SHARIF UNIVERSITY OF TECHNOLOGY, ELECTRICAL ENGINEERING DEPARTMENT

Project 3: Recursive Least Square Estimation

Ali Seyfi Vahid Ahmadi

July 29, 2020

- The system is based on this student number: 94105569
- All the processes are in 1000 seconds and the step input has 100 seconds time delay.
- For the second and third input, the gain is: -0.15.
- In all sections, in the figures that have both blue and red graphs, blue graphs are system response and red graphs are our model response.

1 LEAST SQUARE ESTIMATION

1.1 Determine what is a suitable sample time for this system. Fully explain the reasons behind your choice (you may use information you previously gained about the system).

According to the result of previous project, we determine that the appropriate sample time should be around **1 sample per second** for our system. if you put the sample time a small value, noise spoils the result. Conversely if you consider a big value for sample time, the accuracy of your model will be decreased. Also we have learned some methods to calculate the best sample time in Digital Control course which were helpful.

In the next pages, the inputs and also system's response to each input are plotted.

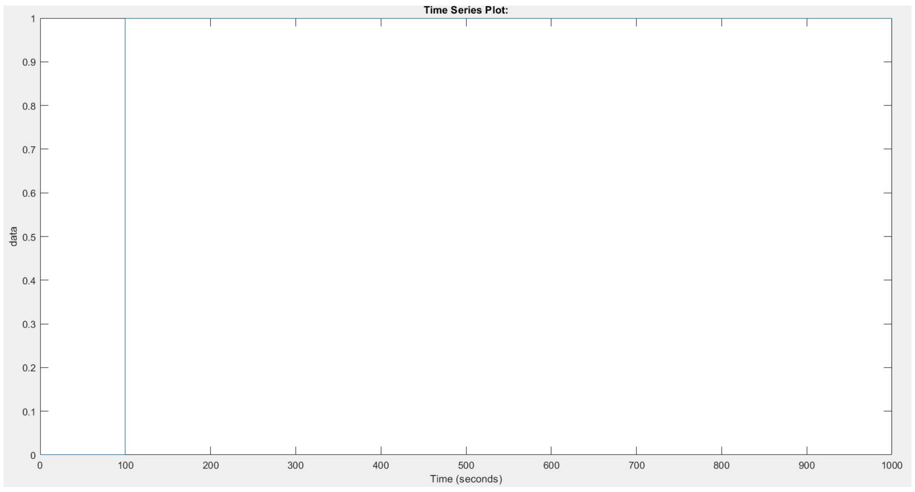

Figure 1.1: step input

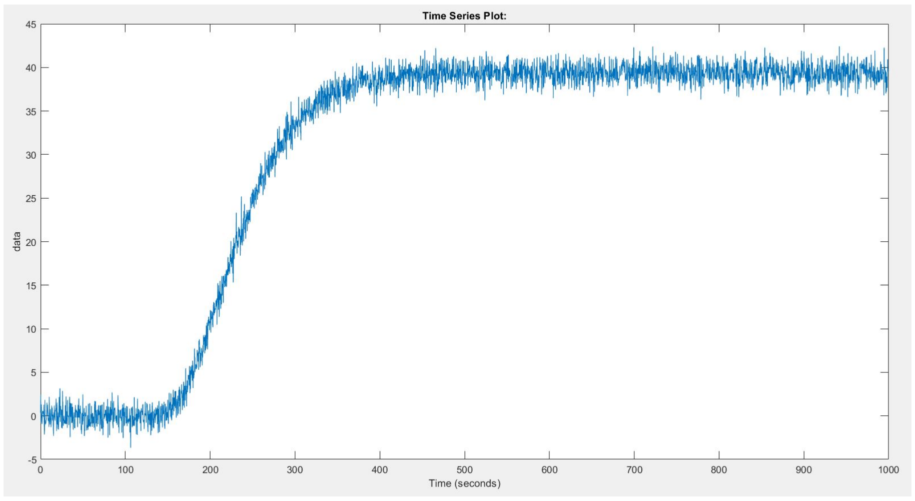

Figure 1.2: System output with step input

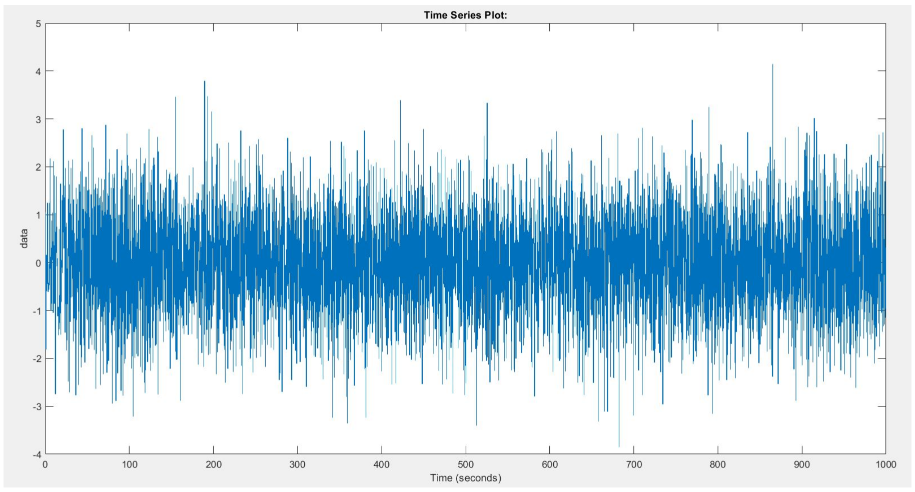

Figure 1.3: zero mean unit variance normal noise input

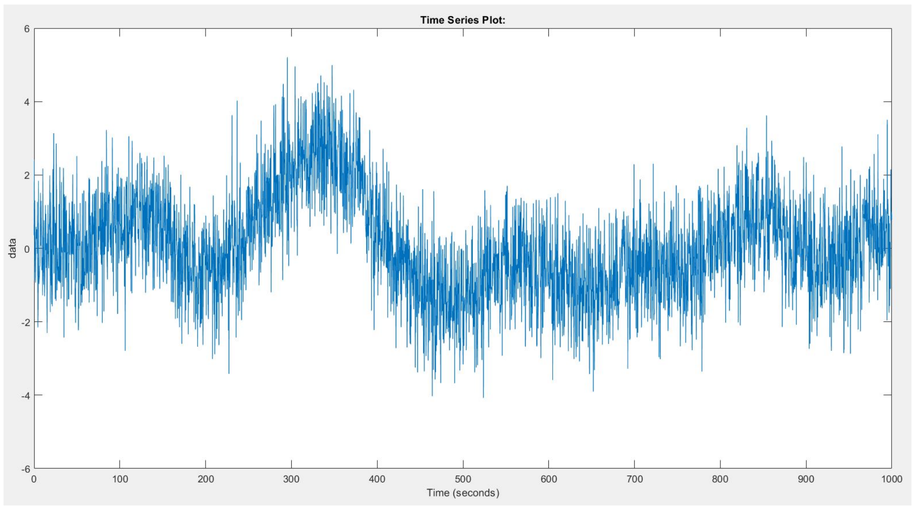

Figure 1.4: System output with first input

