt{PY\_X} = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 1/2 & 0 \\ 1/4 & 1/4 & 1/2 \end{bmatrix}$$

Then PY is computed by the three line MATLAB program

```
PX = [1/3 1/3 1/3];
PY_X = [1/3 1/3 1/3; 1/2 1/2 0; 1/4 1/4 1/2];
PY = PX*PY_X

PY =

    0.3611    0.3611    0.2778
```

14.5.2 Computing PX and PY From PXY

The following two MATLAB commands will do this:

```
PX = sum(PXY')
PY = sum(PXY)
```

Example 12. Let the matrix of input-output probabilities by given by

$$\texttt{PXY} = \begin{bmatrix} .1 & .2 & .05 \\ 0 & .1 & .2 \\ .05 & .2 & .1 \end{bmatrix}$$

Then PX and PY are computed by the following MATLAB program

```
PXY = [.1 .2 .05; 0 .1 .2; .05 .2 .1];
PX = sum(PXY')
PY = sum(PXY)

PX =

    0.3500    0.3000    0.3500


PY =

    0.1500    0.5000    0.3500
```

14.5.3 Computing PXY From PX and PY_X

The following MATLAB command will do this:

```
PXY = diag(PX)*PY_X
```

Example 13. Let PX and PY_X be as given in Example 11. Then the following MATLAB program computes PXY.

```
PX = [1/3 1/3 1/3];
PY_X = [1/3 1/3 1/3; 1/2 1/2 0; 1/4 1/4 1/2];
PXY = diag(PX)*PY_X

PXY =

    0.1111    0.1111    0.1111
    0.1667    0.1667         0
    0.0833    0.0833    0.1667
```

14.5.4 Computing PY_X From PXY

The following MATLAB command will do this:

```
PY_X = PXY./(diag(sum(PXY'))*ones(size(PXY)))
```

Example 14. Let PXY be as given in Example 12. Then the following MATLAB program computes PY_X.

```
PXY = [.1 .2 .05; 0 .1 .2; .05 .2 .1];
PY_X = PXY./(diag(sum(PXY'))*ones(size(PXY)))

PY_X =

    0.2857    0.5714    0.1429
         0    0.3333    0.6667
    0.1429    0.5714    0.2857
```

14.5.5 Computing PX_Y From PXY

The following MATLAB command will do this:

```
PX_Y = PXY./(ones(size(PXY))*diag(sum(PXY)))
```

Example 15. Let PXY be as given in Example 12. Then the following MATLAB program computes PX_Y.

```
PXY = [.1 .2 .05; 0 .1 .2; .05 .2 .1];
PX_Y = PXY./(ones(size(PXY))*diag(sum(PXY)))

PX_Y =

    0.6667    0.4000    0.1429
         0    0.2000    0.5714
    0.3333    0.4000    0.2857
```

Final Remark. Bayes Method is used to perform the following two tasks:

- Given PX and PY_X, compute PY.
- Given PX and PY_X, compute PX_Y.

The first task is accomplished according to Section 14.5.1 and the second task is accomplished according to Sections 14.5.3 and 14.5.5.

EE 3025 S2007 Recitation 14 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:

****

Study Experiment 3 on stock market investment carefully. I will have you do something with this on the lab reports. For more about this, read Section 42.6 of the Lecture 42 Notes.