In The Name of God

Homework 2

System Identification

Mohamad Alikhani

40109494

Dr. Mehdi Aliyari

Fall of 2023

Contents

Analytical questions

Question 1

The ARMAX model is as follows:

$$A(q^{-1})y(t) = B(q^{-1})U(t) + C(q^{-1})e(t)$$

For Recursive ARMAX:

1. Initial the values n_a, n_b, n_c, λ, $\hat{\theta}(0)$ and $P(0)$.
2. In steps $t = n$, calculate the output
3. Recall the values of e, U and y from $\phi(n)$.

   $\phi(n) = [-y(n-1)\,, U(n-1), e(n-1)]^T$

4. Carry out the RLS algorithm for $e(n)$, $\phi(n)$, $\hat{\theta}(n-1)$ and P(n).
   $e(n) = y(n) - \phi(n)^T \hat{\theta}(n-1)$

   $$P(n) = \frac{1}{\lambda}\left[P(n-1) - \frac{P(n-1)\phi(n)\phi(n)^T P(n)}{\lambda + \phi(n)^T P(n-1)\phi(n)}\right]$$

   $K = P(n)\phi(n)$

   $\hat{\theta}(n) = \hat{\theta}(n-1) + K\, e(n)$

5. Replace the previous results with the results from RLS algorithm
6. $t = n + 1$, repeat from step 2

Question 2

To estimate the state space parameters of a system with given impulse response, we need to consider the three cases outlined:

System without Measurement and Process Noise:

For a system without measurement and process noise, the state-space representation is given by:

$$x(k+1) = Ax(k) + B(k)$$

$$y = Cx(k)$$

If you have the impulse response, you can determine the matrices A, B, and C by considering the relationship between the impulse response and the system matrices.

Let h(k) be the impulse response, and n be the order of the system. The state-space matrices are related to the impulse response as follows:

$$C = [1 \quad 0 \quad \ldots \quad 0]$$

$$A = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & 0 & 1 & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ -h(0) & -h(1) & -h(2) & \cdots & -h(n-1) \end{pmatrix}$$

$$B = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

System with Measurement and State Noise:

In the presence of measurement and state noise, the state-space representation becomes:

$$x(k+1) = Ax(k) + B(k) + \omega(k)$$

$$y = Cx(k) + v(k)$$

Here, $\omega(k)$ is the process noise, and $v(k)$ is the measurement noise. The system matrices A, B and C can still be estimated from the impulse response using methods like subspace identification or instrumental variable methods.

Feasibility and Conditions of Realization:

For a system to be physically realizable, the matrices A, B and C must satisfy certain conditions. These conditions ensure that the system can be physically implemented.

Stability: The eigenvalues of matrix A should lie inside the unit circle for a discrete-time system to be stable.

Controllability: The system should be controllable, meaning that the matrix $[B, AB, A^2B, \ldots, A^{n-1}B]$ should have full rank, where n is the order of the system.

Observability: The system should be observable, meaning that the matrix $[C^T, (CA)^T, (CA^2)^T, \ldots, (CA^{n-1})^T]$ should have full rank.

Realizability: There should be a physical interpretation for the state variables. For example, the state variables should represent measurable physical quantities.

By verifying these conditions, you can determine if the given impulse response corresponds to a physically realizable system and estimate the state-space parameters accordingly.

Question 3

In systems with poles on the $j\omega$ axis, if the system is excited with a sine signal in a very long time, the system will become unstable eventually, which is not desired. But if this happens for a short time, there will be no such a problem. So in the identification task of such systems, the input shall not be sine signal. In these cases using signals such as noise, PRBS will not be problematic from the stability perspective, and beside that, they contain a vast array of frequencies. Thus using this array of signals, linear models are better-performing.

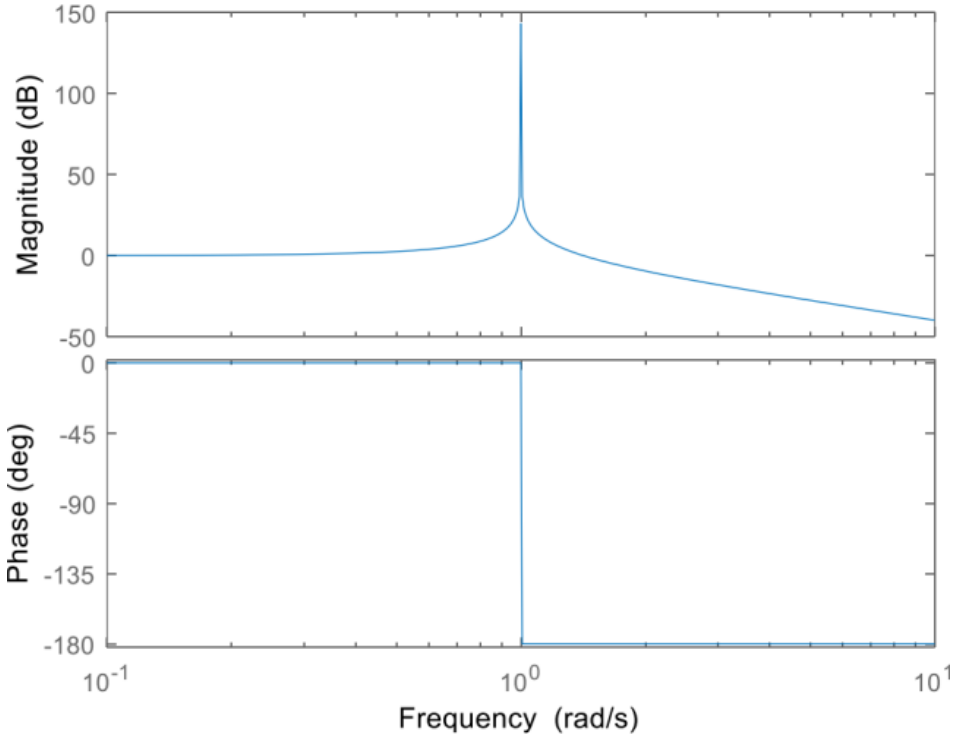

Figure 1 Frequency response of a system, with poles on the imaginary axis to step signal

If the signal has diverse frequencies, there is no problem with it. But if it has the same frequency as the poles of the system, as can be seen in Figure 1 resonance will occur and the system goes to instability and the identification task would not take place.

In integrative systems, with a pole on the origin, the pole will appear in the transfer function and somehow it integrates the input over time. Now if the input has a DC value, or saying in other words, the Furrier transformation of the signal for $\omega = 0$ is not zero, we will have a problem. Therefore for such systems the DC gain of the system must be zero. So we can conclude that the signals such like PRBS that have a DC value are not suitable for identification of these systems, but zero mean noise

is a good choice, and choosing a good linear model would be significantly important as well.

Question 4

Take the model of a system as below:

$$y(k) = b_1u(k-1) + \cdots + b_mu(k-m) - a_1y(k-1) - \cdots - a_my(k-m)$$

Now multiply both sides in $(k-\kappa)$:

$$y(k)u(k-\kappa) = u(k-\kappa)\{b_1u(k-1) + \cdots + b_mu(k-m) - a_1y(k-1) - \cdots - a_my(k-m)\}$$

With the assumption $\kappa<k\leq N$, we sum all the elements:

$$\sum_{k=\kappa+1}^{N} y(k)u(k-\kappa) = corruy(\kappa)$$

According to the above equation and using correlation, we have:

$$corruy(\kappa) = b_1corruu(\kappa-1) + \cdots + b_mcorruu(\kappa-m) - a_1corruy(\kappa-1) - \cdots - a_mcorruy(\kappa-m)$$

Using correlation for parameter estimation, is called COR-LS, which says instead of using LS algorithm, we can use correlation functions.

In this method, the regressors' matrix is:

$$U = \begin{bmatrix} corr_{uu}(0) & \dots & corr_{uu}(1-m) & -corr_{uy}(0) & \dots & -corr_{uy}(1-m) \\ corr_{uu}(1) & \dots & corr_{uu}(2-m) & -corr_{uy}(1) & \dots & -corr_{uy}(2-m) \\ \vdots & & \vdots & \vdots & & \vdots \\ corr_{uu}(l-1) & \dots & corr_{uu}(l-m) & -corr_{uy}(l-1) & \dots & -corr_{uy}(l-m) \end{bmatrix}$$

$$\theta = \begin{bmatrix} b_1 \\ \vdots \\ b_m \\ a_1 \\ \vdots \\ a_m \end{bmatrix}, \quad Y = \begin{bmatrix} corr_{uy}(1) \\ corr_{uy}(2) \\ \vdots \\ corr_{uy}(l) \end{bmatrix}$$

Finally using the above and the LS method we can estimated the system parameters.

Using this algorithm has an **advantage**:

With COR-LS method, the problem of correlation of Nosie and regressors is solved.

The **disadvantage** is:

Because of the use of $corr_{uy}$ and $corr_{uu}$ functions, the computational complexity increases.

Simulation Questions

Question1

1)

Input Signal Generation:

First we have to generate the signal that we want to carry out the identification with.

Using the function below, we can generate the signal. This signal has three levels.

$$u = \text{binrand}(T, N, t_{\min}, t_0, \text{dist})$$

Which in this function T is the time range in which the function will be generated, N is the number of pules, t_{min} is the width of the smallest pules, t_0 starting time, abd dsit is the distribution of the time ranges of pules.

First we need to know the parameters better:

1. Width of the pulses follow a normal distribution in a specific range.
2. The summation of the widths, must be equal to the range specified.
3. None of the pulses' width shall be smaller than t_{min}.
4. The levels of pulses should be chosen in a way that two successive pulses be of two different levels.

For the points above, we must use the below algorithm:

1. Generate $N-1$ random numbers

$$m = \text{rand}(N-1, 1)$$

2. Normalize the random numbers to the range [0,1]

$$m_{\text{nomalized}} = \frac{m - \min(m)}{\max(m) - \min(m)}$$

3. Using the equation below we somehow convert the numbers to the range $\left[t_{\min}, \left(\frac{T-t_0+1}{N-1}\right)\right]$:

$$n = \left[\left(\frac{T - t_0 + 1}{N-1}\right) - t_{\min}\right] \times m_{\text{normalized}} + t_{\min}$$

4. Round the numbers generated from last step to the nearest integer number, this will create $N-1$ pulse widths which are bigger than t_{min}.

$$d = \text{round}(n)$$

5. Now regulate the last pulse's width in a way that the summation of all the widths become T-t0+1. So the last pulse width would be

$$d_N = T - t_0 + 1 - \sum_{i=1}^{N-1} d_i$$

6. Using the conditions below generate the levels(three levels)
   - First pulse must not be zero, it must be between 1 and -1 so as to be sure where the signal starts.
   - Two successive pulses must not have the same levels.

The below plot is the output of this function for parameters ([1:800], 10, 40, 1, 'normal'):

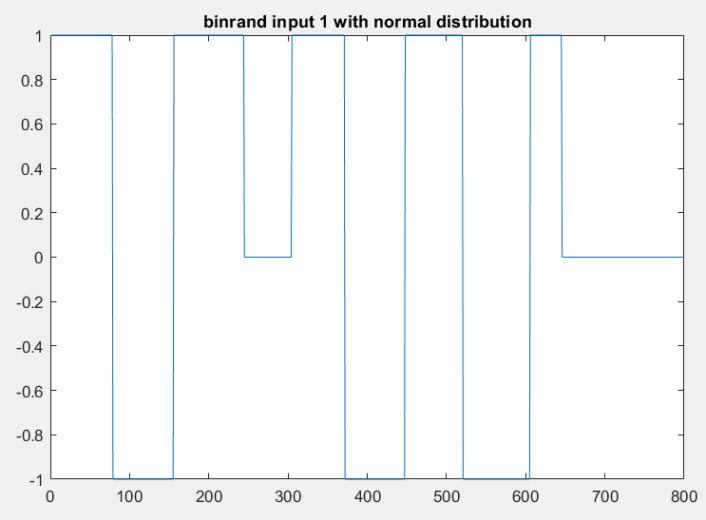

Figure 2 The input signal of first binrand function

The below plot is the output of this function for parameters ([1:800], 30, 10, 1, 'normal'):

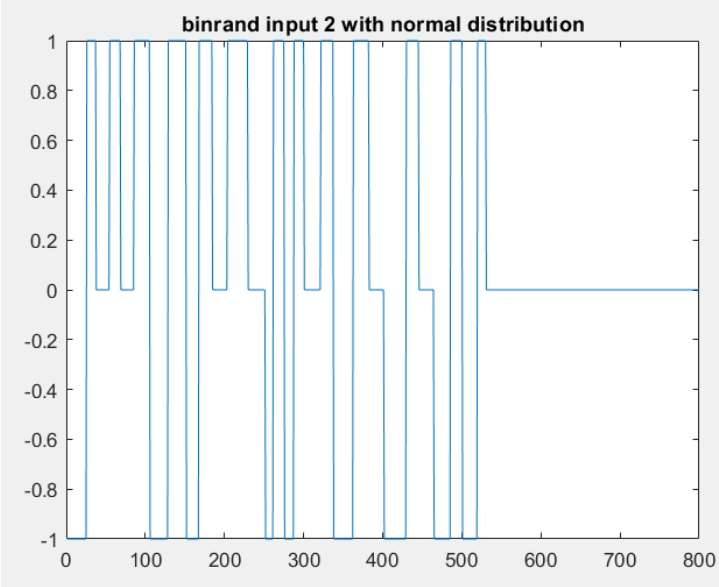

Figure 3 The input signal of second binrand function

Systems transfer function is given as below:

$$T[n] = \frac{y[n]}{r[n]} = \frac{0.48z^{-1} - 0.48z^{-2}}{1 - 1.72z^{-1} + 0.9z^{-2}} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Using the command below we generate the input, and by running the code we plot the corresponding outputs of ARX and ARMAX:

$$r = \text{binrand}([1:800],10,40,1,'\text{normal}')$$

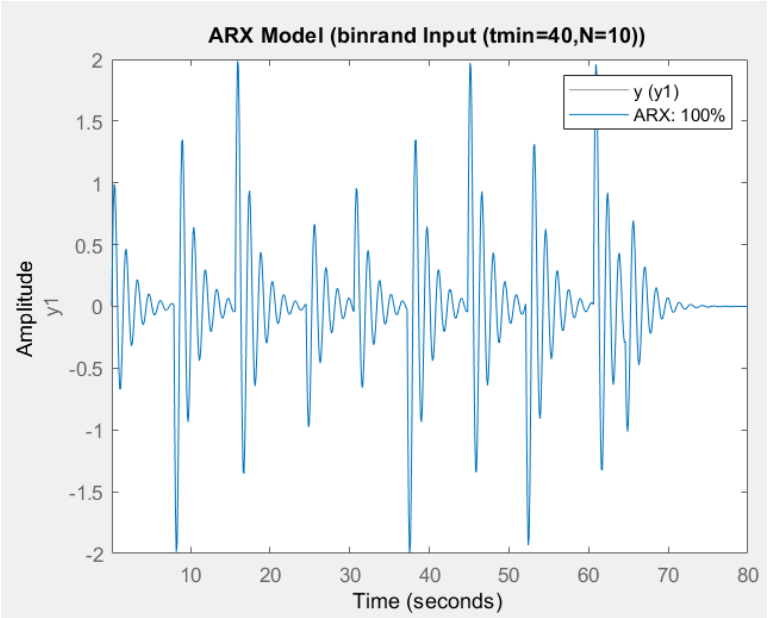

Figure 4 comparison between the real system and ARX algorithm estimation for first input

