Boston University

Department of Electrical and Computer Engineering

EC505 STOCHASTIC PROCESSES

Problem Set No. 5

Fall 2016

Issued: Tuesday, Oct. 18, 2016 **Due:** Wednesday, 26, 2016

Problem 5.1 (Shanmugan and Breipohl 4.6)

Suppose the process $X(n)$ is a stationary, zero mean, discrete-time white Gaussian noise with $R_{XX}(k) = \delta(k)$. The process $Y(n)$ is obtained as the output of a discrete-time linear time invariant moving average system driven by $X(n)$:

$$Y(n) = h(0)X(n) + h(1)X(n-1) + \cdots + h(k)X(n-k).$$

(a) Find $p_{Y(n)}(y)$, the first-order density of the output.

(b) Find $R_{YY}(k)$.

(c) Is the output wide-sense and/or strict-sense stationary?

Problem 5.2 (Shanmugan and Breipohl 4.7)

Consider the following difference equation with initial condition $X(0) = 1$ and $U(n)$, $n \geq 0$ a sequence of zero mean, uncorrelated Gaussian random variables with variance σ_U^2.

$$X(n+1) = \sqrt{n}X(n) + U(n), \quad n = 0, 1, 2, \ldots$$

(a) Find $m_x(n)$.

(b) Find $R_{XX}(0,k)$, $R_{XX}(1,1)$, $R_{XX}(1,2)$, and $R_{XX}(3,1)$.

(c) Is the output process $X(n)$ wide-sense and/or strict-sense stationary?

Problem 5.3 (Shanmugan and Breipohl 4.14)

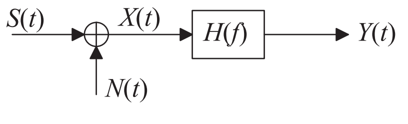
Assume that the input to a linear time-invariant system with impulse response $h(t) = e^{-t}u(t)$ is a zero-mean Gaussian random process with power spectral density $S_{XX}(f) = \eta/2$.

(a) Find the power spectral density $S_{YY}(f)$ of the output $Y(t)$.

(b) Find the average power in the output $E[Y^2(t)]$.

Problem 5.4 (Shanmugan and Breipohl 4.20)

In this problem we study filter design for a stochastic signal in noise. Consider the system shown in the figure below, where the input is given by $S(t) = a\sin(2\pi f_c t + \Theta)$ with a, f_c real constants and Θ uniformly distributed in the interval $[-\pi, \pi]$. The process $N(t)$ is white Gaussian noise with $S_{NN}(f) = \eta/2$ which is independent of Θ.



$$H(f) = \frac{1}{1 + j\left(\frac{f}{f_0}\right)}$$

(a) Let $Y(t)$ be the total output due to $X(t)$, $Y_1(t)$ be the output due to the signal $S(t)$ alone, and $Y_2(t)$ be the output due to the noise $N(t)$ alone. Since the system is linear $Y(t) = Y_1(t) + Y_2(t)$. Find $S_{Y_1Y_1}(f)$, $S_{Y_2Y_2}(f)$, and $S_{YY}(f)$.

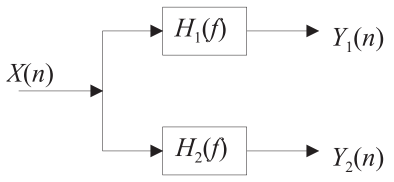
(b) Find the ratio of total average power in the signal component of the output $Y_1(t)$ to the total average power in the noise component of the output $Y_2(t)$. This is the output signal-to-noise ratio (SNR).

(c) What value of the filter bandwidth f_0 will maximize the output SNR you found in part (b).

Problem 5.5

Let $X[n]$ be a real-valued, discrete-time, zero-mean wide-sense stationary random process with correlation function $R_{XX}(m)$ and spectrum $S_{XX}(f)$.

(a) Suppose $X(n)$ is the input to two real-valued linear time-invariant systems as depicted below, producing two new processes, $Y_1(n)$ and $Y_2(n)$. Find $R_{Y_1Y_2}(n)$ and $S_{Y_1Y_2}(f)$ in terms of $h_1(n)$, $h_2(n)$, $R_{XX}(n)$, $H_1(f)$, $H_2(f)$, and $S_{XX}(f)$.



(b) Suppose next that $H_1(f)$ and $H_2(f)$ are non-overlapping frequency responses, i.e.,

$$|H_1(f)| \cdot |H_2(f)| = 0 \ , \qquad \text{all } f \ .$$

Show that in this case $Y_1(n)$ and $Y_2(m)$ are uncorrelated for all n and m. Are $Y_1(n)$ and $Y_2(m)$ statistically independent (for all n and m) in the case that $x(n)$ is a Gaussian random process? Explain.