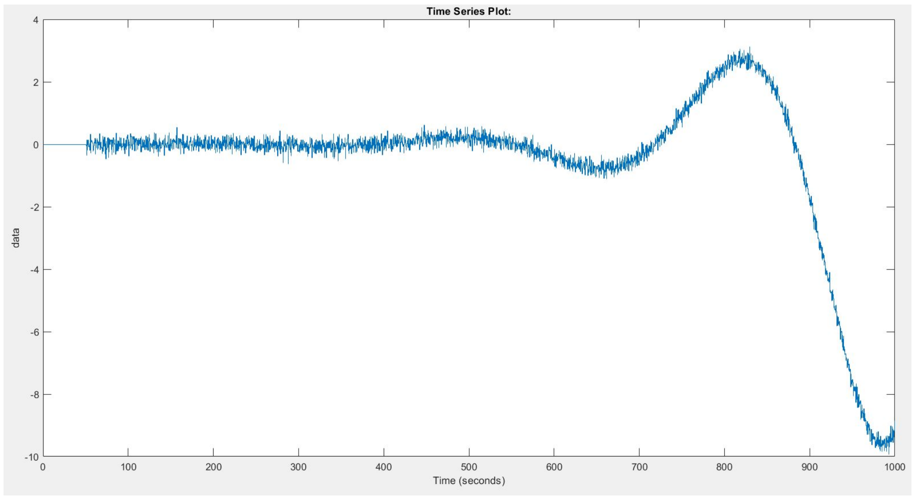

Figure 1.5: second input

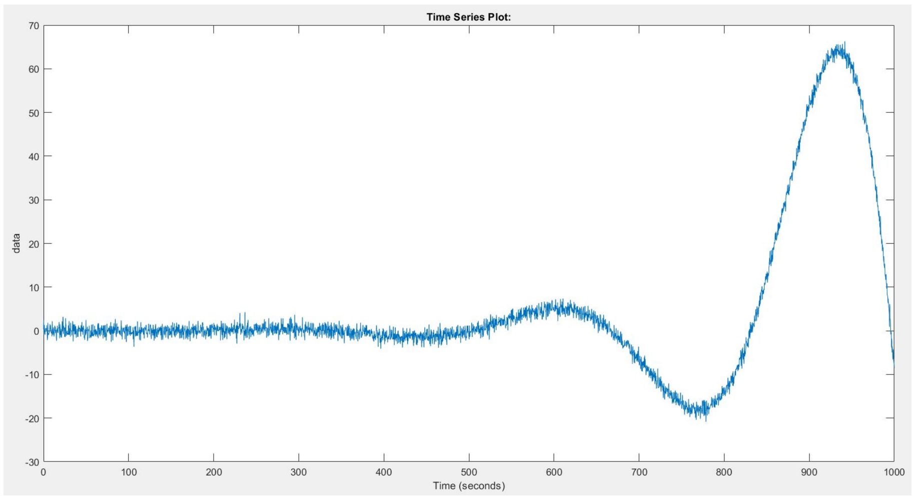

Figure 1.6: System output with second input

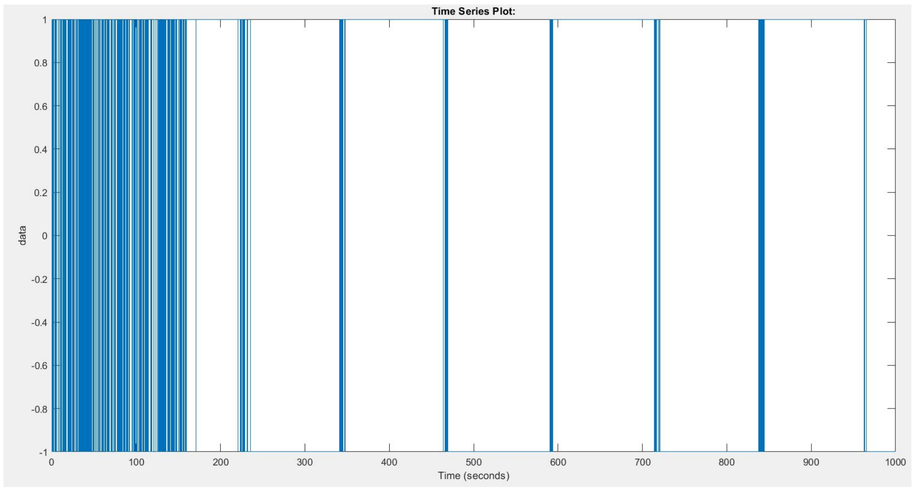

Figure 1.7: third input

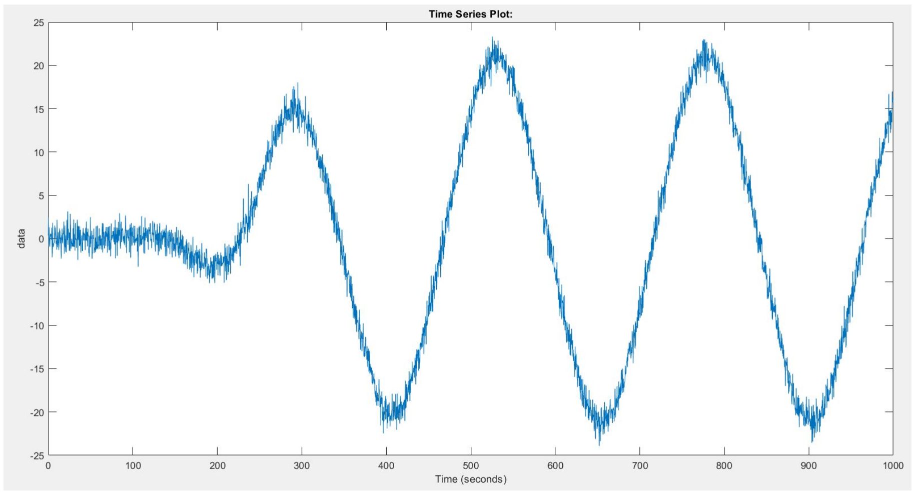

Figure 1.8: System output with third input

1.2 Derive a discrete model structure for a FOTD model.

The continuous time FOTD transfer function is described as follows:

$$P(s) = \frac{K}{Ts+1} e^{-Ls} \tag{1.1}$$

where K is the gain, T is the time constant, and L is the dead time. In this project, the discrete model is designed with sampling interval T_s, and hence, the continuous time model is expressed as follows:

$$P_d\left(z^{-1}\right) = \frac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1}} z^{-(d+1)} \tag{1.2}$$

where a_1, b_0, and b_1 are the coefficient parameters, d is the dead-time in the discrete time, and z^{-1} denotes the backward shift operator. The discrete time plant parameters correspond to the continuous time plant parameters as follows:

$$\begin{aligned} a_1 &= e^{-\frac{T_s}{T}} \\ b_0 &= K\left(1 - a_1 e^{\frac{L_0}{T}}\right) \\ b_1 &= K\left(a_1 e^{\frac{L_0}{T}} - a_1\right) \\ d &= L_1 \end{aligned} \tag{1.3}$$

where a non-negative integer L_1 satisfies the following equation:

$$L = L_1 T_s + L_0 \, (L_0 < T_s) \tag{1.4}$$

Therefore, the input/output relationship in discrete time is given as follows:

$$y(k) = P_d\left(z^{-1}\right) u(k) \tag{1.5}$$

where $y(k)$ is the system output (plant output) and $u(k)$ is the control input.