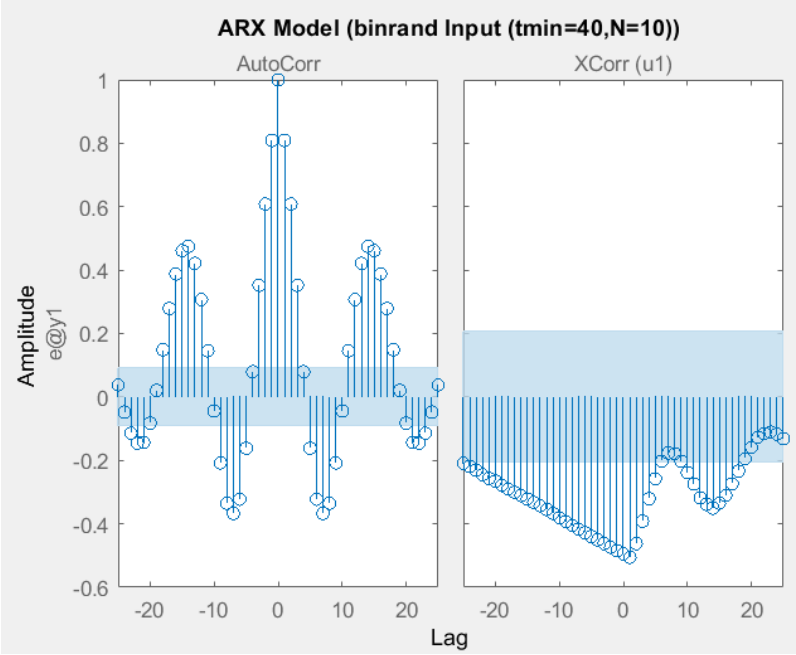

Figure 5 resid for first input

```
>> estimated_y_1_arx

estimated_y_1_arx =
Discrete-time ARX model: A(z)y(t) = B(z)u(t) + e(t)
  A(z) = 1 - 1.72 z^-1 + 0.9 z^-2

  B(z) = 0.48 z^-1 - 0.48 z^-2

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=2   nb=2   nk=1
   Number of free coefficients: 4
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARX on time domain data "data_1".
Fit to estimation data: 100% (prediction focus)
FPE: 4.019e-32, MSE: 3.959e-32
>>
```

Estimation for ARMAX model is as below:

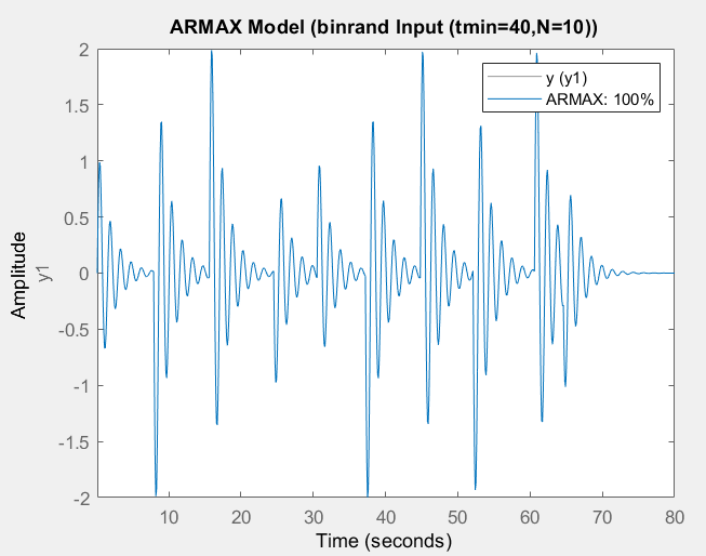

Figure 6 comparison between the real system and ARMAX algorithm estimation for first input

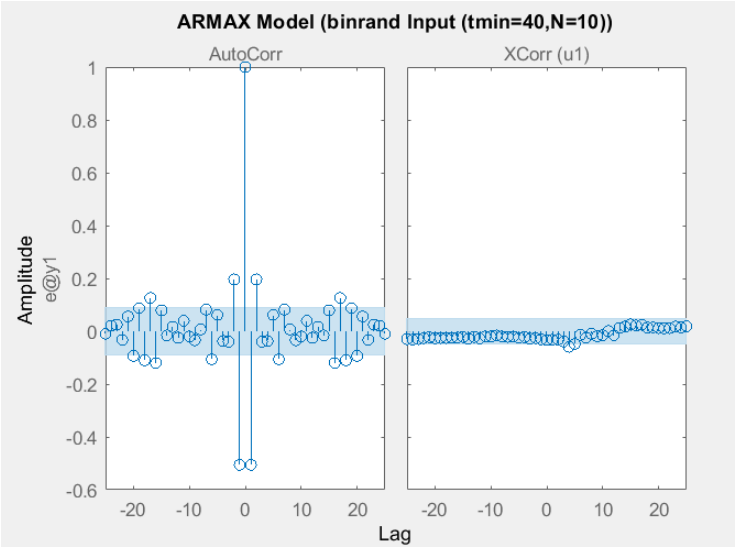

Figure 7 resid for the first input

```
estimated_y_1_armax =
Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)
  A(z) = 1 - 1.72 z^-1 + 0.9 z^-2

  B(z) = 0.48 z^-1 - 0.48 z^-2

  C(z) = 1 + 0.5336 z^-1

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=2   nb=2   nc=1   nk=1
   Number of free coefficients: 5
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARMAX on time domain data "data_1".
Fit to estimation data: 100% (prediction focus)
FPE: 8.173e-33, MSE: 8.071e-33
```

Poles and zeros of the system:

$$poles_{armax_1} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$zeros_{armax_1} = 1$$

$$poles_{arx_1} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$zeros_{arx_1} = 1$$

$$poles_{sys} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$zeros_{sys} = 1$$

It is obvious that the poles and zeros of the system, ARX and ARMAX are the same.

Now we repeat the experiment with the second input which is:

The new input is:

$$r = \text{ binrand } ([1:800],30,10,1, \text{ 'normal' })$$

Number of pulses are increased and the amplitude is decreased!

ARX estimation yields to the results below:

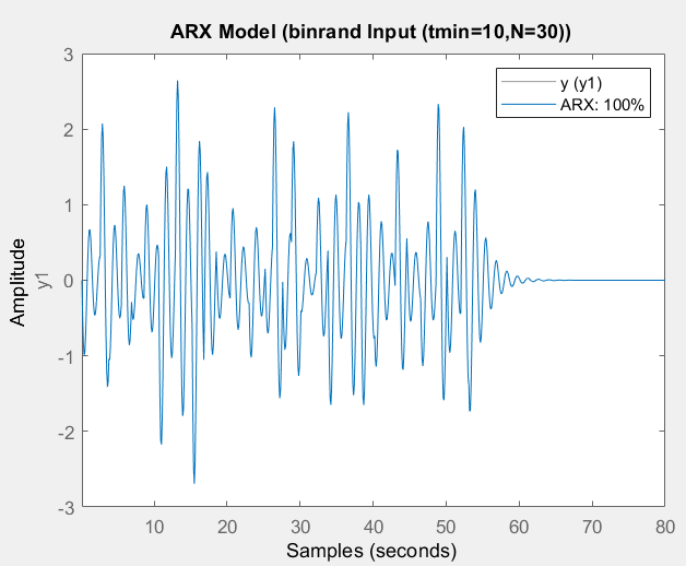

Figure 8 Comparison between ARX and the output for second input

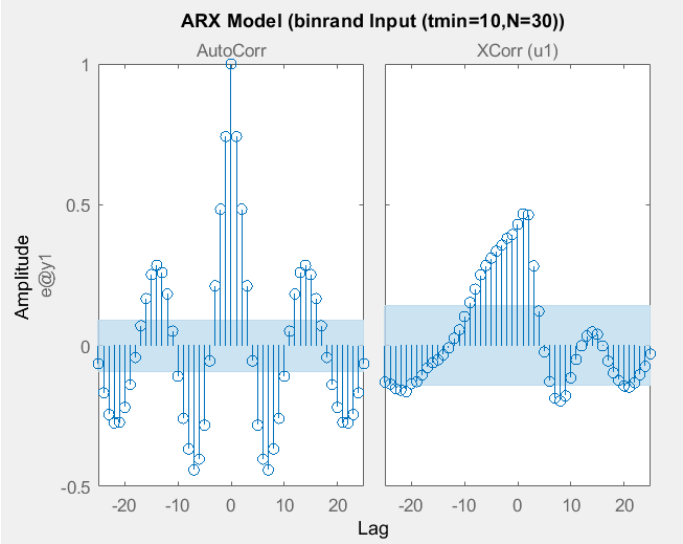

Figure 9 resid for the second input

```
estimated_y_2_arx =
Discrete-time ARX model: A(z)y(t) = B(z)u(t) + e(t)
  A(z) = 1 - 1.72 z^-1 + 0.9 z^-2

  B(z) = 0.48 z^-1 - 0.48 z^-2

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=2   nb=2   nk=1
   Number of free coefficients: 4
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARX on time domain data "data_2".
Fit to estimation data: 100% (prediction focus)
FPE: 8.391e-32, MSE: 8.266e-32
```

ARMAX estimation yields to estimation below:

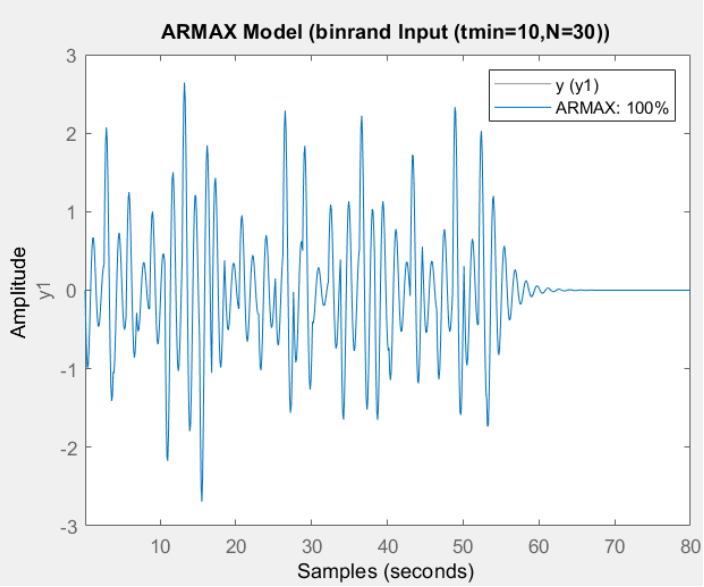

Figure 10 Comparison between ARMAX and the output for second input

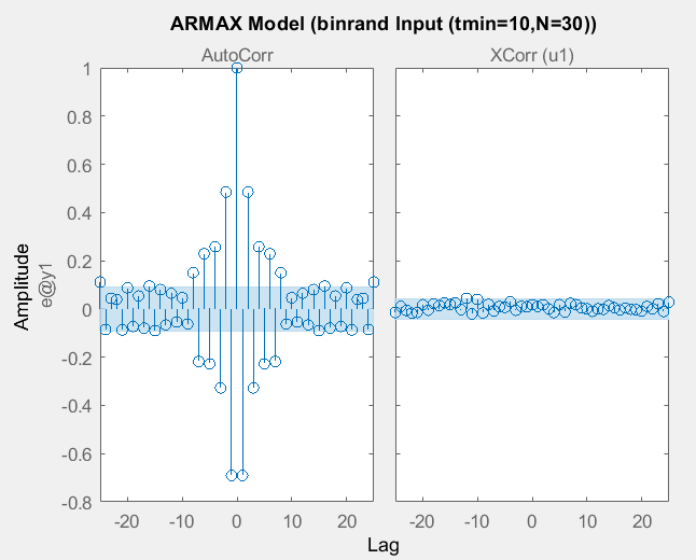

Figure 11 resid for second input

```
estimated_y_2_armax =
Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)
  A(z) = 1 - 1.72 z^-1 + 0.9 z^-2

  B(z) = 0.48 z^-1 - 0.48 z^-2

  C(z) = 1 + 0.7354 z^-1

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=2   nb=2   nc=1   nk=1
   Number of free coefficients: 5
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARMAX on time domain data "data_2".
Fit to estimation data: 100% (prediction focus)
FPE: 2.443e-32, MSE: 2.412e-32
```

Poles and zeros of the system:

$$poles_{armax_1} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$zeros_{armax_1} = 1$$

$$poles_{arx_1} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$zeros_{arx_1} = 1$$

$$poles_{sys} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$zeros_{sys} = 1$$

Just as before there is not a lot of change in the pole and zeros, after changing the input of the algorithms. So it can be inferred that in the absence of noise, the input to the system suffices for the estimation.

For Normal input which is acquired using this:

$$r = \text{normrnd}(0, 1, 600, 1)$$

Third input and ARX:

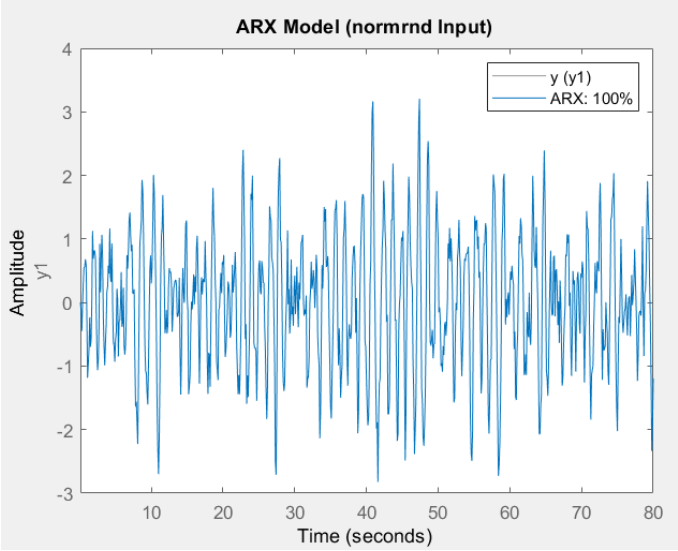

Figure 12 Comparison between ARX and the output for third (normal) input

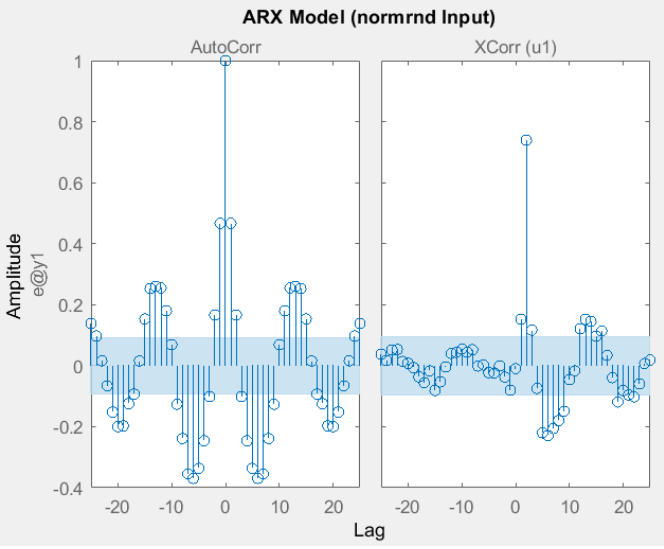

Figure 13 resid for third (normal) input for ARX

```
estimated_y_3_arx =
Discrete-time ARX model: A(z)y(t) = B(z)u(t) + e(t)
  A(z) = 1 - 1.72 z^-1 + 0.9 z^-2

  B(z) = 0.48 z^-1 - 0.48 z^-2

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=2   nb=2   nk=1
   Number of free coefficients: 4
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARX on time domain data "data_3".
Fit to estimation data: 100% (prediction focus)
FPE: 2.676e-31, MSE: 2.636e-31
```

The third input and **ARMAX**:

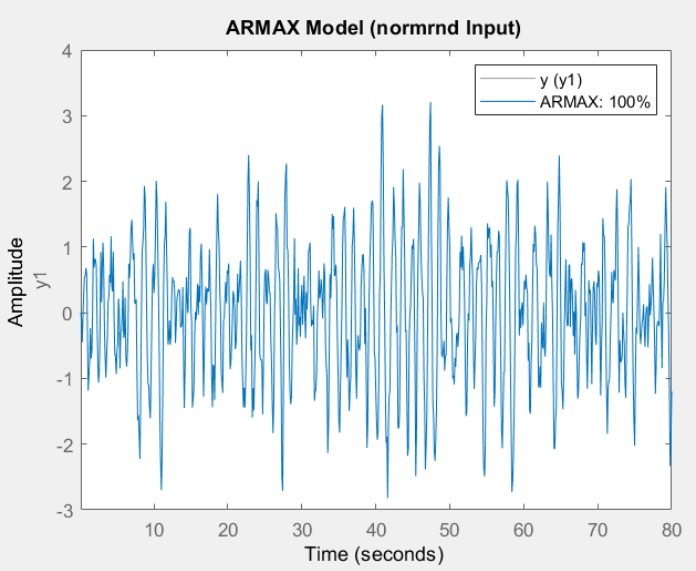

Figure 14 Comparison between ARMAX and the output for third (normal) input

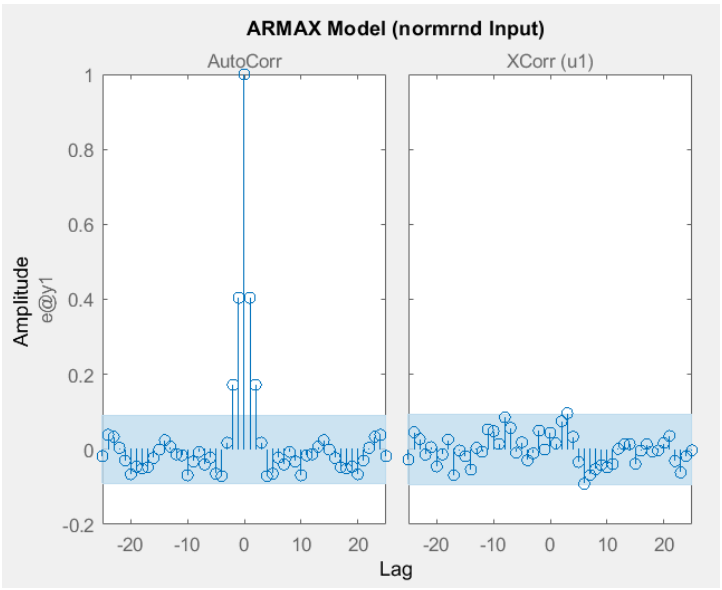

Figure 15 resid for third (normal) input for ARMAX

```
estimated_y_3_armax =
Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)
  A(z) = 1 - 1.72 z^-1 + 0.9 z^-2

  B(z) = 0.48 z^-1 - 0.48 z^-2

  C(z) = 1 - 0.4066 z^-1

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=2   nb=2   nc=1   nk=1
   Number of free coefficients: 5
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARMAX on time domain data "data_3".
Fit to estimation data: 100% (prediction focus)
FPE: 2.817e-32, MSE: 2.782e-32
```

$$poles_{armax_1} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$zeros_{armax_1} = 1$$

$$poles_{arx_1} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$zeros_{arx_1} = 1$$

$$poles_{sys} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$zeros_{sys} = 1$$

As seen again the poles and zeros of the system are not changed using these estimations. We can say in absence of the noise the inputs suffice for the estimation task.