Problem 5.6

Assume that a shaping filter $H(f)$, when driven by white noise $W(t)$ with autocorrelation $R_{WW}(\tau) = \delta(\tau)$, produces a wide-sense stationary output with power spectral density

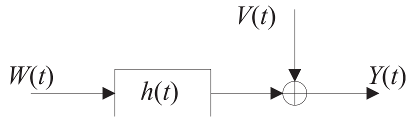
$$S_{YY}(f) = \frac{1 + (2\pi f)^2}{(2\pi f)^4 + 12(2\pi f)^2 + 36}.$$

(a) Find the variance of the output $E[Y^2(t)]$.

(b) Find the transfer function of the shaping filter $H(f)$.

Problem 5.7 (Old Exam Question)

Let $W(t)$ and $V(t)$ be independent stationary zero-mean white Gaussian random processes with correlation functions $R_{WW}(\tau) = \sigma_W^2\delta(\tau)$ and $R_{VV}(\tau) = \sigma_V^2\delta(\tau)$. A new signal $Y(t)$ is generated from $W(t)$ and $V(t)$ via the processing shown in the figure, with $h(t)$ stable and causal and $H(f) = \dfrac{1}{\alpha + j2\pi f}$:



(a) Find $R_{YY}(\tau)$ and $S_{YY}(f)$ (you need not simplify $S_{YY}(f)$).

(b) We want to find a simpler system whose output random process has the same statistical properties as $Y(t)$. Let $E(t)$ be a stationary zero-mean white Gaussian random process with covariance function $R_{EE}(\tau) = \delta(\tau)$. Find the $\underline{\text{impulse response}}$ $h'(t)$ of a linear time invariant system such that $Z(t) = h'(t) * E(t)$ has the same covariance function as $Y(t)$. Use $\alpha = 2$, $\sigma_W^2 = 5$, and $\sigma_V^2 = 1$. Hint: Consider factoring $S_{YY}(f)$.

(c) Is the answer to part (b) unique? Explain.

(d) Do the random processes $Y(t)$ and $Z(t)$ have the same pdfs, i.e. for all $t_0, \ldots, t_n$ and all $a_0, \ldots, a_n$ does: $p_{Y(t_0),Y(t_1),\ldots,Y(t_n)}(a_0, a_1, \ldots a_n) = p_{Z(t_0),Z(t_1),\ldots,Z(t_n)}(a_0, a_1, \ldots a_n)$?

Problem 5.8 A lab test is carried out to determine the model for a spring. The experiment include a position, $X[n]$, and force measuring transducer that measures the force $F[n]$. Data is collected for compression and expansion of the spring. We model the system using a second order ARMA model:

$$X[n] + a_1 X[n-1] + a_2 X[n-2] = F[n]$$

A reasonable assumption is that the noise $W[n]$ is independent of the Force $F[n]$. Setup a linear system of equations to identify the coefficients, a_1, a_2.
(**Bonus:**) You may want to download spring_force.mat from HW5 folder and try to identify a model. See whether you can get a good fit.

Computer Problems

Problem 5.9 Linear Systems Driven by White Noise

In this project we examine the output of a number of systems driven by white noise. The systems will be discrete-time rational systems specified in terms of the vectors of their numerator and denominator coefficients, `b` and `a`. We will use the MATLAB definition of such systems, which differs from that used in the notes – in particular, the non-unity coefficients differ by a sign. MATLAB uses the following system definition:

$$H(e^{j\omega}) = \frac{B(z)}{A(z)} = \frac{b_1 + b_2 z^{-1} + \cdots + b_{N_z+1} z^{-N_z}}{1 + a_2 z^{-1} + \cdots + a_{N_p+1} z^{-N_p}} \tag{1}$$

Thus the a_i coefficients define the poles and the b_j coefficients define the zeros and gain. The numerator coefficients are placed in the vector `b` and the denominator coefficients in the vector `a`. Let us define 5 systems, specified by the following sets of coefficients:

System #0:	`b0 = [5, 0]`	`a0 = [1, -1.3, 0.845]`
System #1:	`b1 = [1, 1.3, 0.845]`	`a1 = [1, -1.3, 0.845]`
System #2:	`b2 = [0.3, 0]`	`a2 = [1, -0.8]`
System #3:	`b3 = [0.06, 0.12, 0.06]`	`a3 = [1, -1.3, 0.845]`
System #4:	`b4 = [0.845, -1.3, 1]`	`a4 = [1, -1.3, 0.845]`