1.3 Using standard least squares, obtain a least squares estimate of the parameters of the model using each on the above inputs.

According to the previous part, our model should be:

$$y(k) = \frac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1}} z^{-(d+1)} u(k) \tag{1.6}$$

With simplifying our equation we have:

$$y(k) - a_1 y(k-1) = b_0 u(k - (d+1)) + b_1 u(k - (d+1) - 1) \tag{1.7}$$

$$y(k) = a_1 y(k-1) + b_0 u(k - (d+1)) + b_1 u(k - (d+2)) \tag{1.8}$$

$$y(k) = [y(k-1) \quad u(k-(d+1)) \quad u(k-(d+2))] \begin{bmatrix} a_1 \\ b_0 \\ b_1 \end{bmatrix} \tag{1.9}$$

We define:

$$x(k) = [y(k-1) \quad u(k-(d+1)) \quad u(k-(d+2))] \tag{1.10}$$

Then if we make a vector of $y(k)$s, we will have:

$$Y = \begin{bmatrix} y(d+3) \\ ... \\ y(n) \end{bmatrix} \tag{1.11}$$

Also we can make the matrix X, repeating (1.10) for each sample:

$$X = \begin{bmatrix} y(d+1) & u(1) & u(0) \\ & ... & \\ y(n) & u(n-d) & u(n-d-1) \end{bmatrix} \tag{1.12}$$

So our the LS solution will be:

$$\hat{\theta} = \left(X^\top X\right)^{-1} X^\top Y \tag{1.13}$$

$$\hat{\theta} = \begin{bmatrix} a_1 \\ b_0 \\ b_1 \end{bmatrix} \tag{1.14}$$

If we consider $d = 170$, using (1.13), we will have:

$$\hat{\theta} = \begin{bmatrix} +0.9890 \\ +0.6879 \\ -1.2497 \end{bmatrix} \tag{1.15}$$

Having a_1, b_0 and b_1, using (1.3), we can have our model's parameters:

$$k = 39.889 \tag{1.16}$$

$$L = 167.086 \tag{1.17}$$

$$T = 90.522 \tag{1.18}$$

Using the parameters we get in the last part, Our FOTD model will be:

$$P(s)_{LS} = \frac{39.889}{90.522s+1} e^{-167.086s} \tag{1.19}$$

We think we achieve a good model because with comparison with results in project 2, the errors decrease:

$$ISE = 1513 \tag{1.20}$$

$$IAE = 701 \tag{1.21}$$

$$MaxError = 3.95 \tag{1.22}$$

In the following you will see the figures with system(blue) and model(red) responses for different inputs. As you can see in the results, our model could estimate the system very well.