Forth input(uniform input)

$$r = unifrnd(-1,1,800,1)$$

ARX:

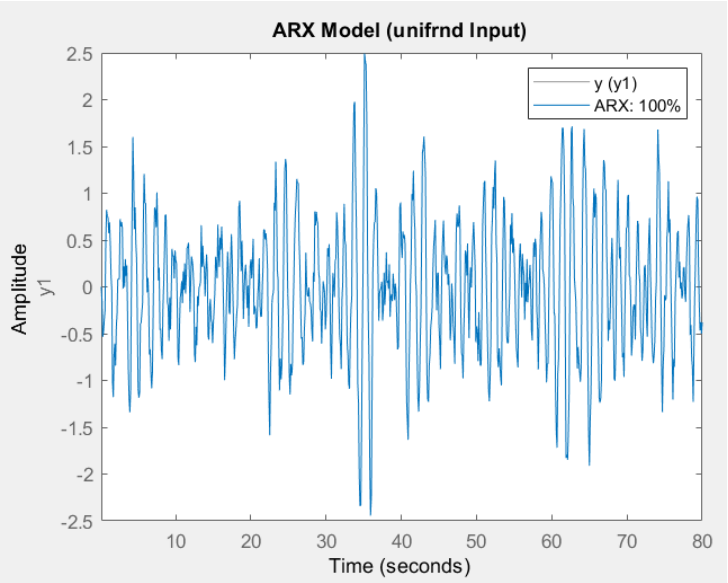

Figure 16 comparison between the ARX output and the system output for forth input

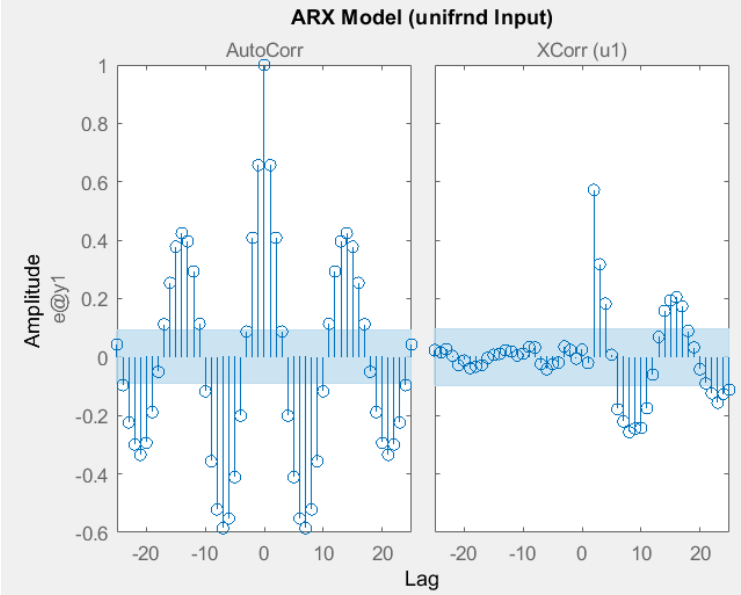

Figure 17 resid for the ARX and forth output

```
estimated_y_4_arx =
Discrete-time ARX model: A(z)y(t) = B(z)u(t) + e(t)
  A(z) = 1 - 1.72 z^-1 + 0.9 z^-2

  B(z) = 0.48 z^-1 - 0.48 z^-2

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=2   nb=2   nk=1
   Number of free coefficients: 4
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARX on time domain data "data_4".
Fit to estimation data: 100% (prediction focus)
FPE: 2.2e-31, MSE: 2.167e-31
```

ARMAX:

We carry out the estimation for the forth input, which in uniform input, on ARMAX method and investigate the results.

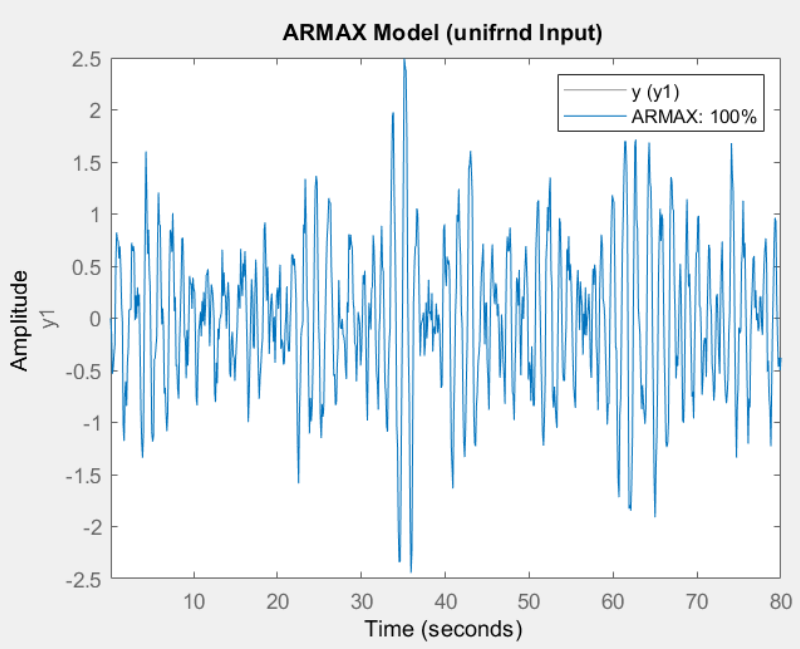

Figure 18 comparison between the ARMAX output and the system output for forth input

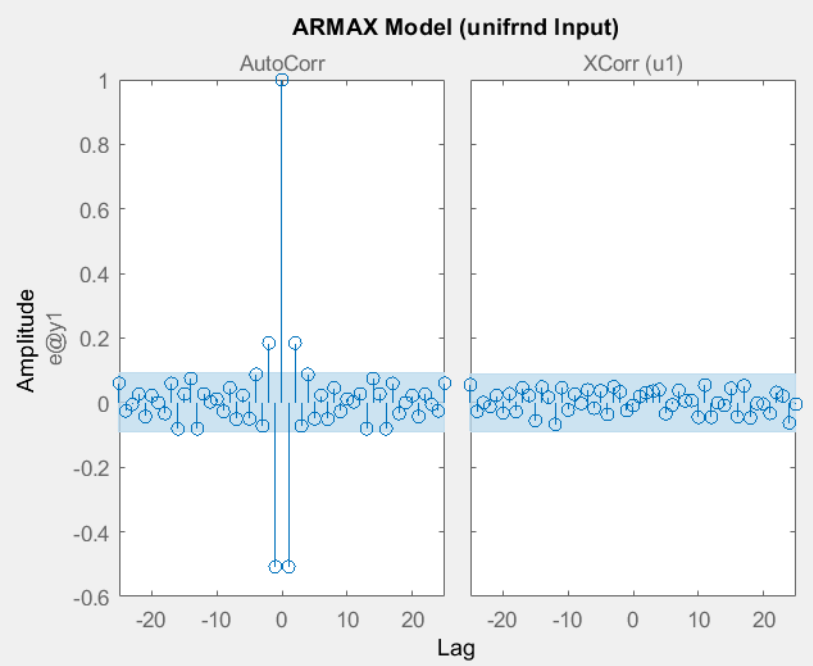

Figure 19 resid for the ARMAX and forth output

```
estimated_y_4_armax =
Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)
  A(z) = 1 - 1.72 z^-1 + 0.9 z^-2

  B(z) = 0.48 z^-1 - 0.48 z^-2

  C(z) = 1 + 0.5162 z^-1

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=2   nb=2   nc=1   nk=1
   Number of free coefficients: 5
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARMAX on time domain data "data_4".
Fit to estimation data: 100% (prediction focus)
FPE: 1.445e-32, MSE: 1.427e-32
```

As we can see the estimation managed to estimate the parameters successfully. So in order to see the estimation performance deeper, we must examine the poles and zeros from estimations and the actual system:

$$poles_{armax_1} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$zeros_{armax_1} = 1$$

$$poles_{arx_1} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$zeros_{arx_1} = 1$$

$$poles_{sys} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$zeros_{sys} = 1$$

Fortunately the poles and zeros are not different and the estimation managed to compute the poles and zeros correctly. So we can conclude that although the input has changed the results are the same, and in absence of noise the ARX and ARMAX algorithms are capable of estimating the system with very high accuracy.

And keep in mind that this is not a heavy assignment for the algorithms because we did limit the random signal to three levels and the width of the pulse to not be smaller than a certain minimum.

But there is still a small difference between ARMAX and ARX algorithms, and that is, if we pay attention for residuals of the two algorithms of the same input we see, the residuals of the ARMAX algorithm is constrained in the bound specified, which means that ARMAX algorithm has given us better estimations.

In the second case, more dynamics are in the specified bounds and this is because we decreased the width and increased number of the pulses. In both ARMAX and ARX the estimation was carried out with more accuracy because the input is richer for the estimation operation.

In the third case more dynamics are inside the bound, because the normal input has managed to excite the system better.

The forth input, uniform input, is not as good as the normal input, and from the result, one can say more dynamics are out of the bound. So the estimation is not as good as the normal input.

You can change the input signals or play up with parameters like n_a and n_b. And one last thing to mention is that the nature of the two algorithms is that we use them on closed loop systems, but we used them here on open loop system, which can degrade the performance of the estimation even more.

2)

In this part we once use the ARX identification toolbox and once we calculate the ARX using the Gradient Descent algorithm, and investigate the results.

The Gradient Descent algorithm is as follows:

Recall that we have $f: \mathbb{R}^n \to \mathbb{R}$, convex and differentiable, want to solve

$$\min_{x \in \mathbb{R}^n} f(x),$$

i.e., find $x^\star$ such that $f(x^\star) = \min f(x)$

Gradient descent: choose initial $x^{(0)} \in \mathbb{R}^n$, repeat:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f\left(x^{(k-1)}\right),\ k = 1,2,3,\ldots$$

Stop at some point(The stopping criterion)

We formulate the algorithm as below:

$$\varepsilon(t) = y - \hat{y}(t \mid t-1) = y - \theta X$$

$$X = [u, y]$$

$$\theta^+ = \theta^- - lr * \frac{\partial(y - \theta X)}{\partial A} \mid \theta = \theta^-$$

In the above formula $-2 < \theta < 2$ and we take the lr or the learning rate very small a number for example 0.0001.

First input:

The results from the Gradient Descent algorithm:

```
sys_arx_Gradient =
 
   0.48 z^-1 - 0.48 z^-2
  ------------------------
  1 - 1.72 z^-1 + 0.9 z^-2

Sample time: 0.1 seconds
Discrete-time transfer function.
```

The results from the Identification toolbox:

```
Y_arx =
Discrete-time ARX model: A(z)y(t) = B(z)u(t) + e(t)
  A(z) = 1 - 1.72 z^-1 + 0.9 z^-2

  B(z) = 0.48 z^-1 - 0.48 z^-2

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=2   nb=2   nk=1
   Number of free coefficients: 4
   Use "polydata", "getpvec", "getcov" for parameters and their unce

Status:
Estimated using ARX on time domain data "data".
Fit to estimation data: 100% (prediction focus)
FPE: 4.019e-32, MSE: 3.959e-32
```

For further comparison, we can just plot the results from GD and toolbox:

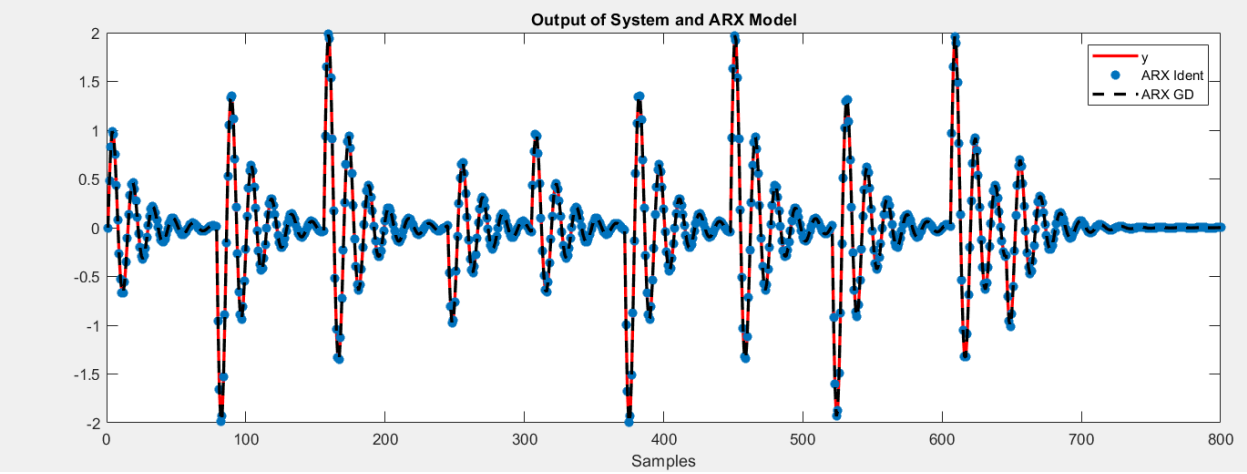

Figure 20 comparison of the ARX estimation from toolbox and gradient Descent first input

$$zeros_{arx_{ident}} = 1.0000$$

$$zeros_{arx_{Gradient}} = 1.0000$$

$$\text{poles}_{\text{arx}_{\text{ident}}} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$\text{poles}_{\text{arx}_{\text{Gradient}}} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

The results are pretty much identical, because the Identification toolbox ARX algorithm uses gradient descent as well.