System #0 is second-order AR, System #2 is first-order AR, while Systems #1 and #3 are 2nd order ARMA, and System #4 is an all pass. We will drive each of these systems by a white noise process $U(n)$ to obtain an output $Y(n)$, as depicted in the figure below.

$$\begin{tikzcd} U(n) \arrow[r] & \boxed{h(n)} \arrow[r] & Y(n) \end{tikzcd}$$

(a) We will need to estimate autocorrelation functions from data in our study of linear systems and random processes. As we learned in class, if a process is ergodic, we can estimate its autocorrelation function from a single sample path by forming the following time average:

$$\widehat{R}_{XX}(k) = \frac{1}{L} \sum_{n=0}^{L-1} x(n)x(n+k) \tag{2}$$

If the signal is finite, of length N, then we must have $L \leq N - k$. Note in particular, that this means that we must average fewer points in estimating large values of lag k. Thus, given a finite signal, our estimate $\widehat{R}_{XX}(k)$ is more reliable for small values of k. The estimate of $R_{XX}(k)$ in (2) based on the data in the vector `x` can be computed in MATLAB as follows:

```
[Rest,lags] = xcorr(x,maxlag,'unbiased');
```

where `maxlag` is the maximum lag to compute (i.e. maximum value of k in (2)) and the parameter `'unbiased'` tells matlab to divide the sum by L as defined in (2) (by default MATLAB only performs the summing operation). The resulting correlation function can be viewed using `plot(lags,Rest)`.

To test this procedure generate a long vector of unit variance discrete-time white Gaussian noise `u` as follows:

```
N = 5000;
u = randn(1,N);
```

Now use the approach described above to estimate the correlation function for the Gaussian white noise vector `u`. Use a value of `maxlag=50`. What is the theoretical autocorrelation function $R_{UU}(k)$ of `u`? Plot both the theory and experiment together. Do they agree? We now have a way of estimating the autocorrelation function of a sequence.

(b) For each of the given systems find and plot together both the impulse response `h` and the pole/zero plot. This is easily done given the coefficient vectors `b` and `a` by using the MATLAB functions `impz.m` and `zplane.m`. Make your impulse responses 50 steps long (i.e. use $N = 50$ in `impz.m`). Save your impulse responses for part (d).

(c) Find the output `y` of each of the systems when driven by the Gaussian white noise process `u`. This can be simply done using the coefficient vectors for the systems together with the MATLAB function `filter.m`. For each of the systems, compute the pdf of the output sequence using `pdf1d.m` when the input is the Gaussian white noise sequence `u`.

(d) Using the output `y` you found in part (c), estimate the output autocorrelation function $R_{YY}(k)$ for each system, in the manner described in part (a) above. Use `maxlag=50`.

On the web site you will find the MATLAB function `syscor.m`, which calculates the *theoretical* autocorrelation function of the output of a discrete-time system when driven by white noise. Recall from class that for a system driven by white noise, this output is given by $R_{YY}(n) = h(n) * h(-n)$. Using this supplied function, calculate the theoretical autocorrelation functions of the output of each system with `maxlag = 50`.

For each system, plot your autocorrelation estimates together with the corresponding theoretical values. How sensitive are the autocorrelation estimates to data length (i.e. what happens if you use only part of the output data in `y`). What happens if you use uniformly distributed white noise instead of Gaussian white noise? Can you explain the results obtained for System #4? What happens to the estimate for larger lag values?

(e) One way to find the impulse response of an linear time-invariant (LTI) system is to drive it with an impulse and record the output. In this part, we will use a different approach based on the cross-correlation function. Recall that the cross-correlation between the input $U(n)$ and output $Y(n)$ of an LTI system with impulse response $h(n)$ is given by:

$$R_{UY}(k) = R_{UU}(k) * h(k) \tag{3}$$

If the input sequence $U(n)$ is unit variance white noise such as we are using, so that $R_{UU}(k) = \delta(k)$, what is the cross-correlation function $R_{UY}(k)$? Use this insight to estimate the impulse response of each of the systems based on the cross-correlation between the white Gaussian noise input and the corresponding output. The cross-correlation between an input `u` and output `y` can be estimated in MATLAB as follows:

```
[Ruy,lag] = xcorr(u,y,maxlag,'unbiased');
```

Use `maxlag = 50`. Plot your estimated impulse responses together with the true impulse responses from part (a) and compare. Note that driving a real system with an impulse can be difficult, while driving it with white noise is often much easier.

