The Cooper Union Department of Electrical Engineering
Prof. Fred L. Fontaine
ECE211 Signal Processing & Systems Analysis
Problem Set IX: Random Signals
April 27, 2019

Note that I am looking for good MATLAB programming style; for example avoid *for* loops when possible, using vectorization methods. [This doesn't mean *for* loops are prohibited, but avoid silly ones].
This assignment all relates to a real WSS signal $x(n)$ that is modeled as:

$$x(n) = v(n) + v(n-1) + 0.36v(n-2) + 0.6x(n-1) - 0.9x(n-2)$$

where $v(n)$ is 0-mean white noise with $\sigma_v^2 = 4$.

1. Before you start coding in MATLAB:
   (a) Is x AR, MA or ARMA?
   (b) Is this filter (with v as input and x as output) the whitening or the innovations filter?
   (c) Write a difference equation for the inverse filter.
   (d) Write an explicit formula by hand (not simplified) for the PSD $S_x(\omega)$ of x.
2. In MATLAB, draw the pole-zero plot for the filter (input v, output x).
3. In MATLAB, generate $N = 10000$ samples of v and then apply the filter to generate N samples of x.
4. Let $m_0 = 5$. In MATLAB, use time-averaging to estimate $r_x(m)$ for $0 \leq m \leq m_0$. Do this in a for loop indexed by m. Rather than presenting the detailed process for actually computing $r_x(m)$, we will use these estimated values for the remainder of the problem as the "true" values.
5. Consider a vector comprised of M consecutive samples of x, defined as:

$$x_M(n) = \begin{bmatrix} x(n) \\ \vdots \\ x(n-M+1) \end{bmatrix}$$

   For which M will the correlation matrix for $x_M(n)$ involve $r_x(m)$ for $|m| \leq m_0$? Generate this matrix in MATLAB using the values you found above (remember that $r(-m) = r(m)$ for real random signals!) **Hint:** Examine the function *toeplitz.*
6. Check that your correlation matrix is positive definite by computing its eigenvalues.

7. The MATLAB function *pwelch* can be used to estimate the power spectral density; the name of the method this function employs is called the *modified Welch periodogram.* Invoke *pwelch* as follows:

$$[s\_est, w] = pwelch(x, \text{hamming}(512), 256, 512);$$

where s_est is the estimated PSD and w is the frequency vector (in normalized digital radian units).

8. We want to compare the estimated spectrum, s_est, and the actual PSD $S_x(\omega)$ computed at the points in w, on a linear (not decibel) scale. There is an issue of normalization. A **simplified** explanation of what *pwelch* does is that it computes the magnitude-squared Fourier transform (DFT actually) of blocks of x and averages the results. But then factors such as the size of the blocks can effect the scaling. So for purposes of comparison here, normalize s_est and the actual $S_x(\omega)$ you compute so they each have AVERAGE value 1 (that is, if s is the vector of PSD values, apply the normalization $s = s/mean(s)$; this approximates the condition $\frac{1}{2\pi}\int_{-\pi}^{\pi} S(\omega)\, d\omega = 1$). Then superimpose graphs of them. The horizontal axis should extend from 0 to π (if you simply call *plot* MATLAB won't produce this– you will have to adjust the graph so it goes from 0 to π only).

9. If you look at the pole-zero plot of H, you will note there is a pole pair fairly close to the unit circle, that is in fact close to the frequency where $S_x(\omega)$ has a peak. Compare the angle of the poles of the filter H with the value ω where $S_x(\omega)$ has a peak. They won't exactly match, but they should be close.

10. Generate a data matrix A whose columns are comprised of sliding blocks of x, that is:

$$A = \begin{bmatrix} x(m_0+1) & x(m_0+2) & \cdots & x(N) \\ \vdots & \vdots & & \vdots \\ x(1) & x(2) & \cdots & x(N-m_0) \end{bmatrix}$$

[Hint: Use toeplitz again!] Note that A is $(m_0+1) \times (N-m_0)$. By stationarity, each column of A has the same statistical properties, and we can think of A as representing a collection of (noisy) measurements with underlying statistical properties we want to extract. Observe that each column, in fact, viewed as a stand-alone random vector, has the same correlation matrix R found above. Now, use MATLAB *svd* to compute the singular values of A, say $\sigma(A)$ (stored in a vector). Compare the vector of values $(\sigma(A))^2 / (N-m_0)$ to the eigenvalues of R. *[Remark: When comparing the vectors, the entries may occur in different orders; use the sort function to overcome this problem]* [Remark: As R is a matrix of second order moments, its eigenvalues represent power; the square of the singular values of the data matrix A similarly represent power, in a sense; the normalization can be thought of giving us an average power related to each column of A]. Also check that the left singular vectors of A are the eigenvectors of R as follows: take $\vec{u}$ to be one of the singular vectors; do elementwise division of $R\vec{u}$ over $\vec{u}$ and all the entries should be the same (ideally) equal to an eigenvalue of R. In fact, there may a couple cases with a mismatch but if you look closely the elements of $\vec{u}$ are close to 0 at those points. The moral of the story: the SVD of the data matrix encapsulates the statistical properties of the data. :-D