Second Input:

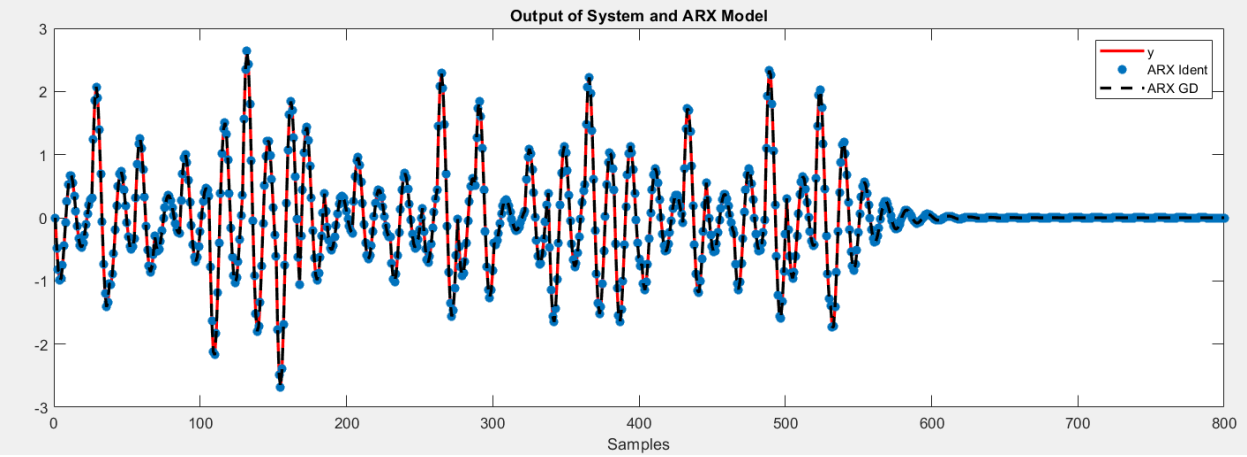

Figure 21 comparison of the ARX estimation from toolbox and gradient Descent for second input

$$zeros_{arx_{ident}} = 1.0000$$

$$zeros_{arx_{Gradient}} = 1.0000$$

$$\text{poles}_{\text{arx}_{\text{ident}}} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$poles_{arx_{Gradient}} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

Same as always, the results are identical, the plots are the same and so are the zeros and poles.

Third input:

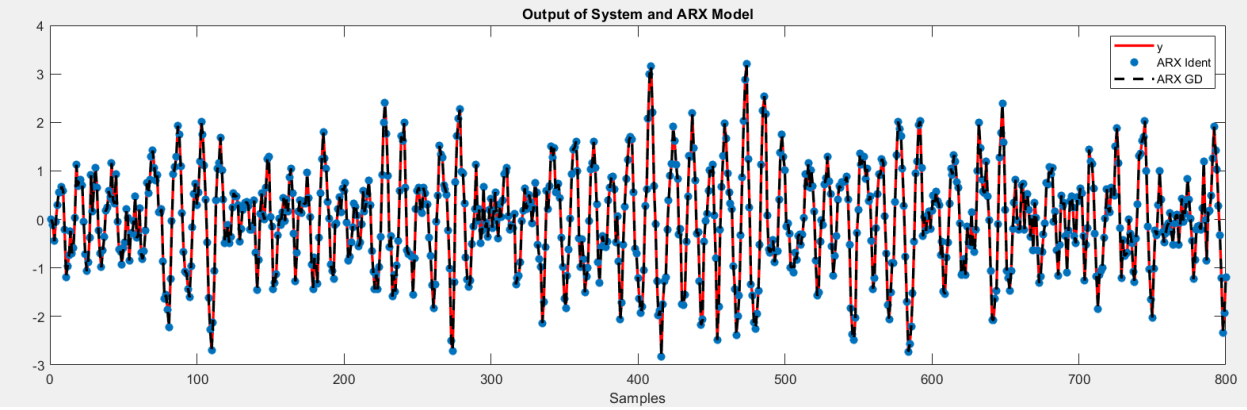

Figure 22 comparison of the ARX estimation from toolbox and gradient Descent for third input

$$zeros_{arx_{ident}} = 1.0000$$

$$zeros_{arx_{Gradient}} = 1.0000$$

$$\text{poles}_{\text{arx}_{\text{ident}}} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$\text{poles}_{\text{arx}_{\text{Gradient}}} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

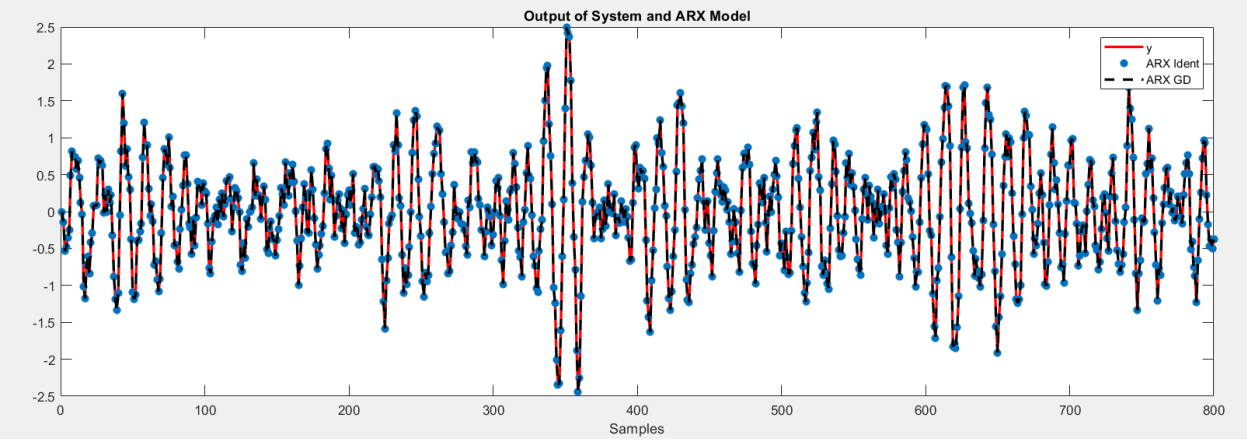

Figure 23 comparison of the ARX estimation from toolbox and gradient Descent for forth input

$$zeros_{arx_{ident}} = 1.0000$$

$$zeros_{arx_{Gradient}} = 1.0000$$

$$\text{poles}_{\text{arx}_{\text{ident}}} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

$$\text{poles}_{\text{arx}_{\text{Gradient}}} = \begin{cases} 0.8600 + 0.4005i \\ 0.8600 - 0.4005i \end{cases}$$

The results from all four inputs the same for Identification toolbox and Gradient Descent, because the toolbox uses the Gradient Descent as well.

3)

From Mason law we have:

$$y = \frac{G}{1+G}u; y = \frac{1}{1+G}v$$

We use these equations to close the loop.

First case; Low Noise $\boldsymbol{\sigma = 0.01}$

ARX:

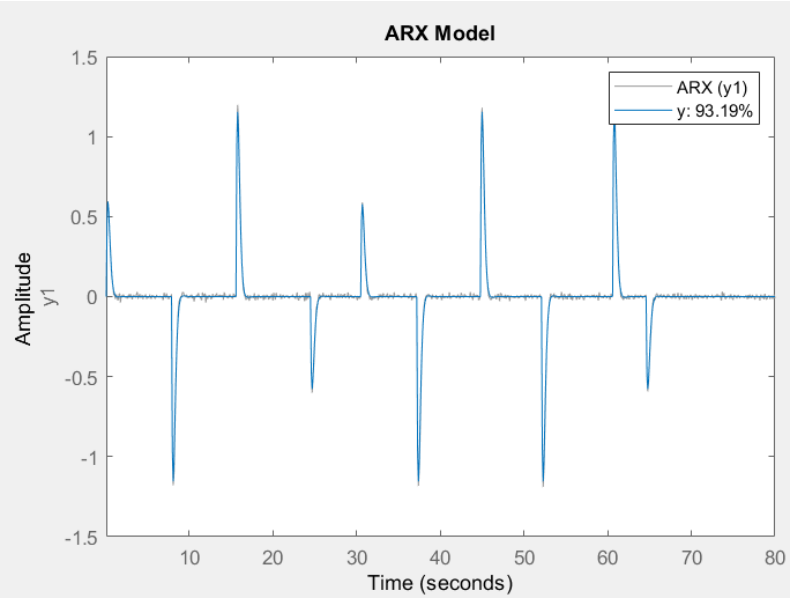

Figure 24 comparison between the ARX output and the system output for first input and low noise

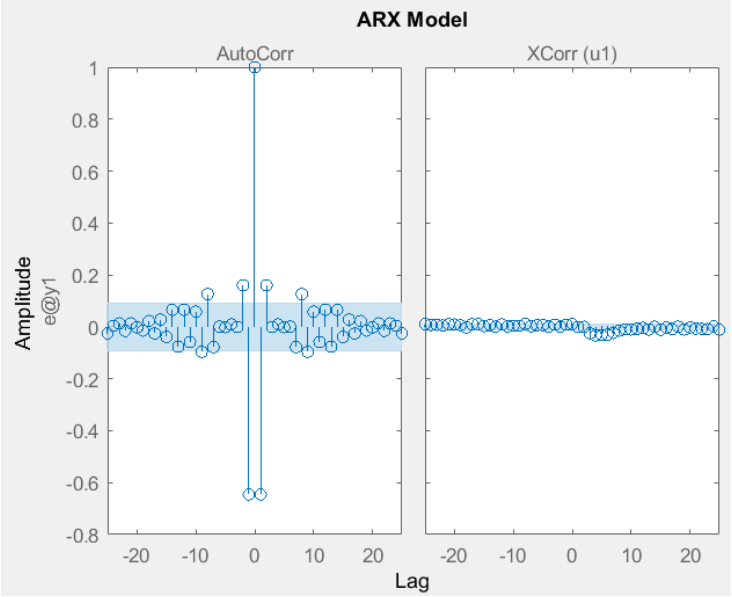

Figure 25 ARX residuals for first input and low noise

ARMAX:

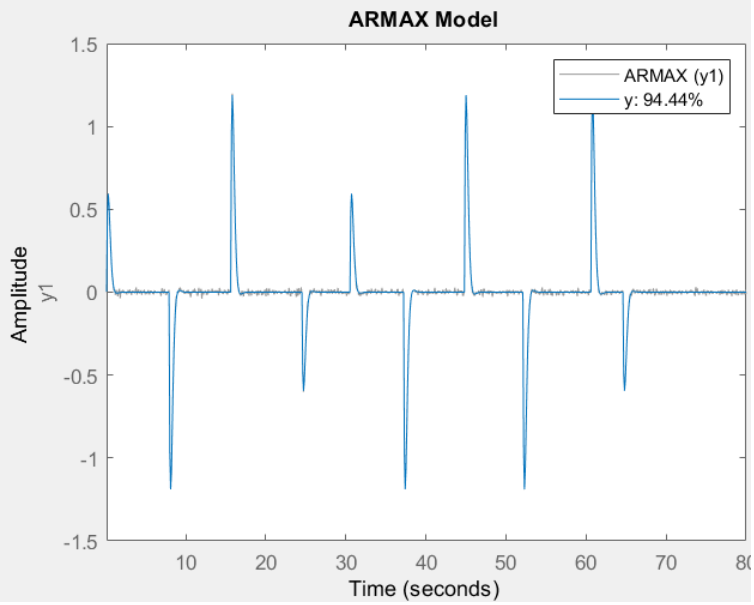

Figure 26 comparison between the ARMAX output and the system output for first input and low noise

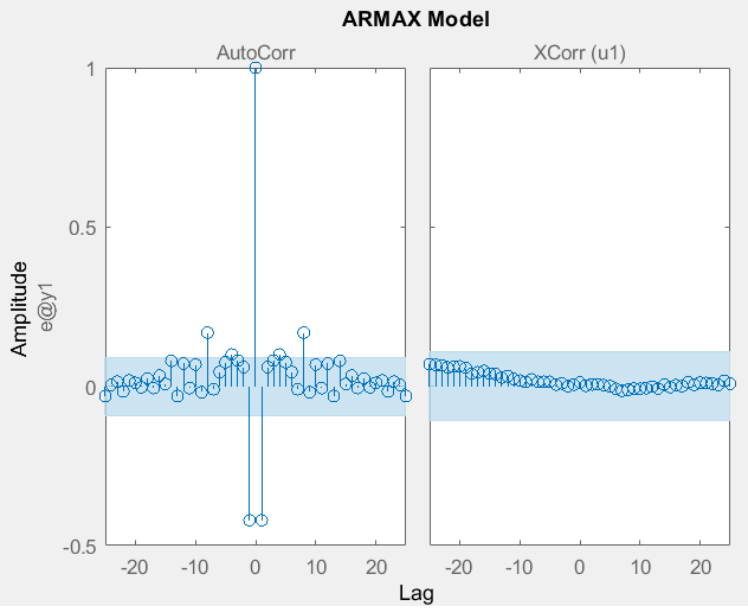

Figure 27 ARMAX residuals for first input and low noise

ARARX:

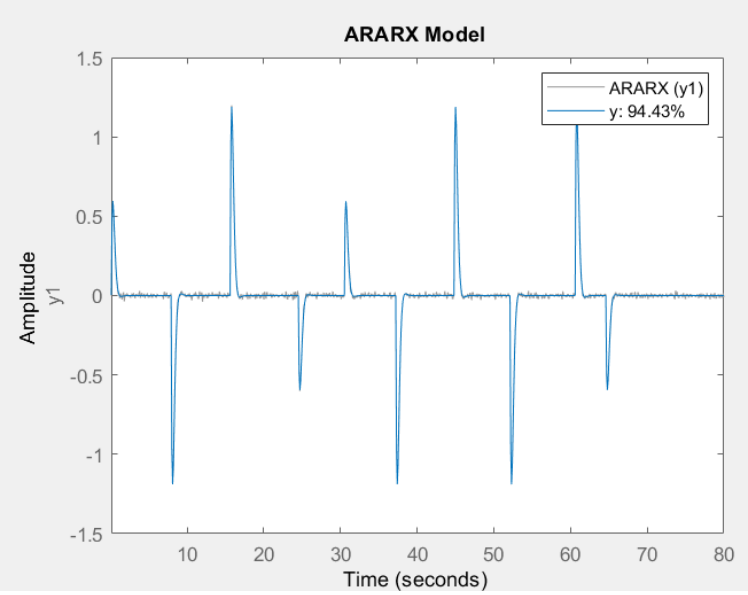

Figure 28 comparison between the ARARX output and the system output for first input and low noise

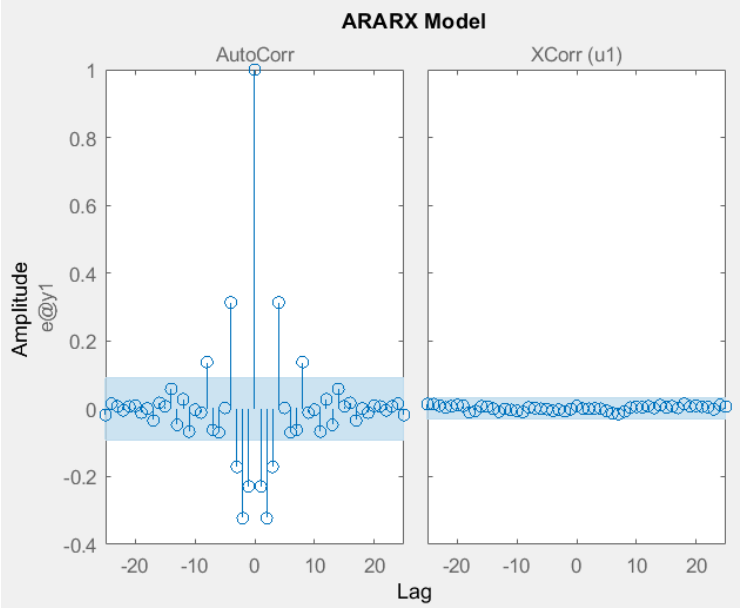

Figure 29 ARAX residuals for first input and low noise

OE:

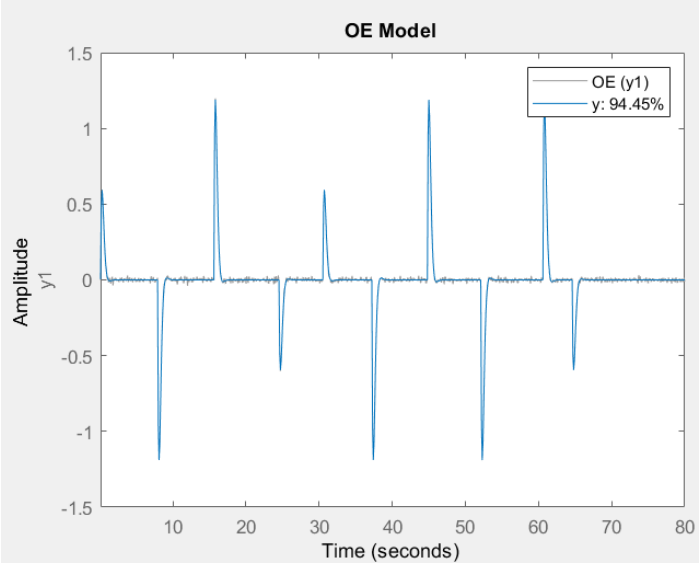

Figure 30 comparison between the OE output and the system output for first input and low noise

