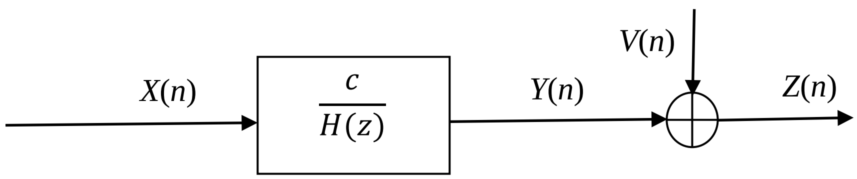
Department of Electrical Engineering, IIT Madras

EE3005: Communication Systems

10 Marks ***Computational Assignment*** **Due by: May 12, 5pm**

Instructions: Submit a soft-copy of your report in PDF. Latex / iPad / tablet / hand-written (and scanned) reports are fine. Your report should be named "ee3005-YourRollNumber-report.pdf." Along with the soft-copy of your report, your working code (Matlab is preferable) must also be submitted. Your properly commented working code should be named "ee3005-YourRollNumber-code.m." Python submissions are also okay. All students with roll numbers **ee22bxxx** must send report and code to **Mr. A. Gaurav** ee20b035@smail.iitm.ac.in; all other students (**ee21, ee20, ep**, etc) should email the same to **Ms. Prasikaa Shree** ee21d700@smail.iitm.ac.in. The TAs will get back to you if additional information is required. Please see additional instructions, if any, on the WhatsApp group.

An auto-regressive (AR) random process or time-series $Y(n)$ which is asymptotically WSS can be obtained by sending a WSS RP $X(n)$ through an appropriate IIR filter $c/H(z)$ as shown below:



Here, $X(n)$ is a discrete, white, Gaussian RP with variance σ_X^2 and $V(n)$ is independent of $X(n)$ and is also a Gaussian RP representing the measurement noise with variance σ_V^2. The third-order transfer function $H(z)$ is in turn given by $(1-\alpha e^{j\theta}z^{-1})(1-\alpha e^{-j\theta}z^{-1})(1-\beta z^{-1})$ where $\theta = \frac{\pi}{3}$ radians and two different choices for α and β are considered for our study.

Case 1: $\alpha = 0.92;\ \beta = 0.95;$

Case 2: $\alpha = 0.997;\ \beta = 0.999;$

For each of the above cases, chose the normalization constant c to ensure that $\frac{c^2}{\sum_{i=1}^{4}|h|_i^2} = 1$; i.e., normalize the filter coefficients $\{h_i\}$ to ensure that the filter has a gain of unity. This way, the signal to noise ratio (SNR) measured on the received samples $Z(n)$ is then given by σ_X^2/σ_V^2 . Also, by setting $\sigma_X^2 = 1$, the SNR in the dB scale is then given by $10log_{10}\left(\frac{1}{\sigma_V^2}\right)$ and it can be varied by varying σ_V^2 appropriately. The input-output relationship (after the gain normalization) can be written as

$$Y(n) = h_1 Y(n-1) + h_2 Y(n-2) + h_3 Y(n-3) + X(n) \tag{1}$$

where the numerical values of the filter coefficients $\{h_i\}$ are different from Case 1 and Case2. The aim of this assignment is to estimate these three values of $\{h_i\}$ as say $\{\hat{h}_i\}$ from the measurements using time-averaged values of the autocorrelation function (assuming that ergodicity holds).

The measurements at the receiver are then given by $Z(n) = Y(n) + V(n)$. Now, if we had access to the noise-free output $Y(n)$ as in (1), the linear equations can be set up as follows:

Since there are 3 unknowns $\{h_i\}$ in (1), we need 3 linear equations. Multiply both sides of (1) by $Y(n-k)$ and take expectations on both sides, for k=1,2, and 3. This gives the well-known Yule-Walker equations as below:

$$\begin{bmatrix} R(1) \\ R(2) \\ R(3) \end{bmatrix} = \begin{bmatrix} R(0) & R(1) & R(2) \\ R(1) & R(0) & R(1) \\ R(2) & R(1) & R(0) \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} \tag{2}$$

where $R(k) = E[Y(n)Y(n-k)]$. Now, we have access to only the $N_o + N$ noisy values $Z(n)$ and hence, using time-domain averaging assuming ergodicity, we find the entries $\hat{R}(k)$ in (2), as follows:

$$R(k) \approx \hat{R}(k) = \left(\frac{1}{N-k}\right) \sum_{i=N_o+k+1}^{N_o+N} Z(i)Z(i-k) \quad (3)$$

Here, N_o is an integer offset we give to enable the AR process $Y(n)$ to "forget" it's initial conditions and become WSS. In other words, you do not use the first N_o samples of $Y(n)$ while computing the auto-correlation estimates in (3). Observe that (2) in vector-matrix notation, can be re-written as

$$\boldsymbol{r} = \boldsymbol{R}\boldsymbol{h} \quad (4)$$

and the estimate of the 3 AR parameters are then given by

$$\hat{\boldsymbol{h}} = \boldsymbol{R}^{-1}\boldsymbol{r} \quad (5)$$

In each of the below experiments:

- You need to generate $N_o + N$ samples of $Y(n)$ using (1), <u>with initial conditions</u> $Y(0) = Y(-1) = 0$. Add noise with variance σ_V^2 to give $Z(n)$.
- Change the SNR in steps of 3dB, from 0dB to 21dB, in each experiment.
- For each SNR, run $j = 1$ to 500 (Monte-Carlo) trials. In the j^{th} trial, the estimation error-square is given by $e^2(j) = \sum_{i=1}^{3}\left(h_i - \hat{h}_i(j)\right)^2$ and the ergodic Mean Square Error in the dB scale, averaged over these 500 trials is then given by

$$MSE_{SNR} = 10log_{10}\left(\frac{1}{500}\sum_{j=1}^{500} e^2(j)\right) \quad (6)$$

For each of the above two choices of α and β (Case 1 and Case 2), answer the following 10 questions, which in total carry 10 marks:

1. For each SNR (in 3dB steps), for each (of the 500) trial, generate using $N_o = 0$ & $N = 100$, the samples of $X(n)$, $Y(n)$, $V(n)$, and finally $Z(n)$.
2. Use (3) to compute the entries in (2). Solve for $\boldsymbol{h}$ as in (5) and get the estimate $\hat{\boldsymbol{h}}$. For each trial, compute $e^2(j)$. Then, averaging over 500 such trials for this SNR, compute the MSE as in (6).
3. Tabulate your SNR versus MSE result for this choice of $N_o = 0$ & $N = 100$.
4. Repeat steps 1. to 3. for $N_o = 0$ & $N = 1000$.
5. Repeat steps 1. to 3. for $N_o = 0$ & $N = 20{,}000$.
6. Repeat steps 1. to 3. for $N_o = 500$ & $N = 1000$.
7. Repeat steps 1. to 3. for $N_o = 5000$ & $N = 20{,}000$.
8. Plot these 5 tables of results with SNR on the x-axis and MSE on the y-axis, both in dB scale. You should have 5 different curves. Label them clearly. Call this Figure-1 for Case-1.
9. Repeat the above steps for Case-2 and generate Figure-2 for Case-2.
10. Compare your results for Case-1 and Case-2 and comment.
11. Bonus Question #1: Can you improve your MSE results for Case-2 (i.e., aim for a lower MSE) by a different choice of N_o & N ? If so, indicate your choice(s) for N_o & N and present the new result and explain it. Clear, original work could fetch you <u>2 bonus marks</u>.
12. Bonus Question #2: The computational complexity of the matrix inverse in (5) is typically high. For a random LxL matrix, the complexity is Order (L^3). However, since $\boldsymbol{R}$ in (2) and (4) is a Toeplitz, Symmetric matrix, the Levison-Durbin algorithm can be used to reduce the complexity to Order (L^2). If you would like to find out about the Levinson-Durbin recursions, implement them, and present your results. This could fetch you <u>2 bonus marks</u>.