ECE-597: Probability, Random Processes, and Estimation

Computer Project #6a

Due: Friday May 6, 2016

In this project we will modify your code from project six a bit and go in the opposite direction: If we know a signal follows an ARMA model and the model (transfer function) is *invertible*, then we can run the signal through the inverse transfer function to get white noise.

The transfer function for an ARMA model can be written in the form

$$\frac{\mathbf{X}(z)}{\mathbf{V}(z)} = \frac{B(z)}{A(z)}$$

If we want to get a white noise sequence $\mathbf{Y}(z)$ from the ARMA sequence $\mathbf{X}(z)$ that follows this model, then we want to use the transfer function

$$\frac{\mathbf{Y}(z)}{\mathbf{X}(z)} = \frac{A(z)}{B(z)}$$

We can use the Matlab function **filter** to implement the filter.

For the remainder of this project you are to repeat project six, but after you have generated the ARMA models and the power spectral density you are to do the following:

- Filter the sequence $\mathbf{X}[n]$ to get a new (white noise) sequence $\mathbf{Y}[n]$. The A and B vectors are generated in the code for you.
- Estimate the autocorrelation for the $\mathbf{Y}[n]$ sequence for the first 20 lags.
- Plot the poser spectral density of the sequence $\mathbf{Y}[n]$ using the Welch method. Using the same scaling you used for the psd estimate for $\mathbf{X}[n]$. Plot both the autocorrelation and the psd estimates for $\mathbf{Y}[n]$ on the same page using the subplot command.

What follows are the original instructions for Project 6.

At the top of your code set the sample time Ts and create a time array, as follows:

```
N = 10000;
Ts = 1/N;
t = [0:Ts:(N-1)*Ts];
```

Modify your ARMA discrete-time sequence so it allows a fourth order AR process (incude more a_i)

$$x[n] = a_1x[n-1] + a_2x[n-2] + a_3x[n-3] + a_4x[n-4] + b_0v[n] + b_1v[n-1]$$

where $v[n]$ is a white noise process (i.i.d) with zero mean and known variance σ_v^2.

We will continue to estimate the autocorrelation with *lag* r of the random sequence $x[n]$ as

$$R_{xx}(r) \;=\; \frac{1}{N-r}\sum_{n=1}^{n=N-r} x[n]x[n+r], \quad r=0,1,...,M$$

However, you will change the lag r.

When you plot the results, you should plot the random sequence as though it is a continuous function of the sample (don't try to plot as discrete points) and plot the time sequence as a function of time (not sample number). You should plot the autocorrelation estimate using the stem command (as discrete points) since we will not be looking at many lags.

We will use three different methods for estimating the psd.

- The first method first estimates the coefficients in the ARMA model and then determines the frequency response. Note that this is some Matlab magic and does not include and estimate of the noise variance. *Be sure that the estimates of the coefficients in the ARMA model are reasonable, however, the signs may be the opposite of what you expect for the a's since the A vector is printed out..* You will need to change the parameters *na* and *nc* in the Matlab code I gave you (follow the brilliant instructions.) Note that this method works well if the random sequence is well modeled as an ARMA process, otherwise it works poorly.
- The second method uses the Welch method based on computing the frequency response of overlapping segments and averaging them.
- The third method uses the frequency response of the autocorrelation function.

Notice that since we are using three different methods of estimating the psd, we are going to get three different results. You will note in the code that there is an attempt to scale the psd estimates.

For the following problems, assume we initially want $N = 1000$ sample points in the random sequence, we will initially use a maximum lag of $r = 20$, and that all necessary initial conditions are set equal to zero (the easiest way to do this is just set $X = zeros(1, N)$).

1) Implement the AR sequence $x[n] = -0.9025x[n-2] + v[n]$ assuming $\sigma_v^2 = 1$ and $v[n]$ is a Gaussian random variable. Plot the autocorrelation estimate for the first 20 lags and plot the estimates of the psd. You will need to change the parameters *na* and *nc* in the Matlab code I gave you. Run your code twice and turn in all plots. Note that each time we run the code we are getting a different *realization* of the noise sequence $v[n]$. You should also try it using the first 100 lags instead of just the first 20 lags.

2) Implement the MA sequence $x[n] = v[n] - 0.5v[n-1]$ assuming $\sigma_v^2 = 1$ and $v[n]$ is a Gaussian random variable. Plot the autocorrelation estimate for the first 20 lags. Run your code twice and turn in all plots. You will need to change the parameters *na* and *nc* in the

Matlab code I gave you. Note that each time we run the code we are getting a different *realization* of the noise sequence $v[n]$. You should also try it using the first 100 lags instead of just the first 20 lags.

3) Implement the ARMA sequence $x[n] = 0.8x[n-1] + v[n] - 0.5v[n-1]$ assuming $\sigma_v^2 = 1$ and $v[n]$ is a Gaussian random variable. Plot the autocorrelation estimate for the first 20 lags. Run your code twice and turn in all plots. You will need to change the parameters *na* and *nc* in the Matlab code I gave you. Note that each time we run the code we are getting a different *realization* of the noise sequence $v[n]$. You should also try it using the first 100 lags instead of just the first 20 lags.

4) Implement the AR sequence

$$x[n] = 2.760x[n-1] - 3.809x[n-2] + 2.654X[n-3] - 0.924X[n-4] + v[n]$$

assuming $\sigma_v^2 = 1$ and $v[n]$ is a Gaussian random variable. Plot the autocorrelation estimate for the first 20 lags. Run your code twice and turn in all plots. You will need to change the parameters *na* and *nc* in the Matlab code I gave you. Note that each time we run the code we are getting a different *realization* of the noise sequence $v[n]$. You should also try it using the first 100 lags instead of just the first 20 lags.

Your write-up should be short and neat, and you should include a copy of your code. I should be able to figure out which psd estimate goes with which sequence, and how many lags were used.