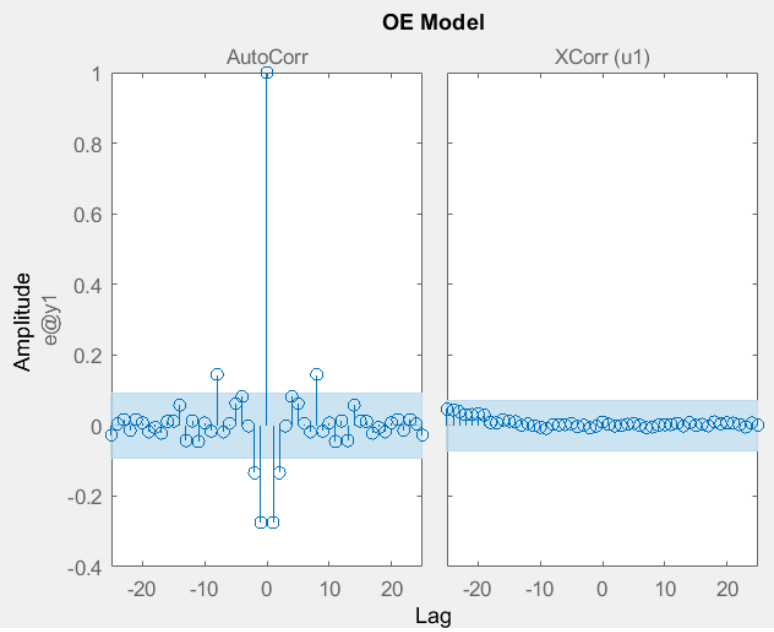

Figure 31 OE residuals for first input and low noise

BJ:

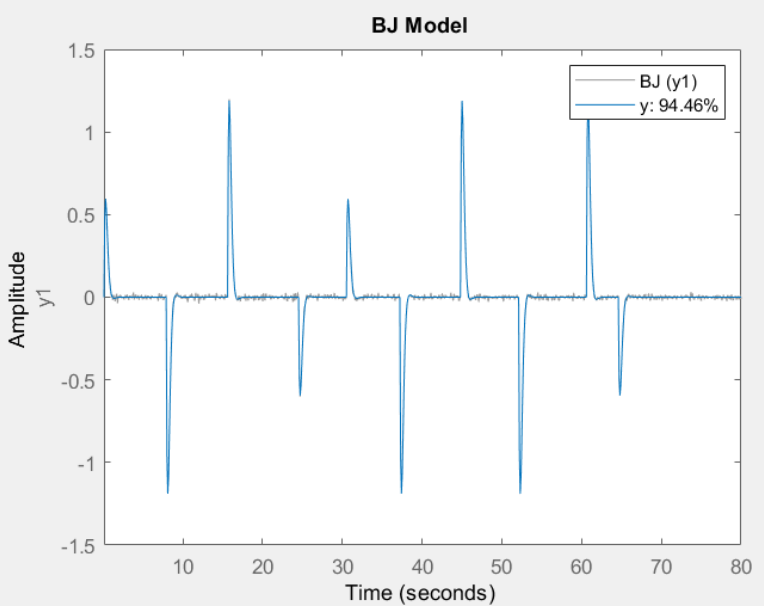

Figure 32 comparison between the BJ output and the system output for first input and low noise

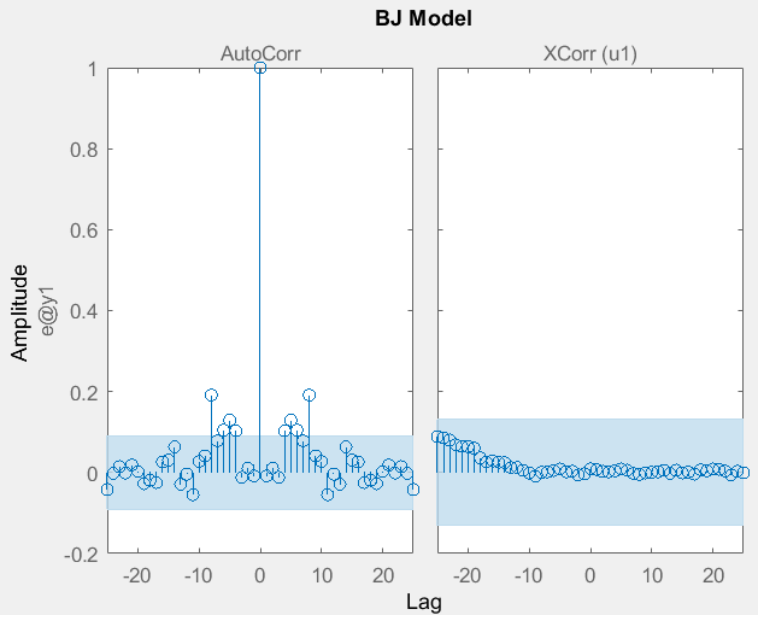

Figure 33 BJ residuals for first input and low noise

The system:

$$\text{poles} = \begin{cases} 0.6200 + 0.1887\text{i} \\ 0.6200 - 0.1887\text{i} \end{cases}$$

$$zeros = 1$$

ARX Poles, Zeros, MSE:

$$zeros\_sys_{\text{ARX}} = 1.0000$$

$$\text{poles}_{\text{sys}_{\text{ARX}}} = \begin{cases} 0.6063 + 0.1624\text{i} \\ 0.6063 - 0.1624\text{i} \end{cases}$$

$$error_{\text{ARX}} = 4.9063e-04$$

$$A(z) = 1 - 1.213\,z^{-1} + 0.394\,z^{-2}$$

$$B(z) = 0.476\,z^{-1} - 0.476\,z^{-2}$$

ARMAX Poles, Zeros, MSE:

$$zeros_{sys_{\text{ARMAX}}} = 1.0000$$

$$\text{poles}_{\text{sys}_{\text{ARMAX}}} = \begin{cases} 0.6187 + 0.1874\text{i} \\ 0.6187 - 0.1874\text{i} \end{cases}$$

$$error_{\text{ARMAX}} = 3.9944e-04$$

$$A(z) = 1 - 1.237\,z^{-1} + 0.4178\,z^{-2}$$

$$B(z) = 0.4806\,z^{-1} - 0.4806\,z^{-2}$$

$$C(z) = 1 - 0.9191\,z^{-1}$$

ARARX Poles, Zeros, MSE:

$$zeros\_sys_{\text{ARARX}} = 1.0001$$

$$\text{poles}_{\text{sys}_{\text{ARARX}}} = \begin{cases} 0.6182 + 0.1862\text{i} \\ 0.6182 - 0.1862\text{i} \end{cases}$$

$$error_{\text{ARARX}} = 3.9992e-04$$

$$A(z) = 1 - 1.236\,z^{-1} + 0.4168\,z^{-2}$$

$$B(z) = 0.4808\,z^{-1} - 0.4808\,z^{-2}$$

$$D(z) = 1 + 1.014\,z^{-1} + 0.5163\,z^{-2}$$

OE Poles, Zeros, MSE:

$$zeros_{sys_{\text{OE}}} = 1.0000$$

$$\text{poles}_{\text{sys}_{\text{OE}}} = \begin{cases} 0.6212 + 0.1909\text{i} \\ 0.6212 - 0.1909\text{i} \end{cases}$$

$$error_{OE} = 3.9846e-04$$

$$F(z) = 1 - 1.242\,z^{-1} + 0.4223\,z^{-2}$$

$$B(z) = 0.4785\,z^{-1} - 0.4786\,z^{-2}$$

BJ Poles, Zeros, MSE:

$$zeros_{sys_{BJ}} = 1.0000$$

$$\text{poles}_{\text{sys}_{BJ}} = \begin{cases} 0.6215 + 0.1915\text{i} \\ 0.6215 - 0.1915\text{i} \end{cases}$$

$error_{BJ} = 3.9849e-04$

$B(z) = 0.4783\, z^{-1} - 0.4783\, z^{-2}$

$C(z) = 1 - 0.2841\, z^{-1}$

D(z) = 1 + 0.07133 z^-1 + 0.1631 z^-2

$F(z) = 1 - 1.243\, z^{-1} + 0.4229\, z^{-2}$

In the presence of noise, the ARX has the worst performance, although all estimation algorithms have managed to retain an acceptable performance. The error is relatively small, and pole zeros have been estimated properly. OE and BJ have the most promising performance and most of the dynamics are in the defined bound.

Medium Noise $\sigma = 0.07$:

ARX:

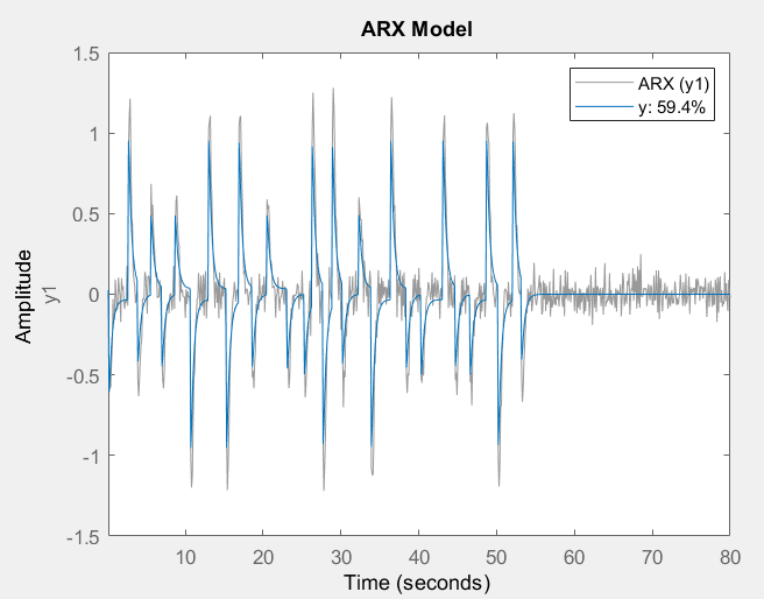

Figure 34 comparison between the ARX output and the system output for first input and Medium noise

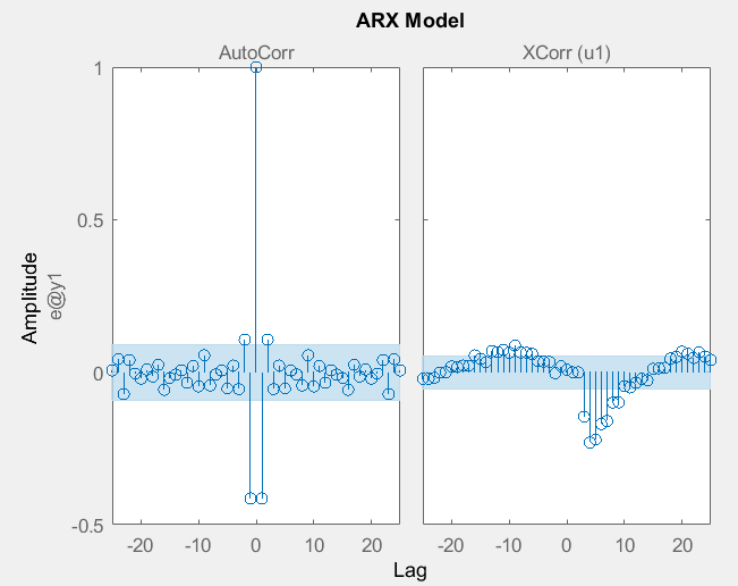

Figure 35 ARX residuals for first input and medium noise

ARMAX:

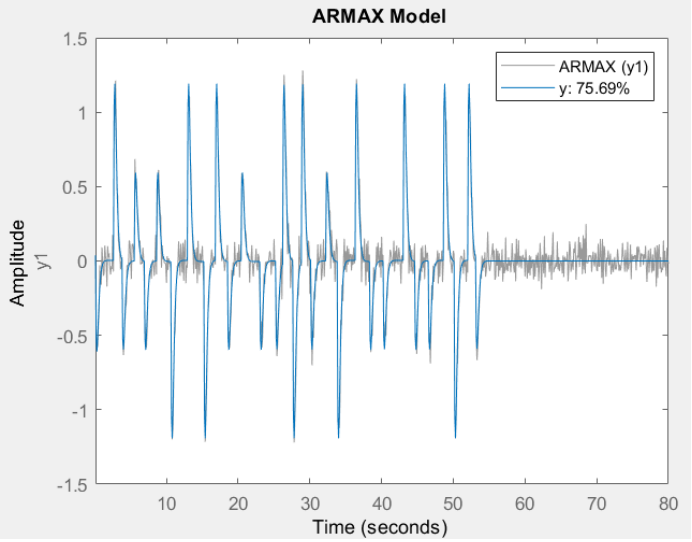

Figure 36 comparison between the ARMAX output and the system output for first input and Medium noise

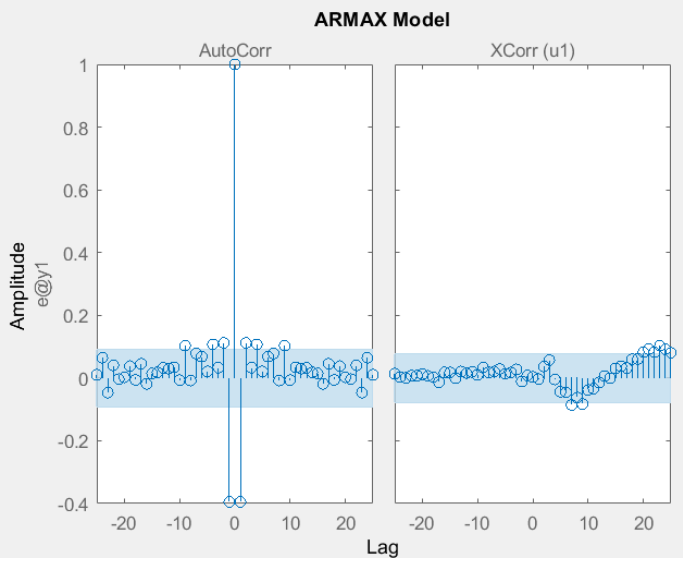

Figure 37 ARMAX residuals for first input and medium noise

ARARX:

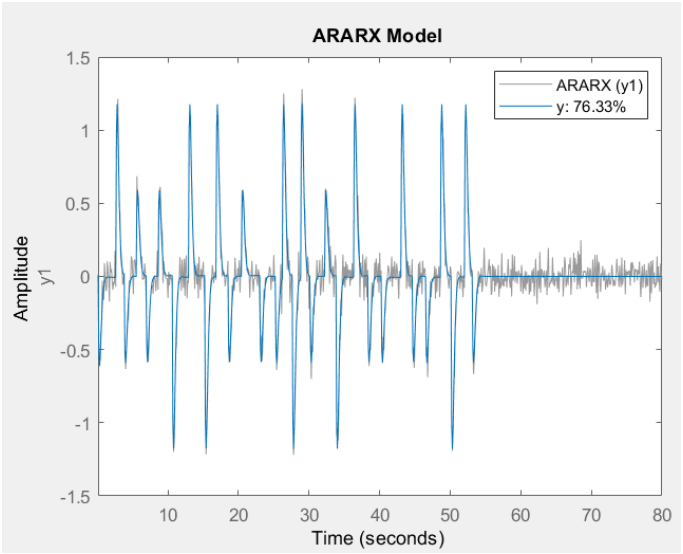

Figure 38 comparison between the ARARX output and the system output for first input and Medium noise

