Amirkabir University of Technology
(Tehran Polytechnic)

Electrical Engineering Faculty

Control Department

MSc Program

Assignment 02

through

System Identification

Course

by

Mohammad Azimi - 402123100

mohammadazimi2000@aut.ac.ir

Course Lecturer

Dr. Mehdi Karrari

I.Question 1

Given the set of data for X and Y, the probability distribution function could be defined as:

x	2.72	4.53	2.75	1.94	4.60	4.23	2.77	1.49	2.56
y	0.84	1.28	0.74	1.44	1.39	-0.25	0.05	0.26	0.49

$$f(x) = \frac{1}{9} \quad ; \; x \in S_X \qquad E(X) = \mu_x = \sum xf(x) = 3.0656$$

$$f(y) = \frac{1}{9} \quad ; \; y \in S_Y \qquad E(Y) = \mu_y = \sum yf(y) = 0.6933$$

And the covariance matrix for the given space data derives:

$$Cov\ Matrix = \begin{bmatrix} COV(X.X) & COV(X.Y) \\ COV(Y.X) & COV(Y.Y) \end{bmatrix} = \begin{bmatrix} E(XX) - E(X)E(X) = \sigma_X^2 & E(XY) - E(X)E(Y) \\ E(YX) - E(Y)E(X) & E(YY) - E(Y)E(Y) = \sigma_Y^2 \end{bmatrix}$$

$$\Longrightarrow Cov\ Matrix = \begin{bmatrix} 1.2708 & COV(X.Y) \\ COV(Y.X) & 0.3679 \end{bmatrix}$$

$$\longrightarrow COV(X.Y) = COV(Y.X) = E\left[(X - \mu_x)(Y - \mu_y)\right] = 0.0999 \Longrightarrow CovMat\ (X.Y) = \begin{bmatrix} 1.2708 & 0.0999 \\ 0.0999 & 0.3679 \end{bmatrix}$$

II.Question 2

We already know the following properties for *mean* and *variance* concepts.

$$\left\{ \begin{matrix} E(aX + b) = aE(X) + b \\ Var(aX + b) = a^2 Var(X) \end{matrix} \right\} \longrightarrow \left\{ \begin{matrix} \mu_y = 5\mu_x + 3 \\ \sigma_y^2 = 25\sigma_x^2 \end{matrix} \right. \longrightarrow P_Y(y) = \frac{1}{\sqrt{2\pi 25\sigma^2}} e^{-\frac{(y - 5\mu_x - 3)}{2 \times 25\sigma^2}}$$

Hence, the desired function is obtained:

$$P_Y(y) = \frac{1}{\sqrt{2\pi 25\sigma^2}} e^{-\frac{(y - 5\mu_x - 3)}{2 \times 25\sigma^2}} = \frac{1}{5\sigma\sqrt{2\pi}} e^{-\frac{(y - 5\mu_x - 3)}{2 \times 25\sigma^2}}$$

III.Question 3

Take the following path step by step to prove the requested solution.

$$COV(X) = E[(X - \mu_x)(X - \mu_x)^T]$$

$$COV(Y) = E\left[(Y - \mu_y)(Y - \mu_y)^T\right]$$

Knowing that:

if $Y = AX + b$, then $\mu_y = A\mu_x + b$. Therefore:

$$COV(Y) = E[(AX + b - (A\mu_x + b))(AX + b - (A\mu_x + b))^T] = E[(AX - A\mu_x)(AX - A\mu_x)^T]$$

Using the transpose properties, we get:

$$COV(Y) = E[(AX - A\mu_x)(X^TA^T - \mu_x^TA^T)] = E[A(X-\mu_x)(X^T-\mu_x^T)A^T] = A[(X-\mu_x)(X-\mu_x)^T]A^T$$

Which proves that:

$$COV(Y) = ACov(X)A^T$$

IV.Question 4

❖ Part a

Being told to obtain the auto correlation function of $Y(t)$, accordingly we do:

$$R_Y(\tau) = E[y(t)y(t+\tau)] \xrightarrow{Y(t)=aX(t)+b} R_Y(t) = E[(ax(t)+b)(ax(t+\tau)+b)]$$

$$\Longrightarrow R_Y(\tau) = E[a^2x(t)x(t+\tau) + abx(t) + abx(t+\tau) + b^2]$$

$$= E[a^2x(t)x(t+\tau)] + abE[\big(x(t)) + x(t+\tau)\big) + E[b^2]$$

$$= a^2E[x(t)x(t+\tau)] + 2abm + b^2$$

$$\Longrightarrow R_Y(\tau) = a^2R_X(\tau) + 2abm + b^2 \quad ; \quad R_X(\tau) = \sigma^2e^{-\beta|\tau|}$$