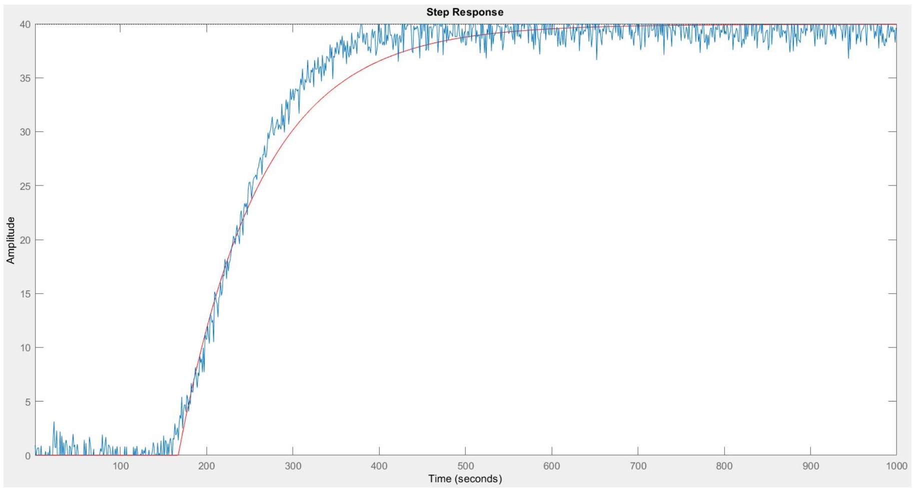

Figure 1.9: System output (blue) and LS model (red) output with step input

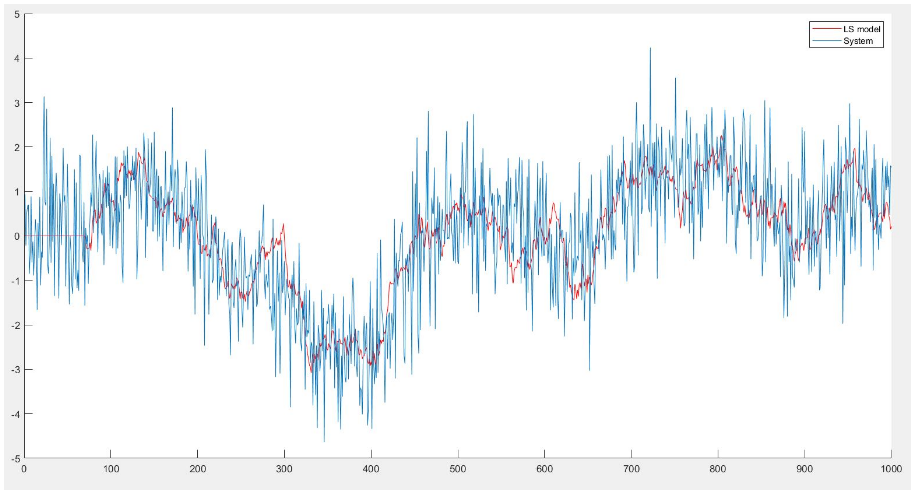

Figure 1.10: System output and LS model output with first input

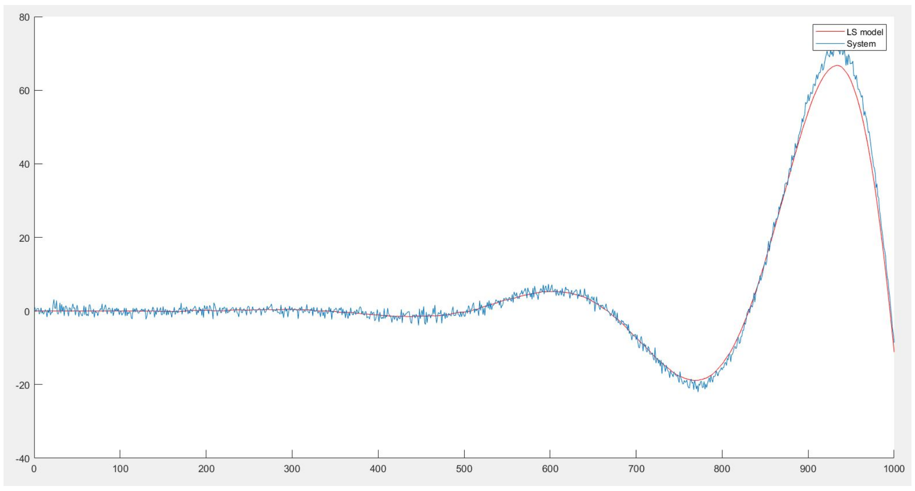

Figure 1.11: System output and LS model output with second input

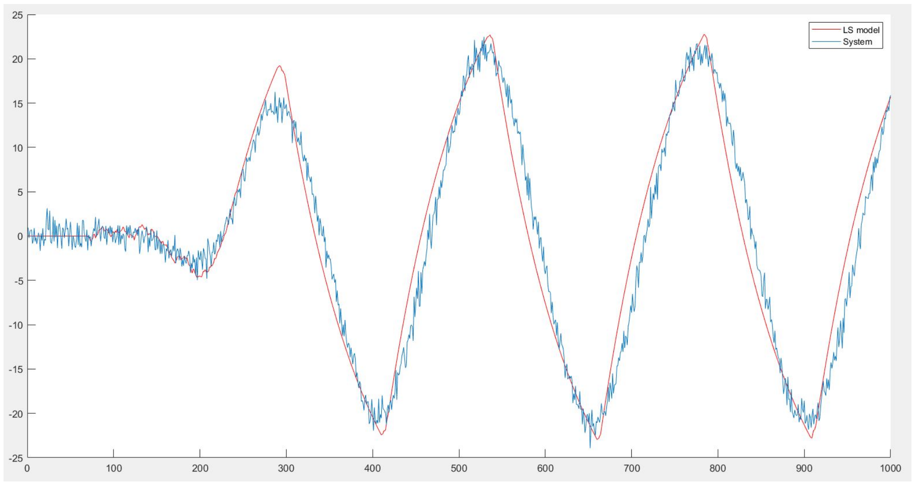

Figure 1.12: System output and LS model output with third input

1.4 Using the RLS algorithm, obtain a least squares estimate of the parameters of the model using each on the above inputs. In each case you need to preset a time graph which shows the estimate of the model parameters at each sample time.

We should follow these steps:
1. Calculate x_{t+1} first. For this calculation we just put $k = t+1$:

$$x(t+1)^T = [y(t) \quad u(t-d) \quad u(t-(d+1))] \tag{1.23}$$

2. Then we should calculate the error. Pay attention that in each iteration $\hat{\theta}$ is obtained from previous iteration.

$$e(t+1) = y(t+1) - x^T(t+1)\hat{\theta}(t) \tag{1.24}$$

3. After that we calculate P_t in each iteration with this formula:

$$P(t+1) = P(t)\left[I_m - \frac{x(t+1)x^\top(t+1)P(t)}{1+x^\top(t+1)P(t)x(t+1)}\right] \tag{1.25}$$

4. In the last step we update $\hat{\theta}$ with an update term plus previous $\hat{\theta}$ data:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + P(t+1)x(t+1)e(t+1) \tag{1.26}$$

If we consider $d = 170$ and repeat 4 steps iteratively, we will have:

$$\hat{\theta} = \begin{bmatrix} +0.9884 \\ +0.7225 \\ -0.2597 \end{bmatrix} \tag{1.27}$$

And again, by having a_1, b_0 and b_1, using (1.3), we can have our model's parameters:

$$k = 39.976 \tag{1.28}$$

$$L = 169.434 \tag{1.29}$$

$$T = 85.871 \tag{1.30}$$

And using these parameters, our FOTD model will be:

$$P(s)_{RLS} = \frac{39.976}{85.871s+1}e^{-169.434s} \tag{1.31}$$

We think we achieve a good model because with comparison with results in project 2, the errors decrease. Even better than LS model!

$$ISE = 1163 \tag{1.32}$$

$$IAE = 614 \tag{1.33}$$

$$MaxError = 3.31 \tag{1.34}$$

For a step input you can see how $\hat{\theta}$ parameter changes through the time, error between system and RLS model through the time and also output of both system and RLS model for it. Strange shape of the error through the time is for noise of the system. Also for all of the three input which are mentioned in the project we do the same process and you can see the results.