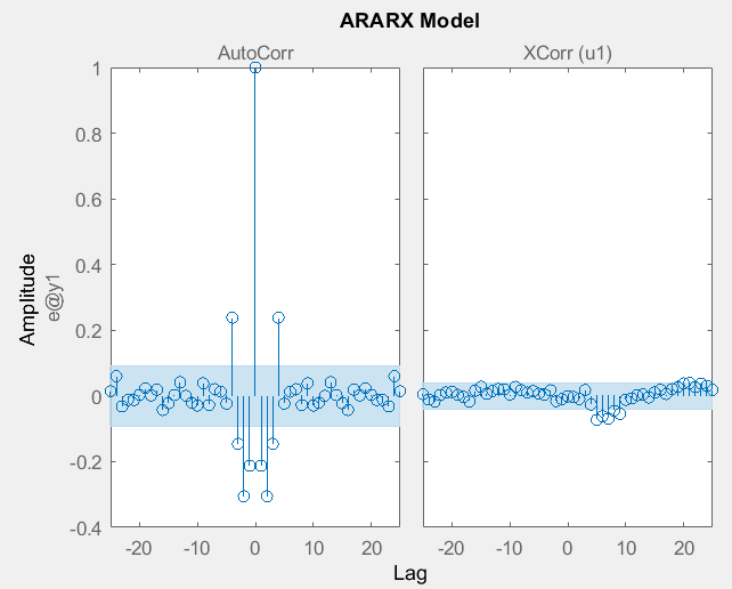

Figure 39 ARARX residuals for first input and medium noise

OE:

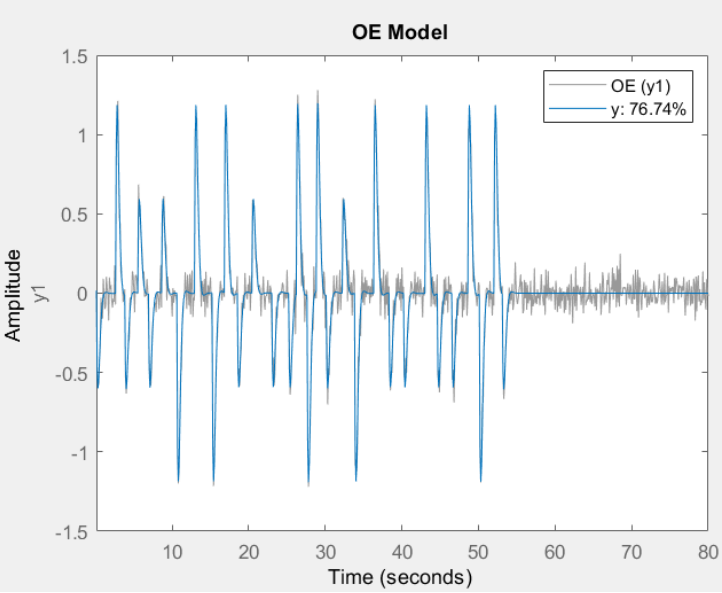

Figure 40 comparison between the OE output and the system output for first input and Medium noise

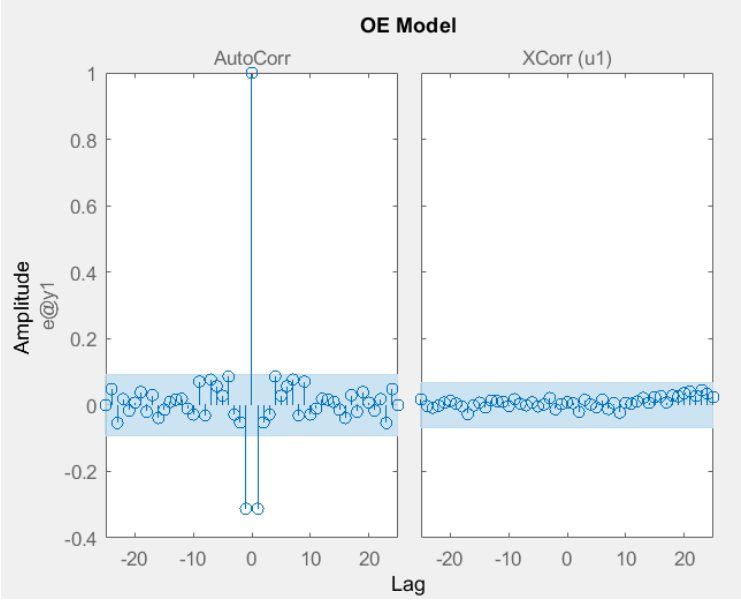

Figure 41 OE residuals for first input and medium noise

BJ:

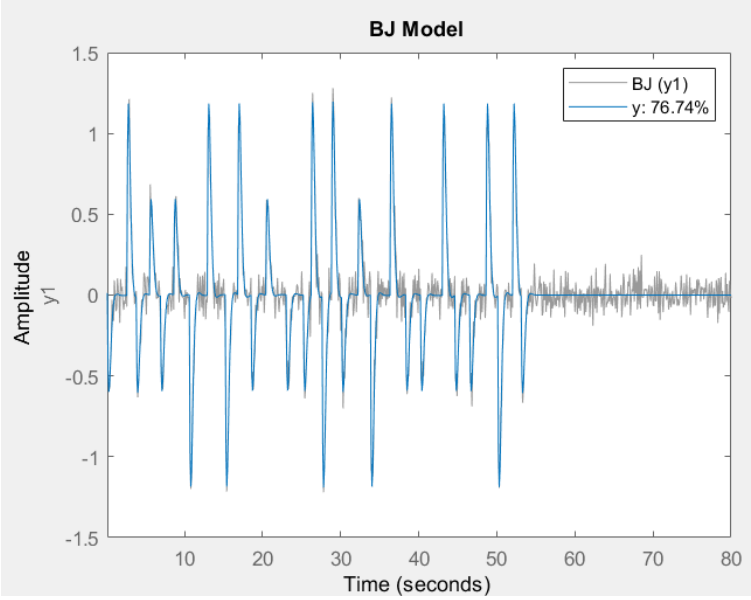

Figure 42 comparison between the BJ output and the system output for first input and Medium noise

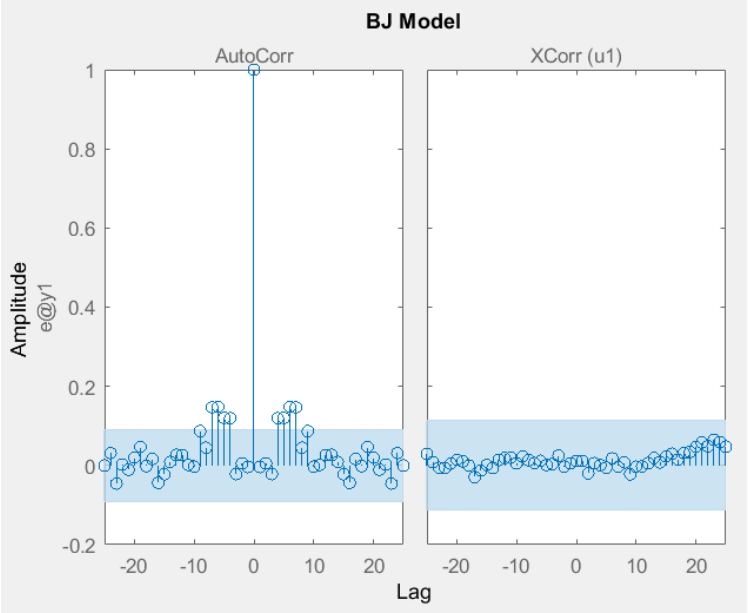

Figure 43 BJ residuals for first input and medium noise

The system:

$zeros = 1$

$$\text{poles} = \begin{cases} 0.6200 + 0.1887\text{i} \\ 0.6200 - 0.1887\text{i} \end{cases}$$

```
csys =

    0.48 z^-1 - 0.48 z^-2
  -------------------------
  1 - 1.24 z^-1 + 0.42 z^-2

Sample time: 0.1 seconds
Discrete-time transfer function.
```

ARX Poles, Zeros, MSE:

$$zeros_{sys_{arx}} = 0.9857$$

$$poles_{sys_{arx}} = \begin{cases} 0.7649 \\ 0.0904 \end{cases}$$

$$error\_arx = 0.0048$$

$$A(z) = 1 - 0.8553\, z^{-1} + 0.06913\, z^{-2}$$

$$B(z) = 0.4937\, z^{-1} - 0.4866\, z^{-2}$$

ARMAX Poles, Zeros, MSE:

$zeros_{sys_{armax}} = 1.0014$

$poles_{sys_{armax}} = \begin{cases} 0.5719 + 0.0942\text{i} \\ 0.5719 - 0.0942\text{i} \end{cases}$

$error\_armax = 0.0029$

$A(z) = 1 - 1.144\, z^{-1} + 0.3359\, z^{-2}$

$B(z) = 0.5199\, z^{-1} - 0.5206\, z^{-2}$

$C(z) = 1 - 0.8888\, z^{-1}$

ARARX Poles, Zeros, MSE:

$zeros_{sys_{ararx}} = 0.9977$

$poles_{sys_{ararx}} = \begin{cases} 0.5952 + 0.1592\text{i} \\ 0.5952 - 0.1592\text{i} \end{cases}$

$error_{ararx} = 0.0028$

$A(z) = 1 - 1.19\, z^{-1} + 0.3796\, z^{-2}$

$B(z) = 0.4958\, z^{-1} - 0.4947\, z^{-2}$

$D(z) = 1 + 1.041\, z^{-1} + 0.4948\, z^{-2}$

OE Poles, Zeros, MSE:

$zeros_{sys_{OE}} = 0.9993$

$poles_{sys_{OE}} = \begin{cases} 0.6228 + 0.1976\text{i} \\ 0.6228 - 0.1976\text{i} \end{cases}$

$error_{OE} = 0.0027$

$B(z) = 0.4763\, z\text{^} - 1 - 0.476\, z\text{^} - 2$

$F(z) = 1 - 1.246\, z\text{^} - 1 + 0.4269\, z\text{^} - 2$

BJ Poles, Zeros, MSE:

$$zeros_{sys_{BJ}} = 0.9990$$

$$poles_{sys_{BJ}} = \begin{cases} 0.6251 + 0.2015\text{i} \\ 0.6251 - 0.2015\text{i} \end{cases}$$

$$error_{BJ} = 0.0027$$

$$B(z) = 0.4743\, z\text{^} - 1 - 0.4739\, z\text{^} - 2$$

$$C(z) = 1 - 0.2942\, z\text{^} - 1$$

$$D(z) = 1 + 0.08421\, z\text{^} - 1 + 0.08913\, z\text{^} - 2$$

$$F(z) = 1 - 1.25\, z\text{^} - 1 + 0.4314\, z\text{^} - 2$$

ARX has the least accuracy. It failed to estimate the imaginary component of the pole and it has bigger MSE. Best algorithms are OE and BJ. They performed the pole and zero estimation with excellent accuracy and the mean square error is smallest in them.

High Noise $\sigma = 0.12$:

ARX:

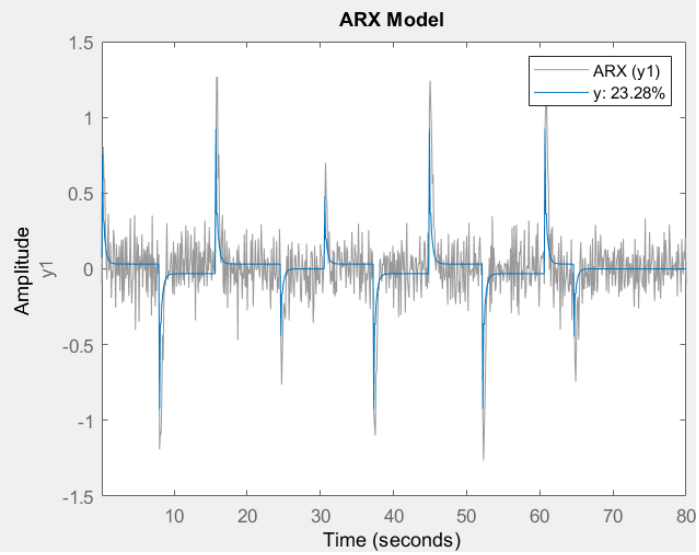

Figure 44 comparison between the ARX output and the system output for first input and High noise

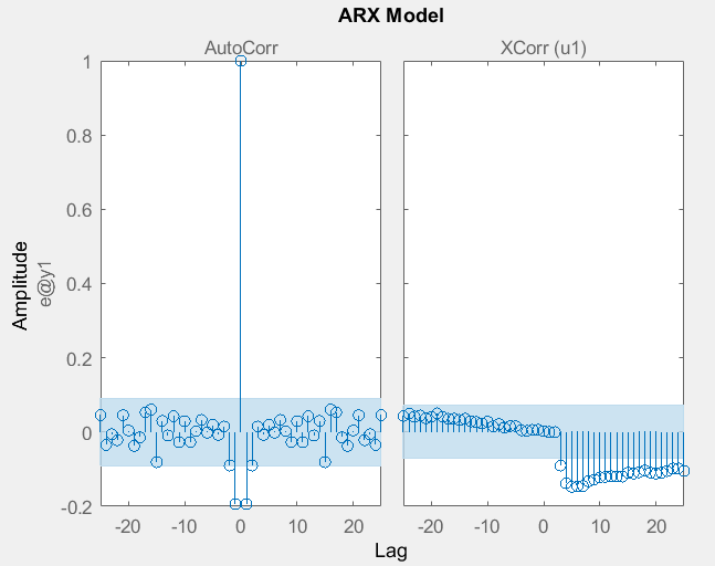

Figure 45 residuals for first input and High noise

ARMAX:

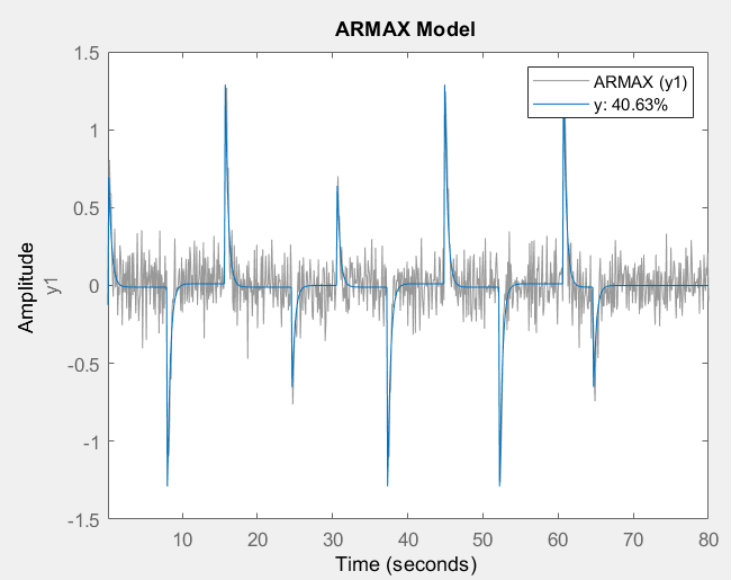

Figure 46 comparison between the ARMAX output and the system output for first input and High noise

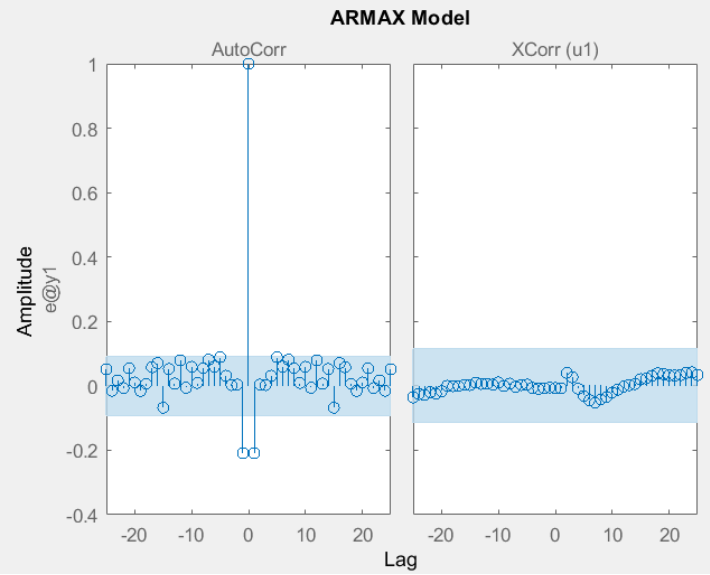

Figure 47 residuals for first input and High noise

ARARX:

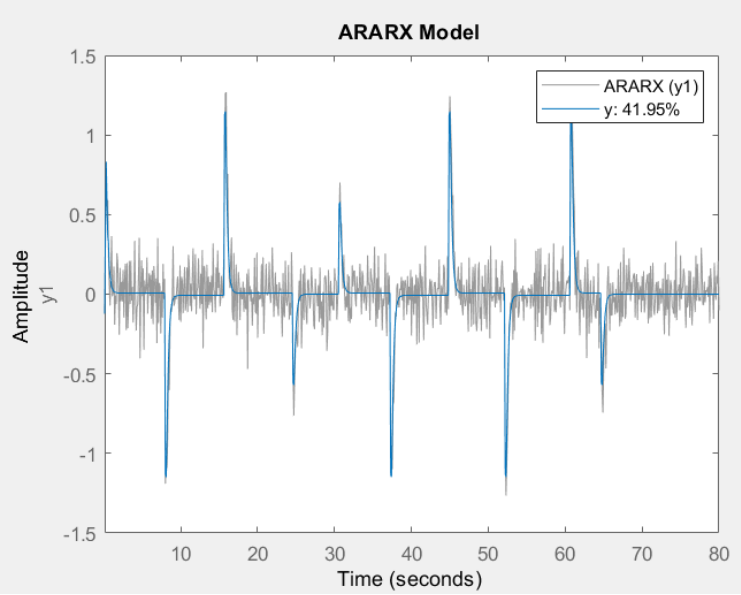

Figure 48 comparison between the ARARX output and the system output for first input and High noise

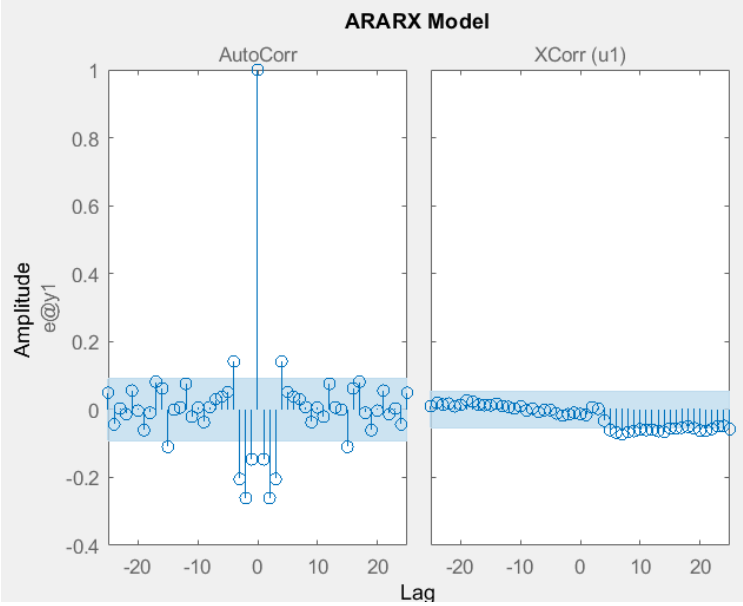

Figure 49 residuals for first input and High noise

OE:

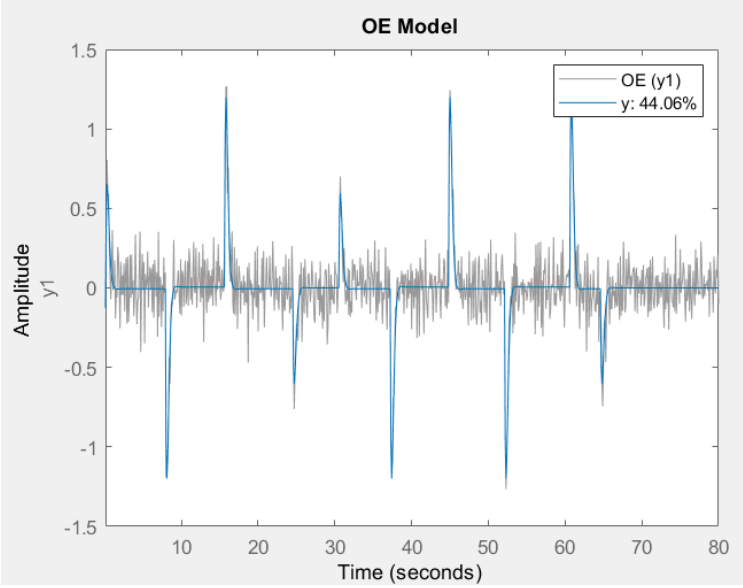

Figure 50 comparison between the OE output and the system output for first input and High noise

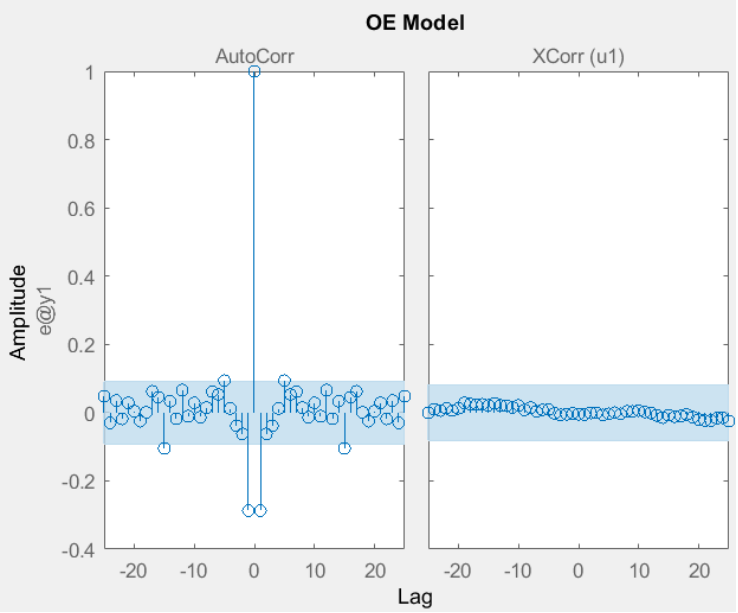

Figure 51 residuals for first input and High noise

BJ:

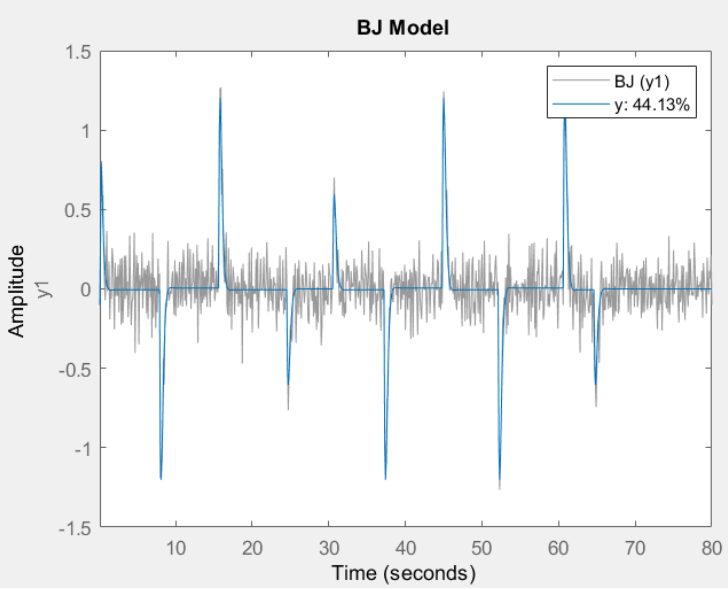

Figure 52 comparison between the BJ output and the system output for first input and High noise

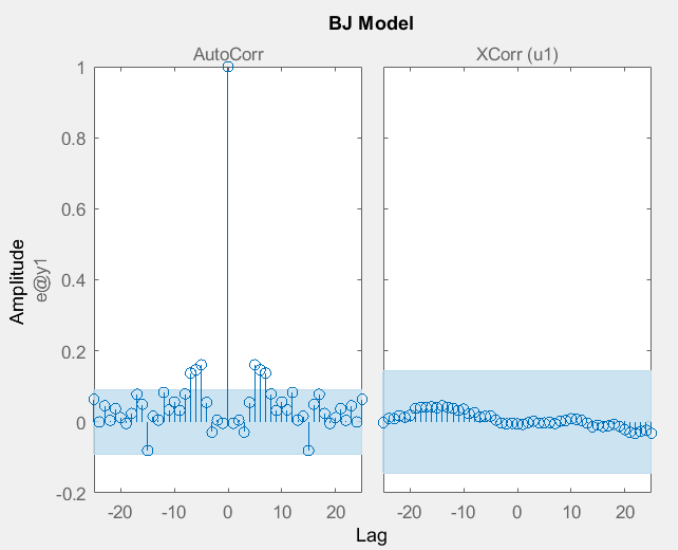

Figure 53 residuals for first input and High noise

System:

$zeros = 1$

$$\text{poles} = \begin{cases} 0.6200 + 0.1887\text{i} \\ 0.6200 - 0.1887\text{i} \end{cases}$$

```
  0.48 z^-1 - 0.48 z^-2
------------------------
1 - 1.24 z^-1 + 0.42 z^-2
```

ARX:

$$zeros_{sys_{arx}} = 0.9744$$

$$poles_{sys_{arx}} = \begin{cases} 0.7119 \\ -0.3279 \end{cases}$$

$$error_{arx} = 0.0067$$

$$A(z) = 1 - 0.384\, z^{-1} - 0.2335\, z^{-2}$$

$$B(z) = 0.4783\, z^{-1} - 0.4661\, z^{-2}$$

ARMAX:

$zeros_{sys_{armax}} = 1.0036$

$poles_{sys_{armax}} = \begin{cases} 0.7679 \\ 0.0797 \end{cases}$

$error_{armax} = 0.0052$

$A(z) = 1 - 0.8476\, z^{-1} + 0.06122\, z^{-2}$

$B(z) = 0.6393\, z^{-1} - 0.6416\, z^{-2}$

$C(z) = 1 - 0.8612\, z^{-1}$

ARARX:

$zeros_{sys_{ararx}} = 0.9970$

$poles_{sys_{ararx}} = \begin{cases} 0.6087 \\ 0.4076 \end{cases}$

$error_{ararx} = 0.0051$

$A(z) = 1 - 1.016\, z^{-1} + 0.2481\, z^{-2}$

$B(z) = 0.5669\, z^{-1} - 0.5652\, z^{-2}$

$D(z) = 1 + 0.9593\, z^{-1} + 0.4528\, z^{-2}$

OE:

$zeros_{sys_{OE}} = 1.0019$

$poles_{sys_{OE}} = \begin{cases} 0.6131 + 0.1601\mathrm{i} \\ 0.6131 - 0.1601\mathrm{i} \end{cases}$

$error_{oe} = 0.0049$

$B(z) = 0.4876\, z^{-1} - 0.4885\, z^{-2}$

$F(z) = 1 - 1.226\, z^{-1} + 0.4015\, z^{-2}$

BJ:

$zeros_{sys_{BJ}} = 1.0020$

$poles_{sys_{bj}} = \begin{cases} 0.6068 + 0.1479i \\ 0.6068 - 0.1479i \end{cases}$

$error_{bj} = 0.0049$

$B(z) = 0.4935\, z^{-1} - 0.4945\, z^{-2}$

$C(z) = 1 - 0.3718\, z^{-1}$

$D(z) = 1 - 0.02227\, z^{-1} + 0.07221\, z^{-2}$

$F(z) = 1 - 1.214\, z^{-1} + 0.3901\, z^{-2}$

Almost all algorithm have problem with this amount of noise(High Noise). ARX, ARARX and ARMAX did not succeed in estimating the poles of the system, but OE and BJ algorithms give better results for poles and zeros. In comparison to low and medium noise, the estimations have bigger errors. It's significant that the OE method yields o better results than ARX, ARMAX and ARARX, although it is very simple.

It's seen that for ARX model in presence of noise the accuracy is reduced and the parameters are estimated with error. The bigger the variance of the noise, the worse off the estimation. And the position of the poles and zeros are different than the actual poles and zeros. So we can conclude that in presence of noise, ARX model is not suitable.

In ARMAX in the presence of noise the, when the variance is low the estimation is carried out with relative accuracy, but the poles and zeros will be a little off.

The ARARX has a worse performance in comparison to ARMAX, but this model is a great tool dealing with colored noise.

OE model yields to parameters closer to the actual values of the system and it gives a better estimation of the position of poles and zeros.

BJ model in presence of noise with high variance has the best performance among all models, and the parameters will be close to their actual values. And the poles and zeros will be close to the actual poles and zeros.

Question 2

We use one model at a time to reach a accuracy of 70%, then we choose the simplest of all models.

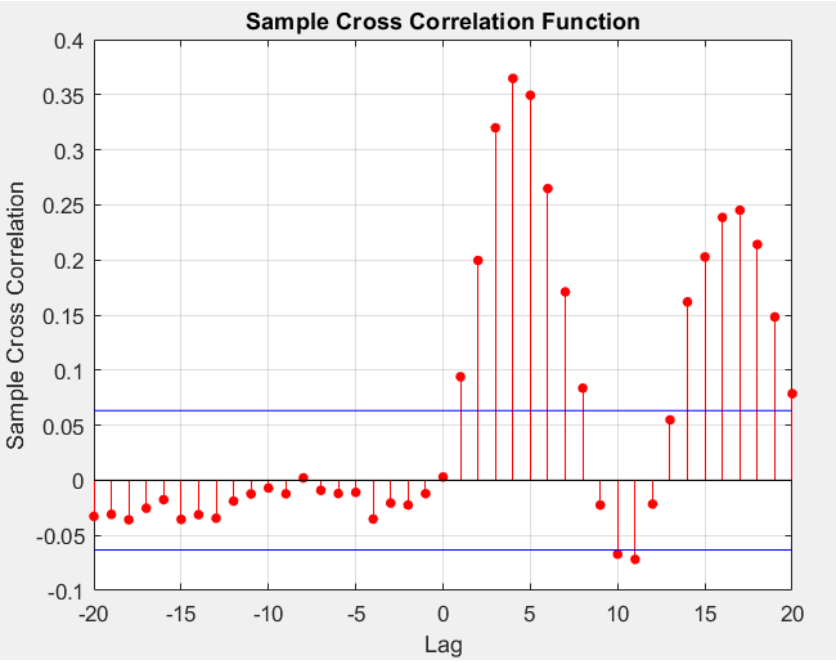

Figure 54 The cross correlation of Samples

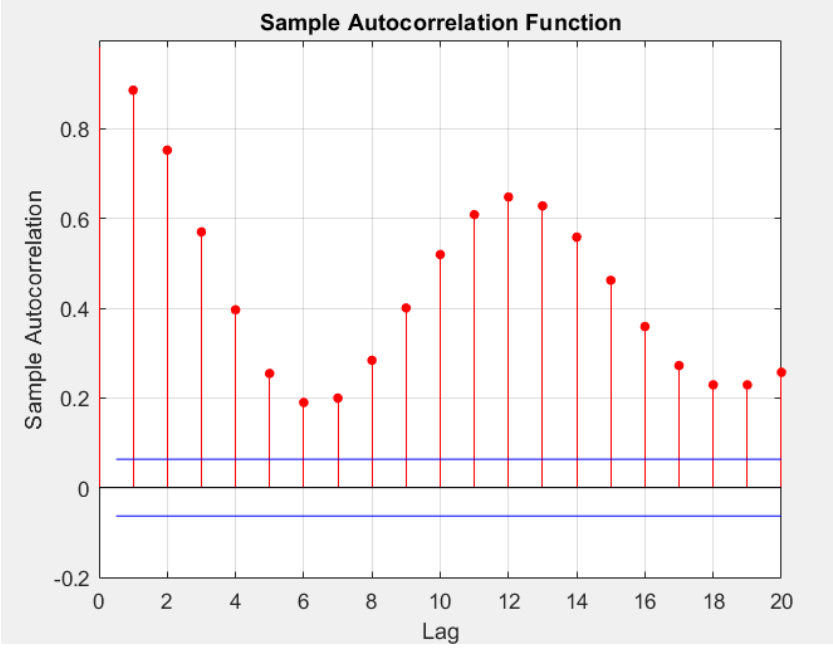

Figure 55 The Autocorrelation plot of the Samples.

These plots, do not follow a distinct and obvious pattern, so we cannot use these tools (Autocorrelation and Cross correlation). The first approach is trial and error.