❖ Part b

The cross correlation function of $X(t)$ and $Y(t)$ are derived as follows:

$$R_{XY}(\tau) = E[x(t)y(t+\tau)] \xrightarrow{Y(t)=aX(t)+b} R_{XY}(t) = E[x(t)(ax(t+\tau)+b)] = E[ax(t)x(t+\tau)] + E[bx(t)]$$

$$\Longrightarrow R_{XY}(\tau) = aE[x(t)x(t+\tau)] + bE[x(t)] \Longrightarrow R_{XY}(\tau) = aR_X(\tau) + bm \quad ; \quad E[x(t)] = m$$

V.Question 5

Follow the procedure below to have the desired equation proven.

$$\rho(A.B) = \rho = \frac{Cov(A.B)}{\sigma^2} \quad ; \quad Cov(A.B) = E[AB] - E[A]E[B] \quad ; \quad E[A] = E[B] = \mu_A = \mu_B = 0$$

$$\Longrightarrow Cov(A.B) = E[AB] \Longrightarrow \rho = \frac{E[AB]}{\sigma^2} \Longrightarrow E[AB] = \rho\sigma^2 \qquad \longrightarrow R_{XY}(\tau) = E[x(t)y(t+\tau)]$$

$$= E[ASin(\omega t+\theta)BSin(\omega t+\theta+\omega\tau)] = E[ABSin(\omega t+\theta)Sin(\omega t+\theta+\omega\tau)]$$

$$= E\left[AB\left(\frac{1}{2}Cos(-\omega\tau) - \frac{1}{2}Cos(2\omega t+2\theta+\omega\tau)\right)\right]$$

$$= E\left[AB\left(\frac{1}{2}Cos(\omega\tau)\right)\right] - E\left[AB\left(\frac{1}{2}Cos(2\omega t+2\theta+\omega\tau)\right)\right]$$

$$= \frac{1}{2}E\left[AB\left(Cos(\omega\tau)\right)\right] - \frac{1}{2}E\left[AB\left(Cos(2\omega t + 2\theta + \omega\tau)\right)\right] = \frac{1}{2}E\left[AB\left(Cos(\omega\tau)\right)\right]$$

$$= \frac{1}{2}Cos(\omega\tau)E[AB] \Longrightarrow R_{XY}(\tau) = \frac{1}{2}Cos(\omega\tau)\rho\sigma^2 \Longrightarrow \boxed{R_{XY}(\tau) = \frac{1}{2}\rho\sigma^2 Cos(\omega\tau)}$$

VI.Question 6

❖ Part a

Let's take the samples number *N=10000*. With just a few lines as in ***main.m*** script, we are able to generate the seeking white noise x(t) and accordingly, y(t) signal. Finally, both signals are plotted easily as in Figure.1.

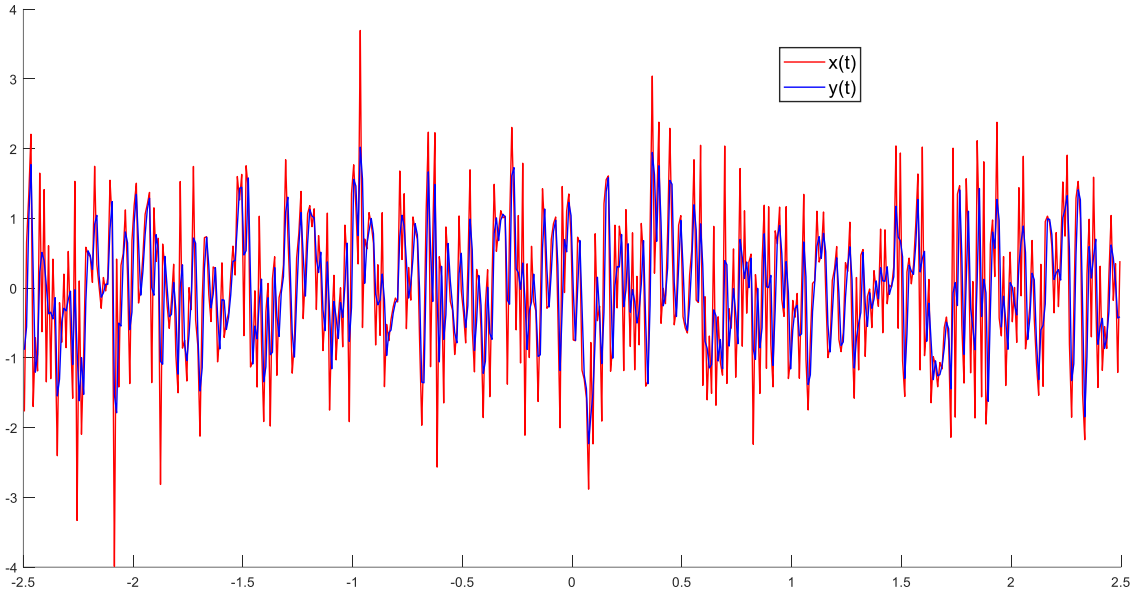


Figure 1- Plots for both x(t) and y(t) signals

❖ Part b

Starting with expanding the equation, we have:

$$C_y(t.l) = E[(y_t - E[y_t])(y_l - E[y_l])] = E[y_t y_l - y_t E[y_l] - y_l E[y_t] + E[y_t]E[y_l]]$$