Problem 5.10 Power Spectral Density Estimation via FFT

An important precursor to many signal processing problems is the estimation of the power spectrum of a process. In this project we will construct and investigate an FFT-based method for power spectral density estimation. This topic is discussed in more detail in the reserve texts Shanmugan and Breiphol, Section 9.3 or Therrien Section 10.1.

(a) You will write a function `psdest1.m` to estimate the power spectral density of a process given the sample path data in the vector `y`. The function will first estimate the autocorrelation function of the process and then will take an FFT of these values to obtain an estimate of the power spectral density function. The function call for your program will be the following: `[Syy,w] = psdest1(y,maxlag)`, where `y` is the input data vector, `maxlag` will be the number of lags to use in the correlation estimate, `Syy` will be the estimated power spectral density, and `w` will be the corresponding vector of frequency values. Each step below will form a line of the program:

(i) The first step is estimate the values of the autocorrelation function for lags between `-maxlag` and `+maxlag` as described previously:

```
[Ryy,lag] = xcorr(y,maxlag,'unbiased');
```

(ii) Given the estimated autocorrelation function in `Ryy`, we now estimate the power spectral density by taking its FFT:

```
Syy = abs(fft(Ryy));
Syy = Syy(1:maxlag+1);
```

There are two slightly subtle implementational details in this part. First, because MATLAB assumes the first element of a vector representing a time series occurs at time $n = 0$ and not the "center" of the vector, an undesired linear phase will be added to the transform of `Ryy`. This is removed by taking the magnitude of the FFT via `abs.m`. Since we know the transform of `Ryy` is real, this will have no effect on the true `Syy`. Second, the FFT program calculates the values of the spectrum from $\omega = 0$ to $\omega = 2\pi$, but we know the spectrum is symmetric. Thus we only keep those values corresponding to frequency samples in the range $[0, \pi]$, which are the first `maxlag+1` values.

(iii) Finally, we need to generate the vector of corresponding frequency points `w`. The FFT samples we generate correspond to `maxlag+1` uniformly spaced frequency values in the range $[0, \pi]$, thus:

```
w = linspace(0,pi,maxlag+1);
```

Add these steps together to create your program. Test your function by estimating and plotting the power spectral densities of the white noise process `u` vs frequency. You may wish to use `semilogy.m` for your plots.

(b) Use your function `psdest1.m` to estimate the power spectral densities of the outputs of the 5 systems for the white Gaussian noise input `u`. On the web site you will find the function `syspsd.m` which finds the *theoretical* power spectral density for the output of a discrete-time system driven by white noise. Recall from class that for a system with frequency response $H(e^{j\omega})$ driven by white noise this output PSD is given by $|H(e^{j\omega})|^2$. For each of the 5 systems, use this function to find the corresponding theoretical power spectral density and plot it along with your experimentally determined power spectral density. Use `maxlag = 100` and `Nf = 100`. Again, you may wish to use `semilogy.m` for your plots so you can see more detail. How do the estimated power spectral densities change as `maxlag` is made smaller or larger?

The parameter `maxlag` truncates the correlation function $R_{YY}(m)$ to the range `[-maxlag, +maxlag]`, and thus essentially defines a rectangular window which we apply in the "time" domain to the estimated correlation function before taking the Fourier transform. As a result, in the frequency domain it can be shown that the expected value of the estimated spectrum is the convolution of the Fourier transform of this rectangular window (i.e. a sinc function) with the true spectrum. Thus, shorter rectangular windows (smaller values of `maxlag`) introduce blurring and bias into the estimate. There can also be problems with oscillations due to the sidelobes of the sinc function in the frequency domain.

At the other extreme, as we let the rectangular window size `maxlag` become large and approach the length of the data N_y, it can be shown that the estimate produced by `psdest1.m` approaches $\widehat{S}_{YY}(\omega) = \frac{1}{N_y}|\mathcal{F}\{y(n)\}|^2$, where $\mathcal{F}[y(n)]$ is the discrete time Fourier transform of the data sequence. This estimate is known as the "periodogram" and it is unfortunately true that the variance of this estimate is as large as its mean! Thus, in choosing a value of `maxlag` we must trade off bias and resolution against variance. A general rule of thumb is to choose `maxlag` no larger than say 10% of N_y. Note that we have been using `maxlag` $= 1\%$ of N_y. Using the output of System #0, see what happens if you use a value of `maxlag` significantly larger than our rule. More on these issues can be found in the reserve texts Shanmugan and Breiphol Section 9.3 or Therrien Section 10.1.