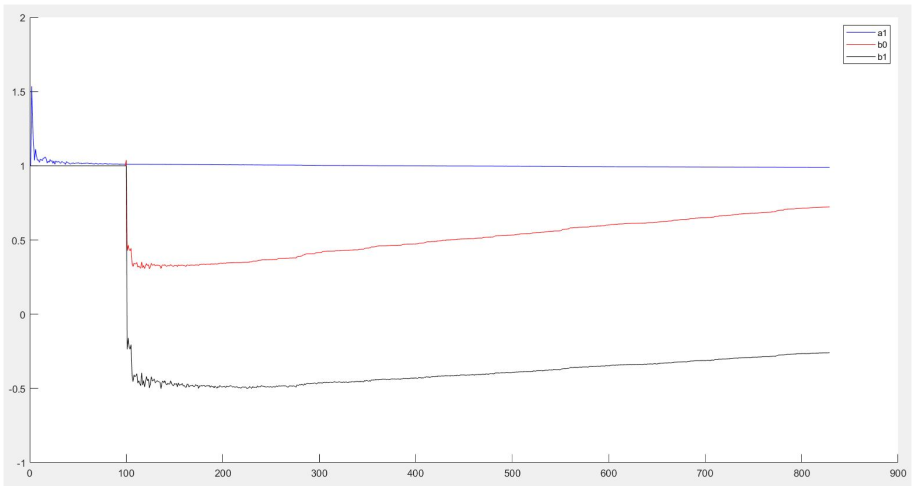

Figure 1.13: parameter changes through the time for a step input

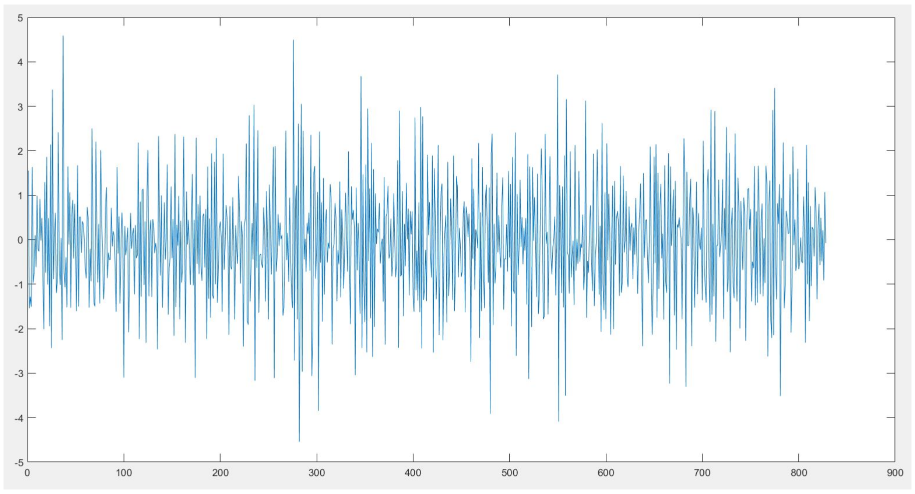

Figure 1.14: error between system and RLS model through the time for a step input