Also it could be concluded that:

$$E[y_t] = E\left[\frac{1}{2}(x_t + x_{t-1})\right] = \frac{1}{2}(E[x_t] + E[x_{t-1}]) \Longrightarrow E[y_t] = E[y_l] = 0$$

Accordingly:

$$C_y(t.l) = E[y_t y_l]$$

Given $y_t = \frac{1}{2}(x_t + x_{t-1})$, we have:

$$E[y_t y_l] = E\left[\frac{1}{2}(x_t + x_{t-1})\frac{1}{2}(x_l + x_{l-1})\right] = \frac{1}{4}E[x_t x_l + x_{t-1}x_l + x_t x_{l-1} + x_{t-1}x_{l-1}]$$

$$\Longrightarrow E[y_t y_l] = \frac{1}{4}(E[x_t x_l] + E[x_{t-1}x_l] + E[x_t x_{l-1}] + E[x_{t-1}x_{l-1}])$$

For each term above, it is:

$$E[x_t x_l] = \begin{cases} \sigma^2 & for \;\; t = l \\ 0 & for \;\; t \neq l \end{cases}$$

$$E[x_{t-1}x_l] = \begin{cases} \sigma^2 & for \;\; t-1 = l \\ 0 & for \;\; t \neq l \end{cases}$$

$$E[x_t x_{l-1}] = \begin{cases} \sigma^2 & for \;\; t = l-1 \\ 0 & for \;\; t \neq l \end{cases}$$

$$E[x_{t-1}x_{l-1}] = \begin{cases} \sigma^2 & for \;\; t = l \\ 0 & for \;\; t \neq l \end{cases}$$

This leads us to:

$$E[y_t y_l] = \frac{1}{4} \times \begin{cases} 2\sigma^2 & t = l \\ \sigma^2 & |t-l| = 1 \\ 0 & otherwhere \end{cases} \Longrightarrow C_y(t.l) = \frac{1}{4} \times \begin{cases} 2\sigma^2 & t = l \\ \sigma^2 & |t-l| = 1 \\ 0 & otherwhere \end{cases}$$

VII.Question 7

❖ Part a

Follow the *main.m* script and take the sample counts as *N=10000*. Generate the *e*(t) signal using wgn command. Then we try to decrease its mean value to zero by detrend command. Assign 0.3 to a and generate

the $x(t)$ signal using a for loop. The same process happens for $a = 0.9$. Finally, results are obtained as in Figure.2.

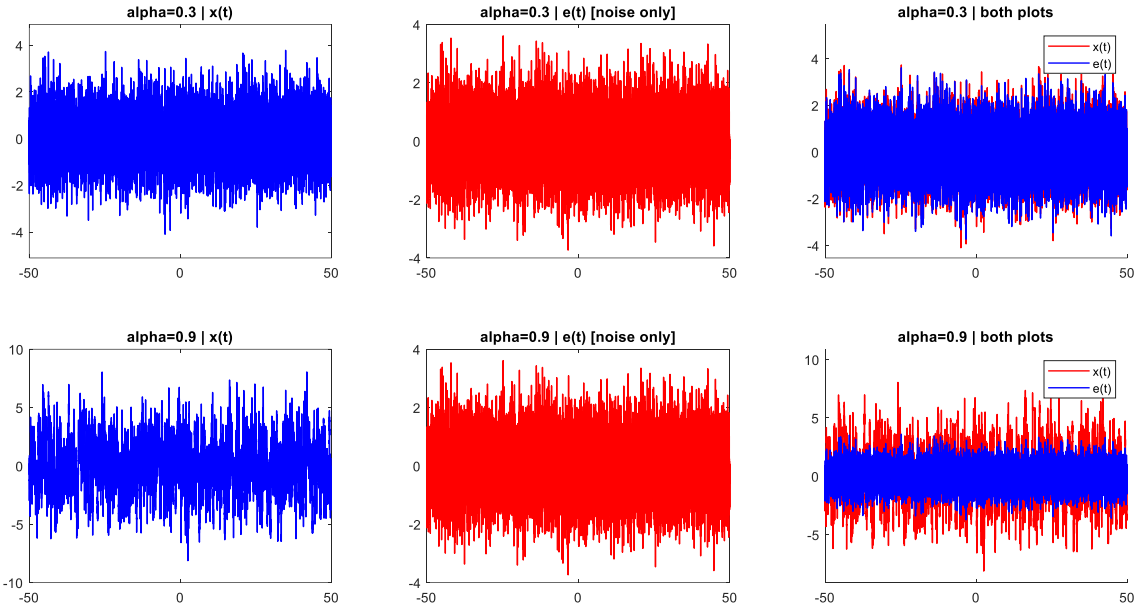


Figure 2- signals plots

❖ Part b

Mathematically, we have:

$$R_X(\tau) = E[x(t)x(t+\tau)] = E\big[\big(ax(t-1)+e(t)\big)\big(ax(t-1+\tau)+e(t+\tau)\big)\big]$$
$$= E[a^2x(t-1)x(t-1+\tau) + ae(t)x(t-1+\tau) + ax(t-1)e(t+\tau) + e(t)e(t+\tau)]$$
$$= E[a^2x(t-1)x(t-1+\tau)] + E[ae(t)x(t-1+\tau)] + E[ax(t-1)e(t+\tau)]$$
$$+ E[e(t)e(t+\tau)]$$

Since

$$\begin{cases} E[a^2x(t-1)x(t-1+\tau)] = a^2E[x(t-1)x(t-1+\tau)] = a^2R_X(\tau) \\ E[ae(t)x(t-1+\tau)] = aE[e(t)x(t-1+\tau)] = aR_{Xe}(\tau-1) \\ E[ax(t-1)e(t+\tau)] = aE[x(t-1)e(t+\tau)] = aR_{Xe}(\tau+1) \\ E[e(t)e(t+\tau)] = 0 \end{cases}$$

then

$$R_X(\tau) = a^2R_X(\tau) + aR_{Xe}(\tau-1) + aR_{Xe}(\tau+1) + 0 \Longrightarrow \boxed{R_X(\tau) = \frac{a}{1-a^2}\big(R_{Xe}(\tau-1) + R_{Xe}(\tau+1)\big)}$$

On the other hand, it could be calculated programmatically in MATLAB too. For so, follow the part b code section of the corresponding *main.py* script. Since we are going to work with numerical values, the a parameter is taken as both 0.3 and 0.9 like the previous part. Figure.3 is illustrating the results for both values of a. You can refer to the script to find out how these signals are obtained.

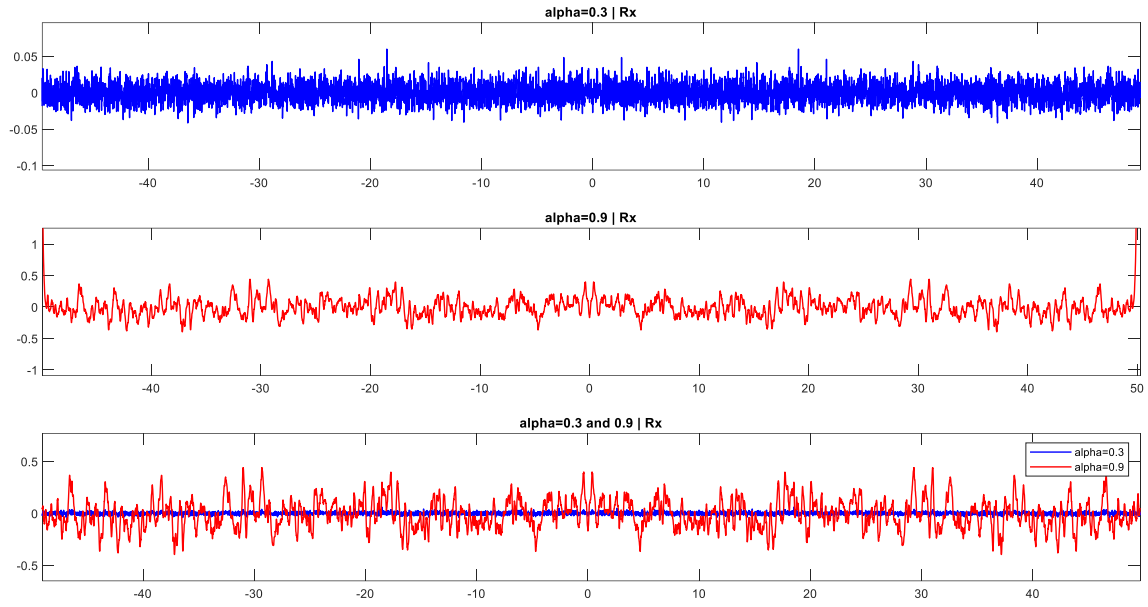


Figure 3- Auto-Correlation plots of the system for a=0.3 and a=0.9

❖ Part c and d

Moving forward with the main script, in code section part c, you could observe how the Power Spectral Density (PSD) is calculated. As a result, the plots are coming in Figure.4 to show the PSD for both a=0.3 and a=0.9.

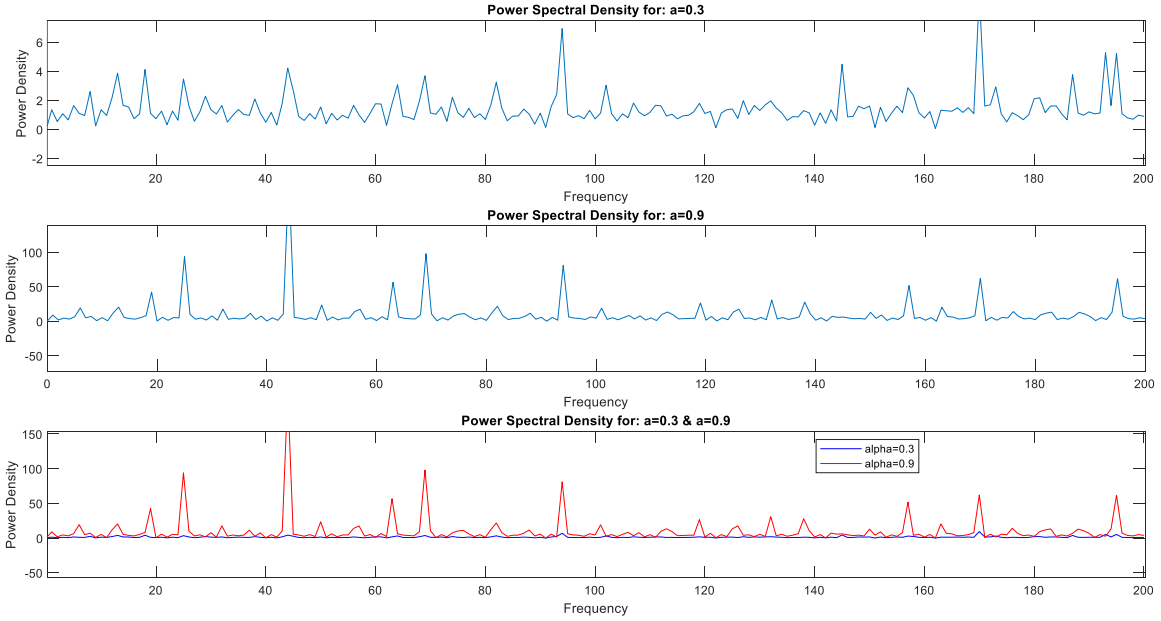


Figure 4- Power Spectral Density Plots for a=0.3 and a=0.9

Note that you can easily access the numeric values by simply running the *main.m* script within the question7 directory in MATLAB.

VIII.Question 8

❖ Part a

Let's plot the given data. See Figure.5.

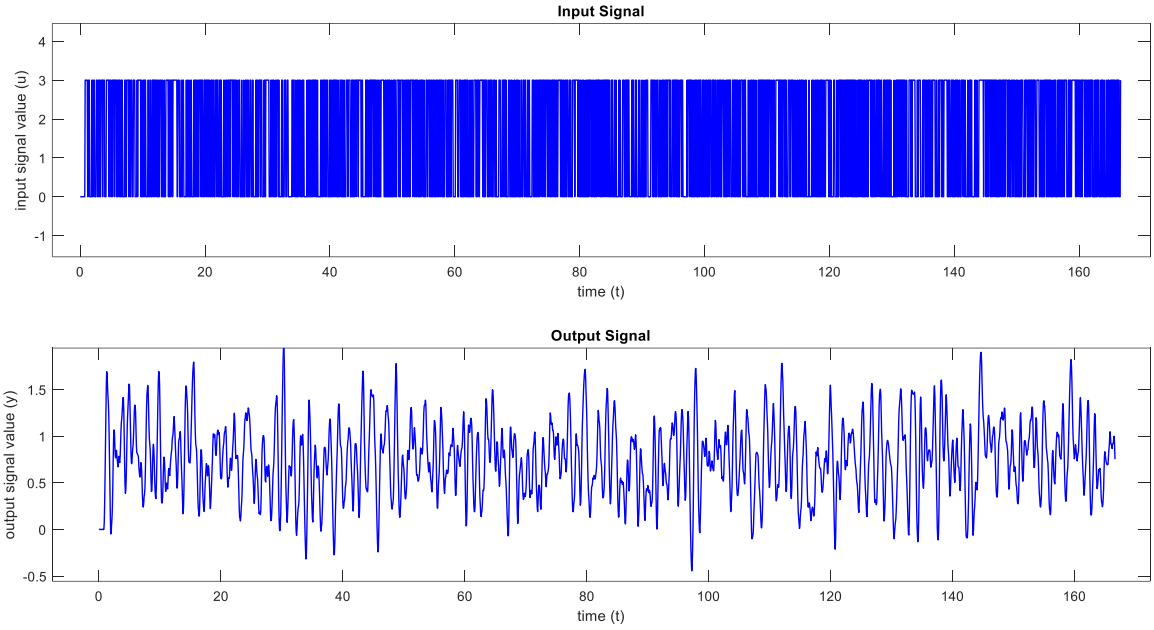


Figure 5- Input and Output signals

❖ Part b

See code section part b within the *main.m* script to find the lines of code for this. Calculations result in:

$$mean(u) = 1.5069 \xrightarrow{detrend(u)} mean(u) = +0.00000$$

$$mean(y) = 1.7545 \xrightarrow{detrend(y)} mean(y) = -0.00000$$

❖ Part c

To answer to this part, let's write two functions in MATLAB so that we can calculate the auto/cross-correlation values easily. The first function is named AutoCorrelate, which is saved as an individual script with the same name as the function, helps us calculate the auto-correlation value of a signal and *k* value. CrossCorrelate is the second function which is written in another separate script with the same name and as its name suggests, it helps us calculate the cross correlation value between two signals with specific *k* value. For your convenience, functions codes are reported in the following.

```
function Rx = AutoCorrelate(x, k)
    N = length(x);
    Rx = 0;
    for i=1:N
        if i+k > N
            Rx = Rx + x(i)*x(i+k-N);
        else
            Rx = Rx + x(i)*x(i+k);
        end
    end
    Rx = Rx/N;
end
```

```
function Rxy = CrossCorrelate(x, y, k)
    N = length(x);
    Rxy = 0;
    for i=1:N
        if i+k > N
            Rxy = Rxy + x(i)*y(i+k-N);
        else
            Rxy = Rxy + x(i)*y(i+k);
        end
    end
    Rxy = Rxy/N;
end
```

Now it is so easy to calculate the auto and cross correlation values using two *for* loops in the *main.m* script within the *part c* code section. Finally, the following plots would be obtained (Figure.6).

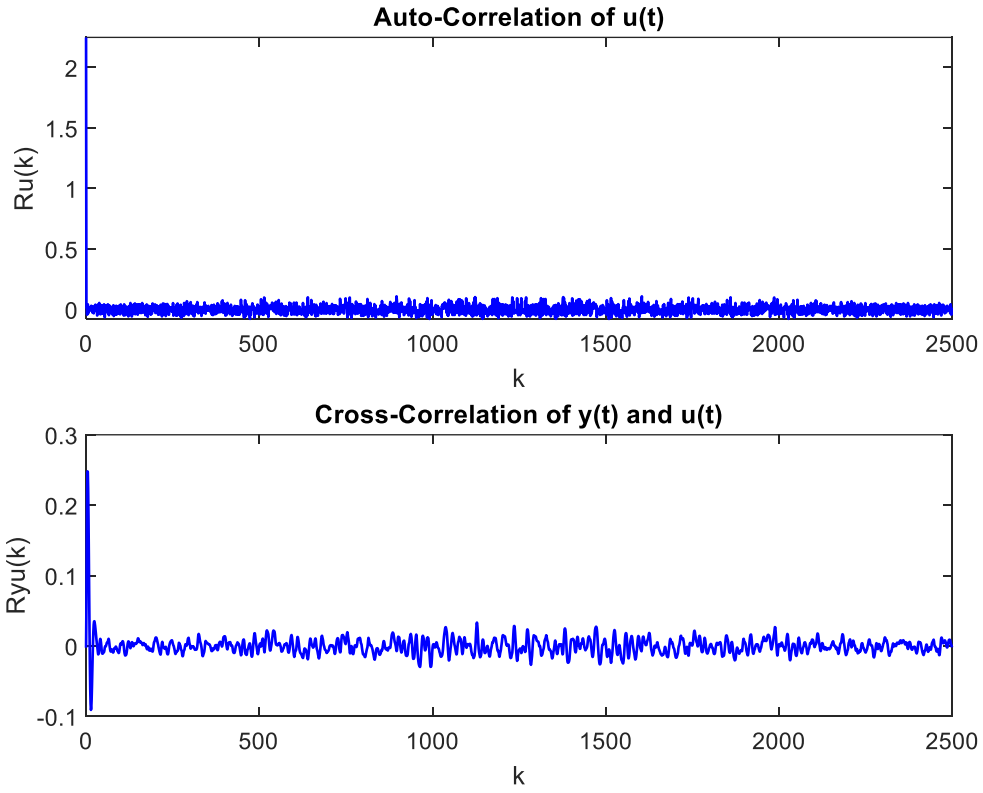


Figure 6- auto & cross correlation plots

Note that the functions are written based on the following formulas:

$$R_{xx}(k) = \frac{1}{N}\sum_{i=1}^{N} x_i\, x_{i+k} \qquad R_{xy}(k) = \frac{1}{N}\sum_{i=1}^{N} x_i\, y_{i+k}$$