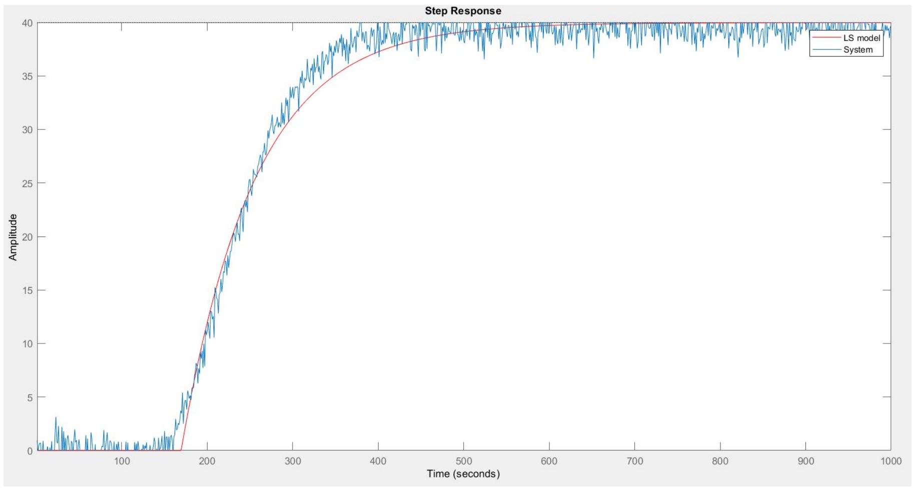

Figure 1.15: output of both system and RLS model for a step input

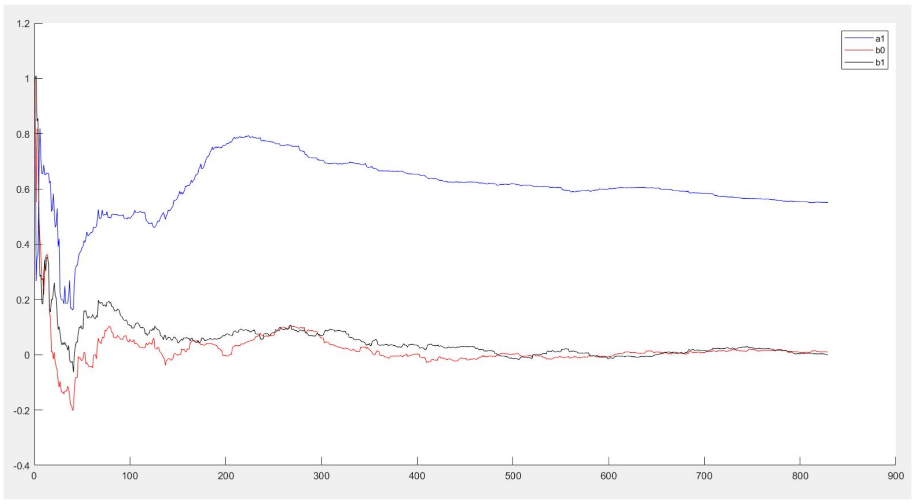

Figure 1.16: parameter changes through the time for first input

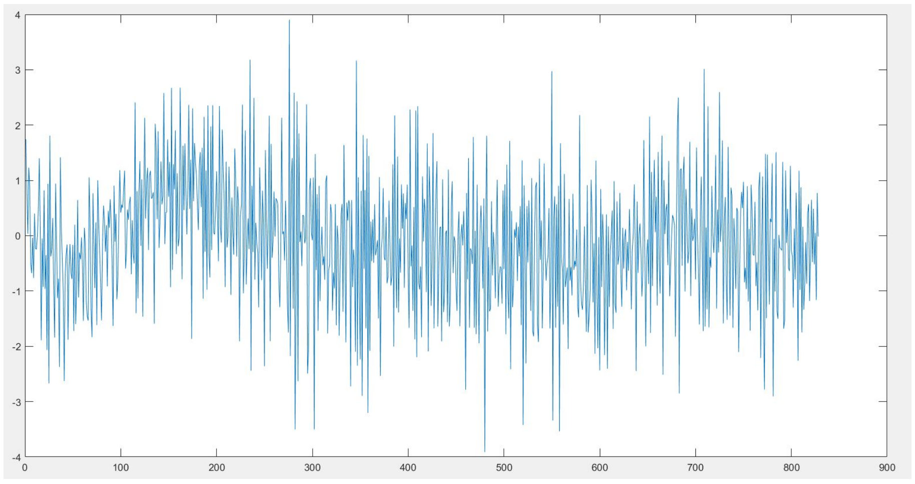

Figure 1.17: error between system and RLS model through the time for first input

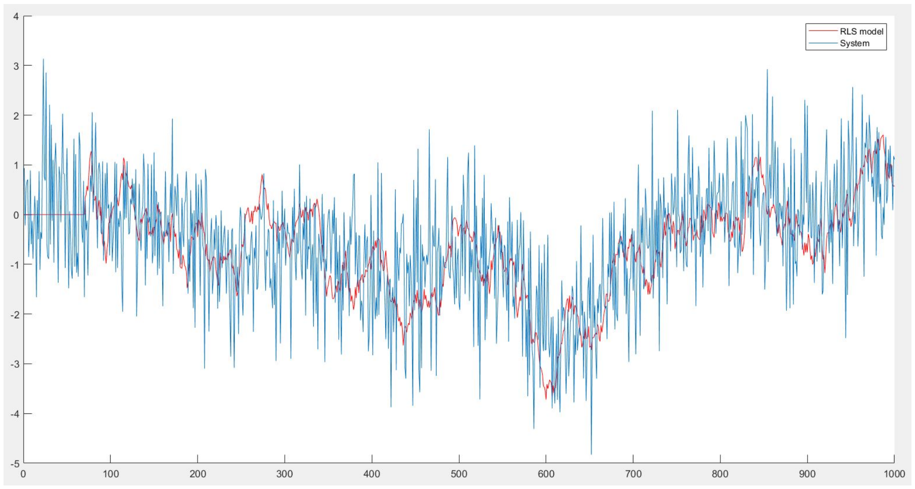

Figure 1.18: output of both system and RLS model for first input

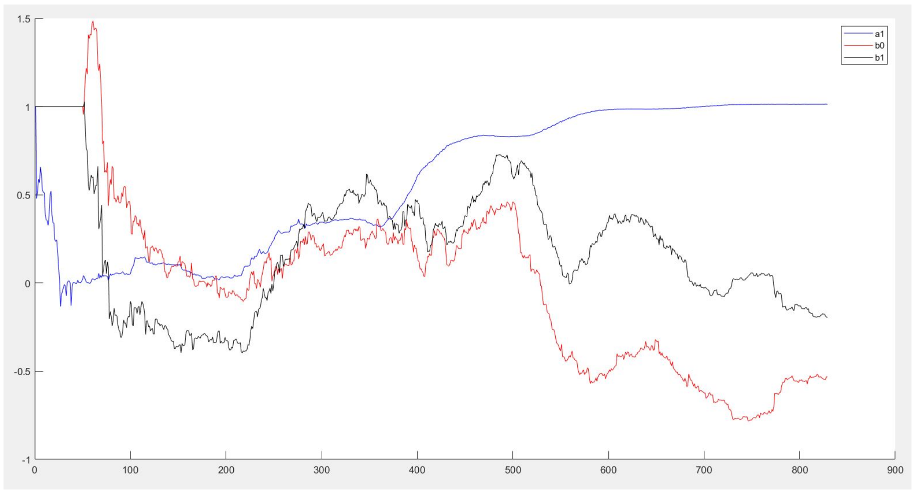

Figure 1.19: parameter changes through the time for second input

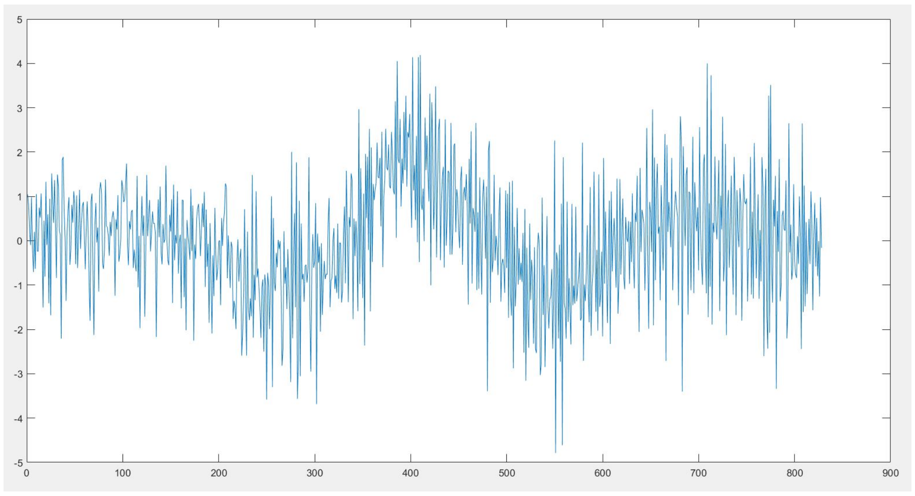

Figure 1.20: error between system and RLS model through the time for second input