❖ Part d and e

We need to take advantage of the following equation in order to estimate the FIR model of the system.

$$R_{uy}(\omega) = R_{uu}(\omega)h(\omega)$$

So the aim is to calculate $R_{uy}(\omega)$ and $R_{uu}(\omega)$, then obtain $h(k)$ using $F^{-1}\left\{h(\omega) = \frac{R_{uy}(\omega)}{R_{uu}(\omega)}\right\}$.

It has been known from the previous section what $R_{xx}(k)$ and $R_{xy}(k)$ functions are. So taking Fourier Transform of these two functions would have $R_{uy}(\omega)$ and $R_{uu}(\omega)$ achieved. Simply use the fft command of MATLAB to take Fourier Transforms of the sequences (see code section *part d*). Then we are capable of performing $h(\omega) = \frac{R_{uy}(\omega)}{R_{uu}(\omega)}$. Finally, it is just required to revert from the frequency domain to time domain with the help of Inverse Fourier Transform (MATLAB command: ifft). At this moment, the FIR model of the system is estimated. Since the real impulse response of the system is available, the estimated FIR model and the real impulse response are compared in Figure.7. It could be observed that the estimated model is perfectly matching with the actual system response.

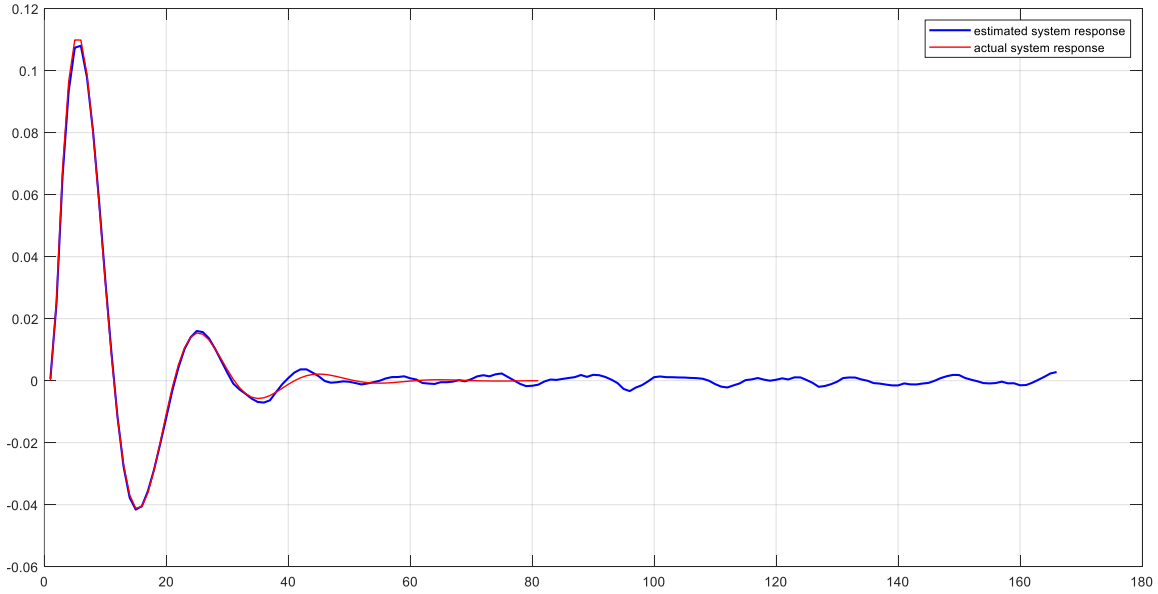


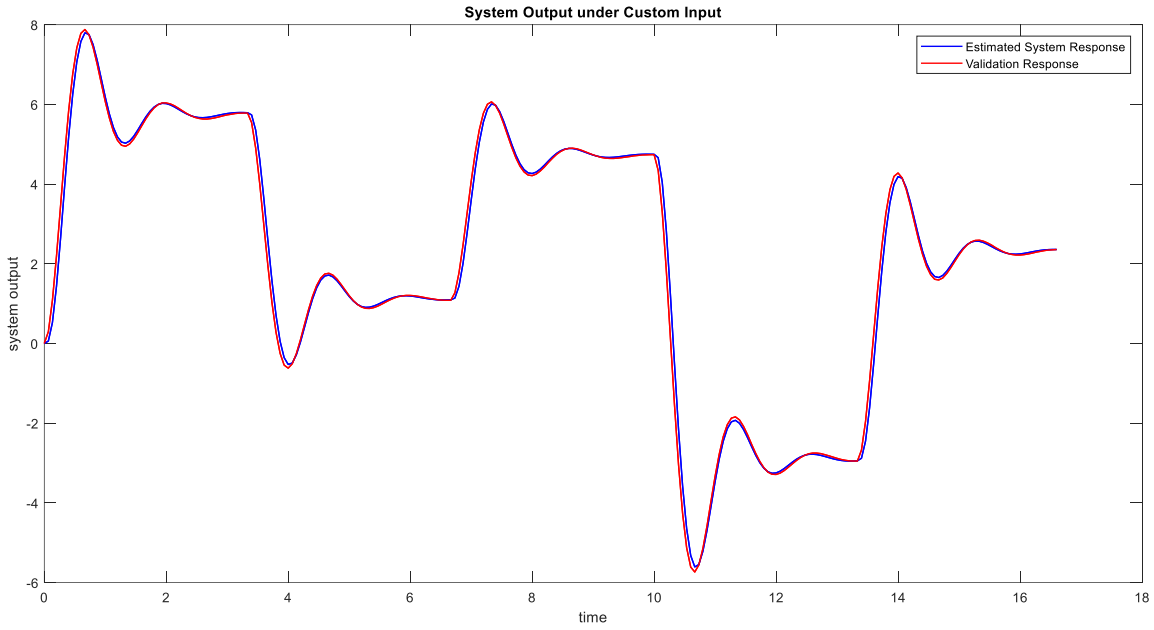
Figure 7- Estimated model and actual system impulse response

❖ Extra Curriculum

We can obtain the transfer function of the system using the Prony method as in the assignment 01. To do so, let's implement the Prony method in a function (see *prony_estimator.m* script). The estimated transfer function using this method is as following:

```
  (0.0006059+3.093e-16i) s^3 - (2.002+2.188i) s^2 + (3.153+51.1i) s + (993-1865i)
-------------------------------------------------------------------------------
s^4 + (73.35-47.14i) s^3 + (1409-2210i) s^2 + (5634-7916i) s + (3.146e04-5.6e04i)
```

In order to validate our estimated model, we can use the available set of validation data. As a result, Figure.8 is approves that the estimated model is working well compared to the actual system.



❖ Appendix

The Prony estimator function is also available in the following:

```
function c_sys = prony_estimator(ht, assumed_degree, Ts)
    y = ht;
    N = length(y);

    % Form D.a = Y ========================================
    % for D ------------------------------------------
    D = zeros(N-assumed_degree-1, assumed_degree);
    for i=1:1:N-assumed_degree-1
        for j=assumed_degree:-1:1
            D(i, assumed_degree-j+1) = y(i+j-1);
        end
    end
    % for Y ------------------------------------------
    Y = zeros(N-assumed_degree-1,1);
    for i=1:1:N-assumed_degree-1
        Y(i, 1) = y(i+assumed_degree+1-1);
    end
    Dplus = inv(D'*D)*D';
    a = Dplus*Y;


    % Calculate the real degree of the system
    threshold = 0.01;
    pseudo_D = D'*D;

    real_degree = length(find(eig(pseudo_D) > threshold));

    % Reform the Prony method with the real degree
    assumed_degree = real_degree;


    % Form D.a = Y ========================================
    % for D ------------------------------------------
    D = zeros(N-assumed_degree-1, assumed_degree);
    for i=1:1:N-assumed_degree-1
        for j=assumed_degree:-1:1
```

```
            D(i, assumed_degree-j+1) = y(i+j-1);
        end
    end
    % for Y -----------------------------------------
    Y = zeros(N-assumed_degree-1,1);
    for i=1:1:N-assumed_degree-1
        Y(i, 1) = y(i+assumed_degree+1-1);
    end
    Dplus = inv(D'*D)*D';
    a = Dplus*Y;

    poly_z = [1 -a'];

    % calculate the Z roots (Zi-s)
    Zi = roots(poly_z);


    % Form Z.B = Y =======================================
    % for Z ------------------------------------------
    Z = zeros(N, assumed_degree);
    for i=1:1:N
        for j=1:assumed_degree
            Z(i,j) = (Zi(j))^(i-1);
        end
    end
    % for Y ------------------------------------------
    Y = zeros(N,1);
    for i=1:1:N
        Y(i, 1) = y(i);
    end

    % solve for `B` using Least Squares method
    Zplus = inv(Z'*Z)*Z';
    B = Zplus*Y;

    s_poles = -log(Zi)/Ts;
    s_gains = B;


    c_sys = 0;
    for index=1:length(s_gains)
        c_sys = c_sys + tf([s_gains(index)], [1 s_poles(index)]);
    end
end
```