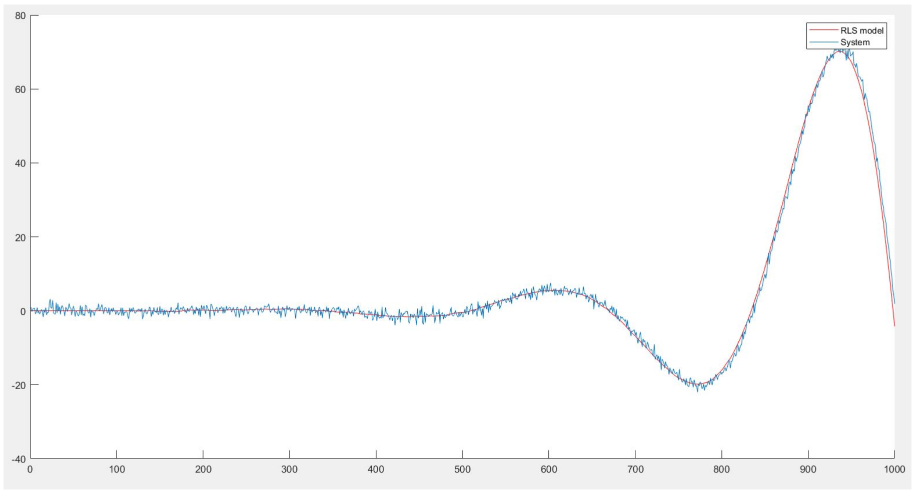

Figure 1.21: output of both system and RLS model for second input

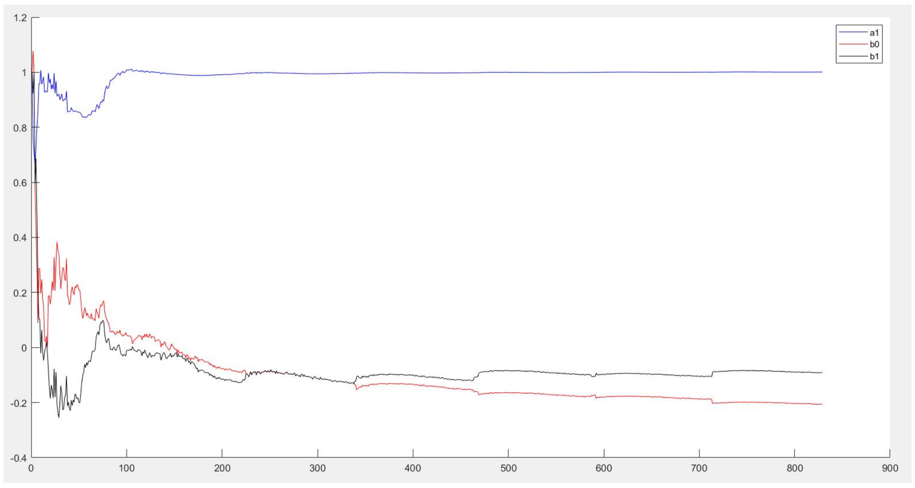

Figure 1.22: parameter changes through the time for third input

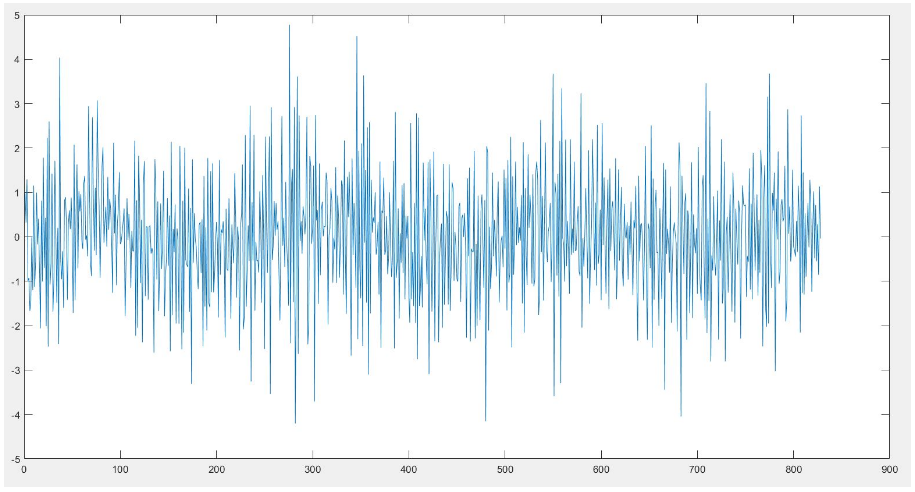

Figure 1.23: error between system and RLS model through the time for third input

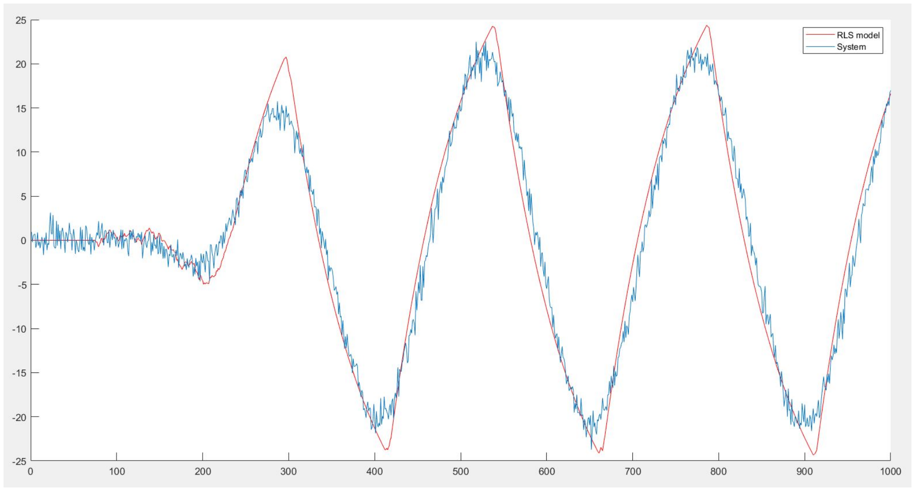

Figure 1.24: output of both system and RLS model for third input

1.5 Present a detailed discussion on the effect of the input on the performance of the RLS algorithm. Your analysis should;

1. Include an analysis of the estimation residuals including a statistical analysis (covariance).
2. Contain suitable graphical means to display the results of your analysis.
3. Contain a discussion on whether the observations are inline with your expectations.

1. We know that:

$$Cov(\hat{\theta}) = \sigma_e^2 \underset{x}{\mathbb{E}}[(X^T X)^{-1}] \tag{1.35}$$

so from this equation we can estimate our residual.

2. For this part, you can look at the Figure 1.18. Where $\sigma_e^2 = 1$ and $\mathbb{E}_X[(X^T X)^{-1}] = \begin{bmatrix} 0.0000 & -0.0008 & -0.0003 \\ -0.0008 & 1.0233 & -0.9917 \\ -0.0003 & -0.9917 & 1.0043 \end{bmatrix}$. So from (1.35), we can figure out that the co-variance of our $\hat{\theta}$ is equal to $\begin{bmatrix} 0.0000 & -0.0008 & -0.0003 \\ -0.0008 & 1.0233 & -0.9917 \\ -0.0003 & -0.9917 & 1.0043 \end{bmatrix}$

3. As we compute the variance of our $\hat{\theta}$, we obtain an almost similar result which was inline with our expectations from theory.

1.6 By experimenting with different initial values for P and $\hat{\theta}$, discuss the impact of the initial conditions on the speed of convergence and the accuracy of the estimate. You need to find a suitable means of displaying your results. What can you conclude from your observations?

According to the results here, if we change initial vector of $\hat{\theta}$, there will be no difference in the results, because after a little time, they converge to the same vector and it does not matter from what vector of $\hat{\theta}$ you start (but not strange initial point!).

Conversely, the initial matrix of P is important for your accuracy and also speed of the convergence. If you choose smaller initial values for P indices, $\hat{\theta}$ will convergence slowly, but if you choose larger initial values for P indices, $\hat{\theta}$ will convergence faster. For obtaining a good accuracy you have to choose a good initial vector for $\hat{\theta}$ and a good initial matrix for P. Their initial values are depend on the system and you should set them by experimenting with different initial values.

By our observations we can conclude that if you set initial values near to the final values of the P and $\hat{\theta}$, it would be better. But it depends on your situation and also your budget maybe! We put 1 for both initial values in our model.

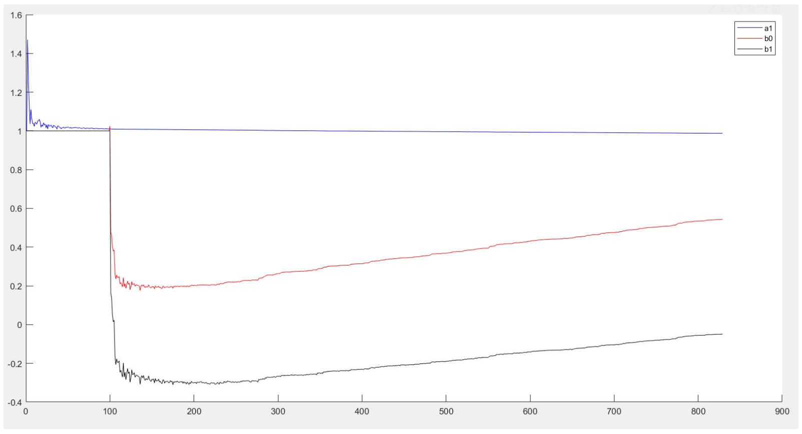

Figure 1.25: step input - initial $P = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$

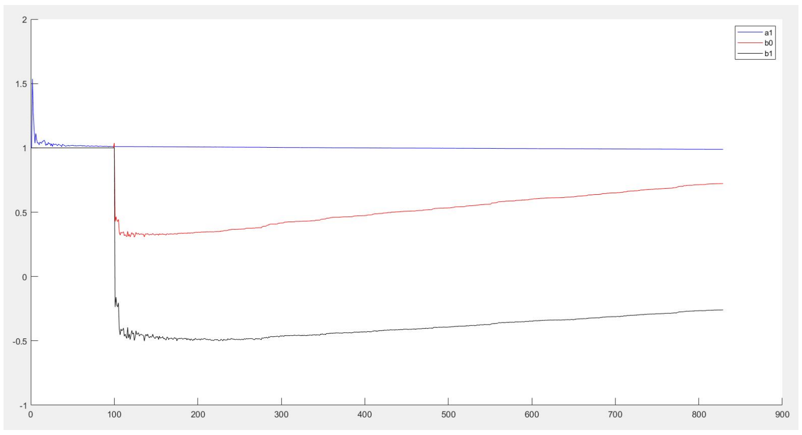

Figure 1.26: step input - initial $P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

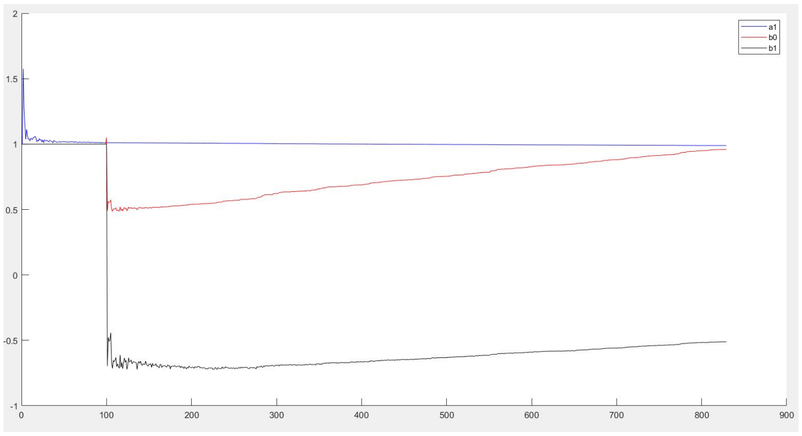

Figure 1.27: step input - initial $P = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$

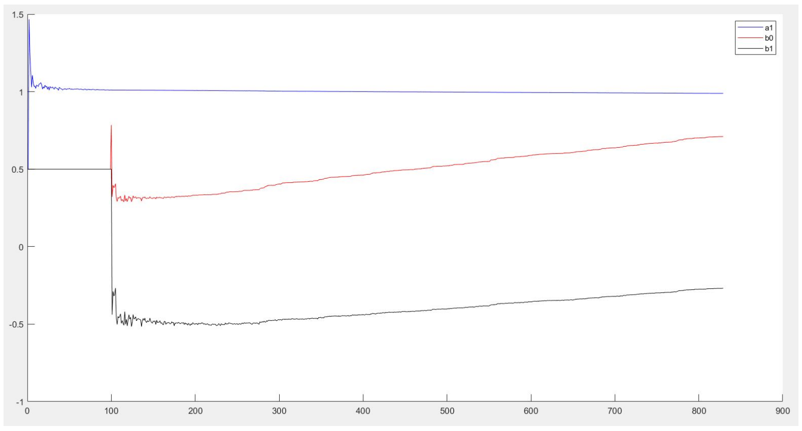

Figure 1.28: step input - initial $\theta = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$

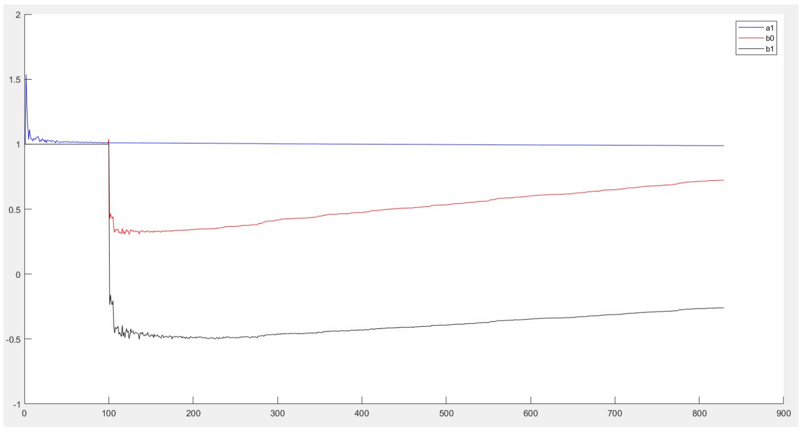

Figure 1.29: step input - initial $\theta = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

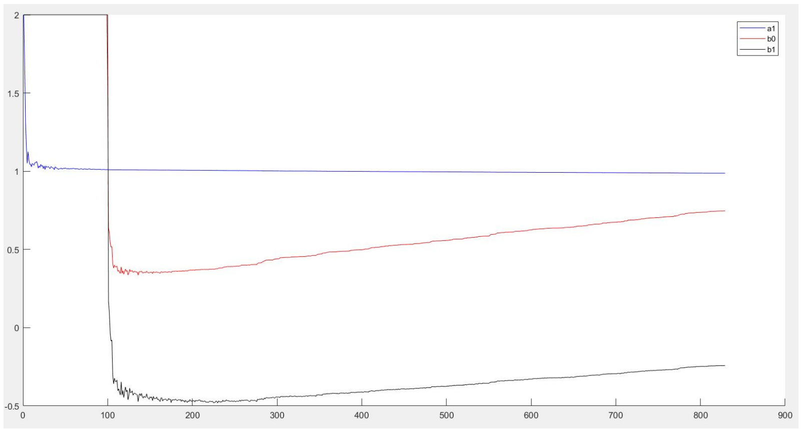

Figure 1.30: step input - initial $\theta = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}$

1.7 Show mathematically, if the results of the LS and the RLS algorithms should be the same? Use the LS and RLS models you found to verify your prediction.

In the LS method, our problem is actually finding the minimum of a degree two function. In RLS method, we use an iterative approach to find the minimum. If the cost function which we want to minimize it, had several minimums, our RLS approach could go and stuck in that minimun. But as our cost function have only one minimum, the result of LS and RLS methods should be same. As we can see in the following the systems that these systems give us are almost the same.

1.8 Compare the computational CPU and Memory resources required to calculated the LS and the RLS models.

Memory resources:

In LS method, As we store all the input and output vectors and their delay vectors, We have a $(1000-d)*1$ vector for storing Y values, and a $(1000-d)*3$ vector to store X values. In contrast, in RLS method we just store integers $e(t+1)$ and $y(t+1)$,and also matrix $P(t+1)$ which is $3*3$ and vectors $X(t+1)$ and $\theta(t+1)$ which are $3*1$. So if we compare these two method due to memory usage, the LS method uses memory more.

Computational CPU:

In LS method, we need to compute the inverse of matrix $X^T X$ which is a $3*3$ matrix and that is the most time consuming step of the algorithm. On the other hand, in RLS method, we have to compute e and matrix P and vector θ each time for each time samples. So if we want to compare two algorithms due to CPU usage, the RLS method uses CPU more.

References:

1. Discrete-Time First-Order Plus Dead-TimeModel-Reference Trade-off PID Control Design - Ryo Kurokawa (2019)
2. PID Controllers, Theory, Design and Tuning (2nd Edition) - Åström, Karl