Then we can use the below rule:

$$n_b = n_a - 1; n_c < n_d; n_a \approx n_f$$

$$n_d = n_a \ ; \ n_d > n_f$$

One other option could be looping a for around the algorithm to record the values n_a, n_b, n_c and the accuracy percentage

We create different models for the system:

ARX:

For 70% we will have:

$$n_a = 12$$

$$n_b = 12$$

On the test data we will have:

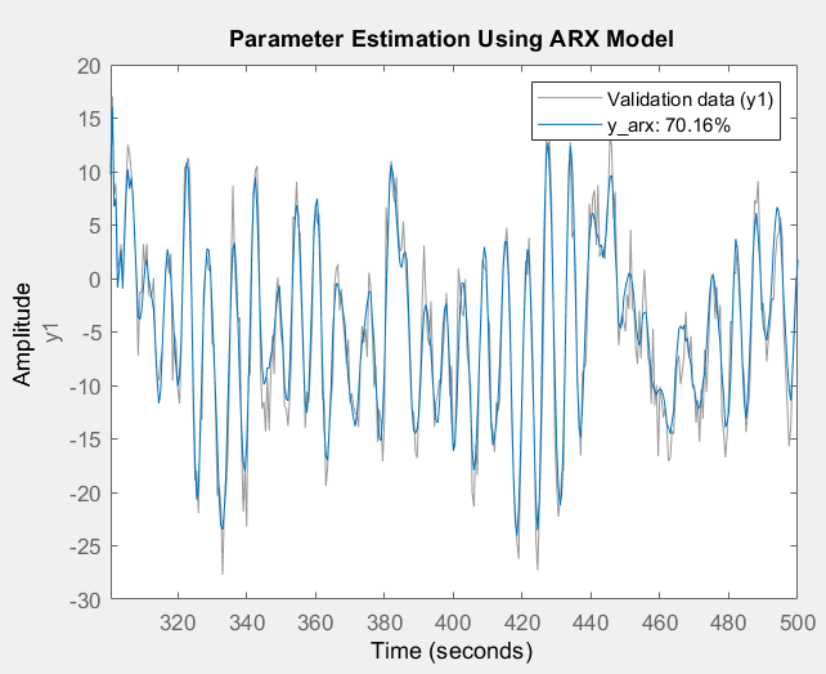

Figure 56 the output of the algorithm using the test data

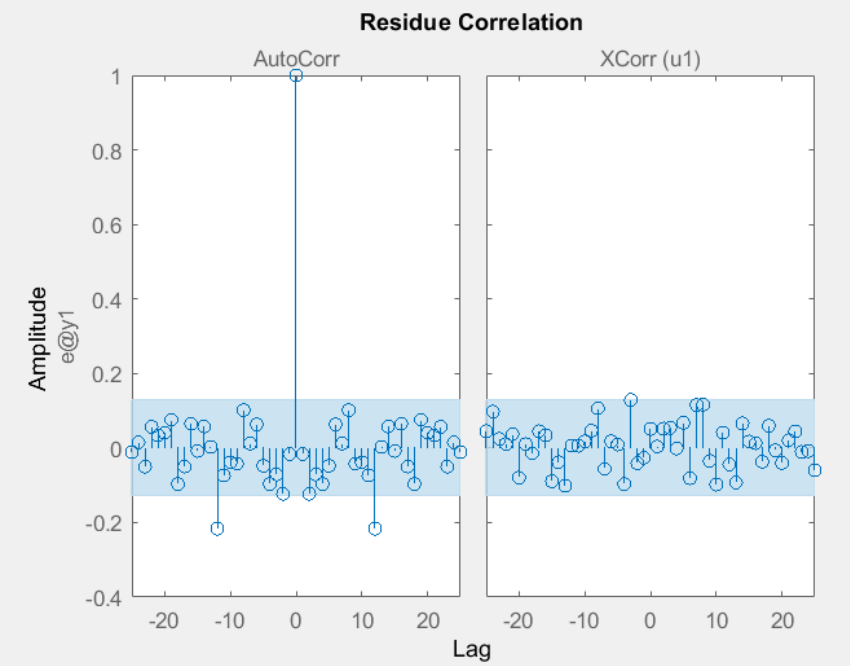

Figure 57 residuals of the ARX algorithm

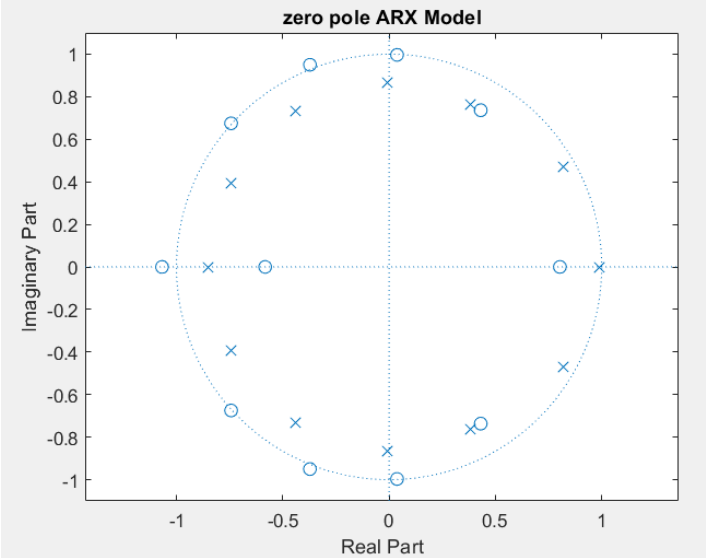

Figure 58 poles and zeros of ARX model

```
sys_arx =

    0.7226 z^-1 + 1.548 z^-2 + 2.346 z^-3 + 2.474 z^-4 + 1.894 z^-5 + 1.062 z^-6 + 0.1191 z^
                                -7 - 0.3368 z^-8 - 0.7409 z^-9 - 0.6407 z^-10 - 0.6422 z^-11 - 0.2746 z^-12
  ----------------------------------------------------------------------------------------------------------
  1 - 0.1553 z^-1 - 0.1485 z^-2 - 0.0673 z^-3 - 0.1098 z^-4 + 0.01756 z^-5 + 0.0439 z^-6 + 0.04653 z^-7
                              - 0.0172 z^-8 - 0.09161 z^-9 - 0.09008 z^-10 - 0.1284 z^-11 - 0.2107 z^-12


Sample time: 0.5 seconds
Discrete-time transfer function.
```

ARMAX

$$n_a = 3; n_b = 3; n_k = 1$$

On the test data:

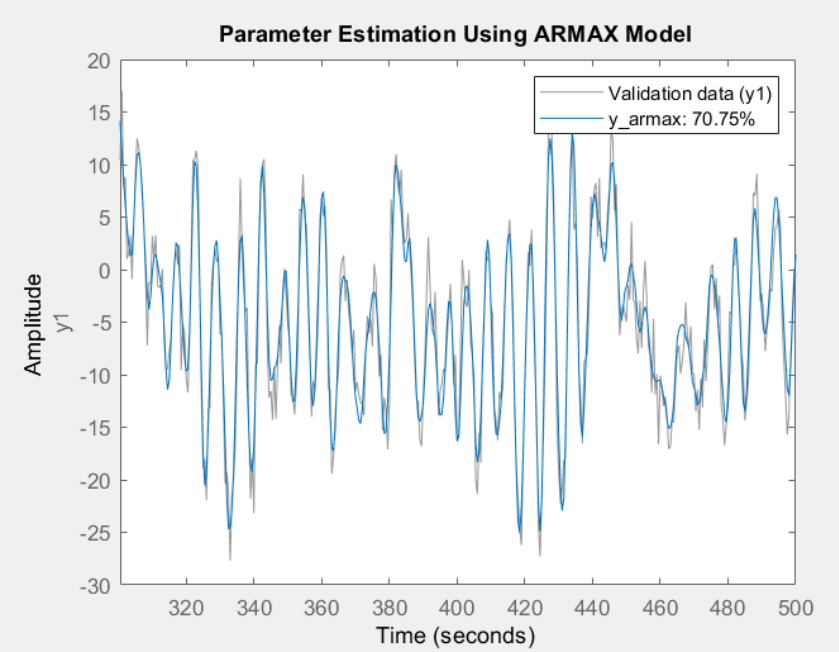

Figure 59 the output of the algorithm using the test data

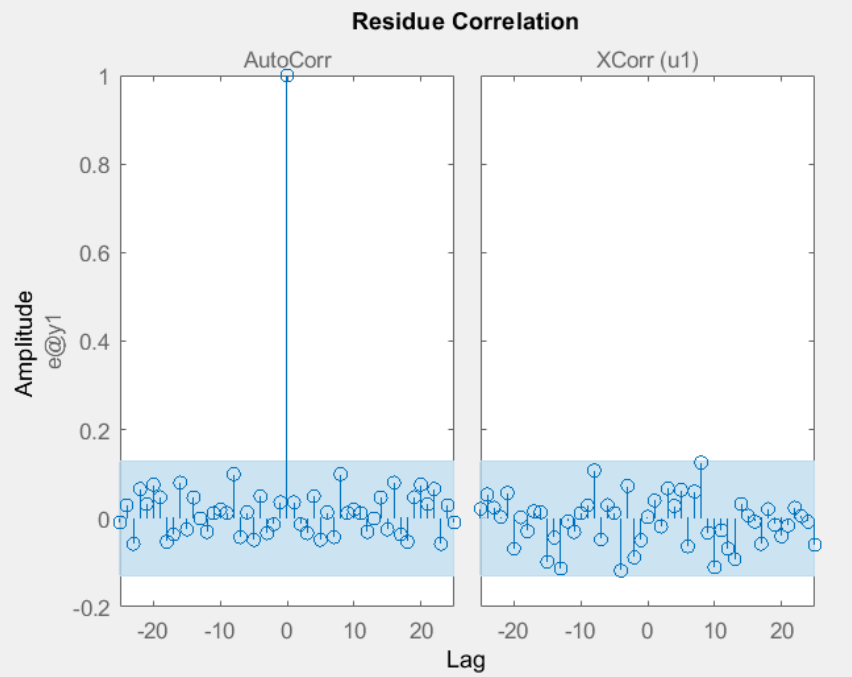

Figure 60 residuals of the ARMAX algorithm

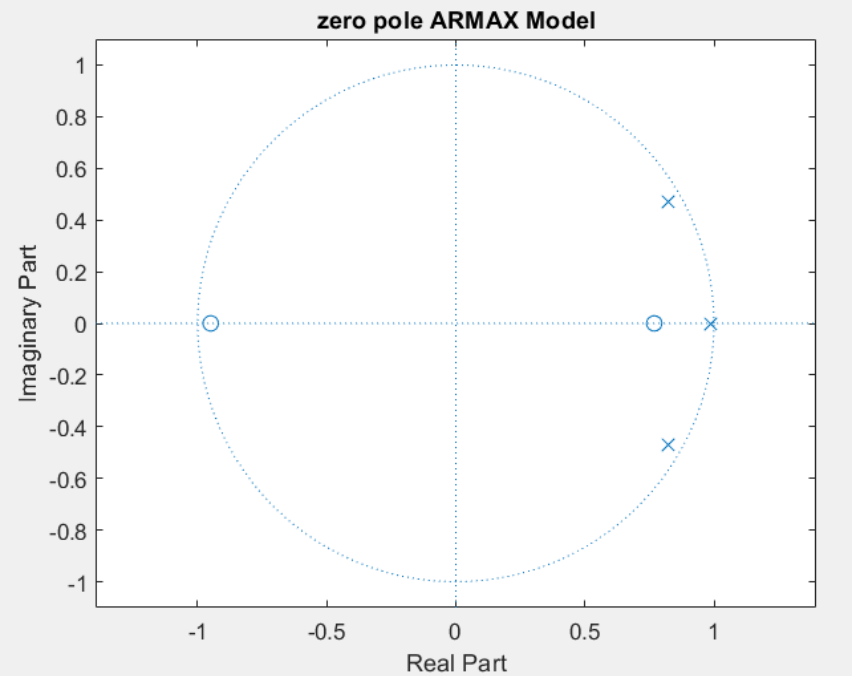

Figure 61 poles and zeros of ARMAX model

```
y_armax =
Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)
  A(z) = 1 - 2.637 z^-1 + 2.528 z^-2 - 0.8876 z^-3

  B(z) = 0.6649 z^-1 + 0.1202 z^-2 - 0.4849 z^-3

  C(z) = 1 - 2.658 z^-1 + 2.567 z^-2 - 0.9095 z^-3

Sample time: 0.5 seconds

Parameterization:
   Polynomial orders:   na=3   nb=3   nc=3   nk=1
   Number of free coefficients: 9
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARMAX on time domain data.
Fit to estimation data: 71.8% (prediction focus)
FPE: 6.856, MSE: 6.653
```

$$n_A = 5; n_b = 5; n_D = 7; n_k = 1$$

ARARX:

$$n_A = 5; n_B = 5; n_D = 7; n_k = 1$$

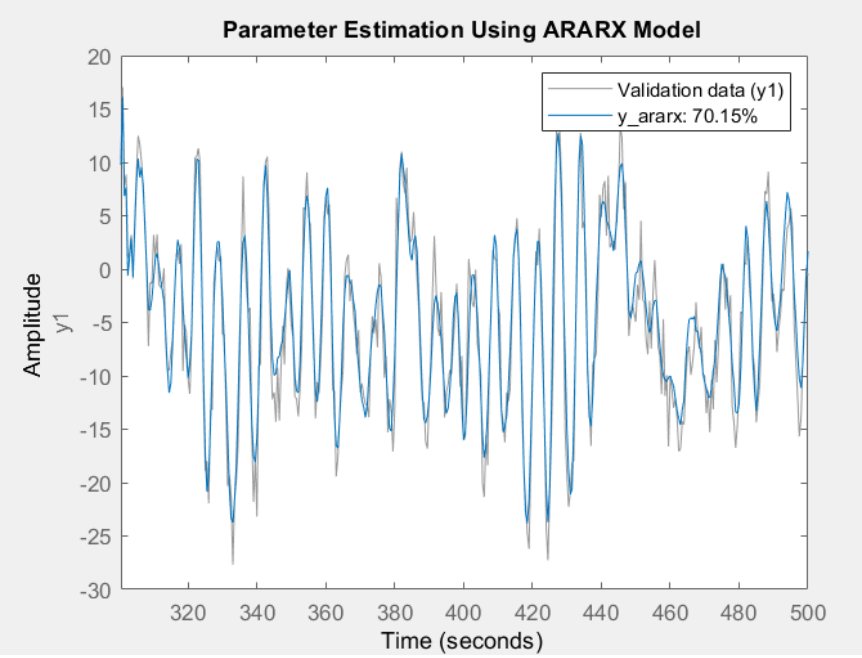

Figure 62 the output of the algorithm using the test data

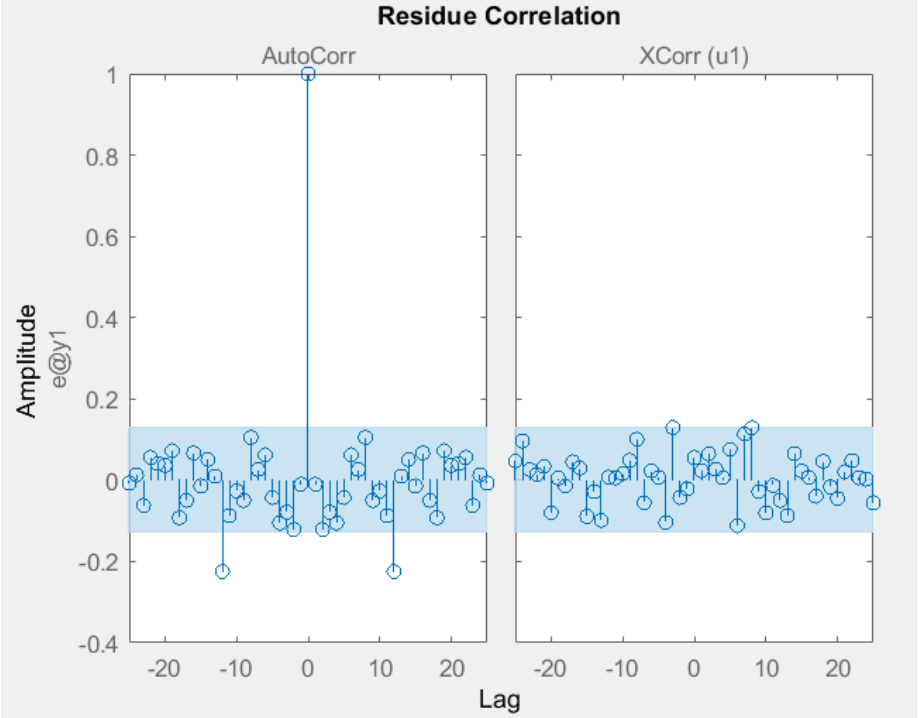

Figure 63 residuals of the ARARX algorithm

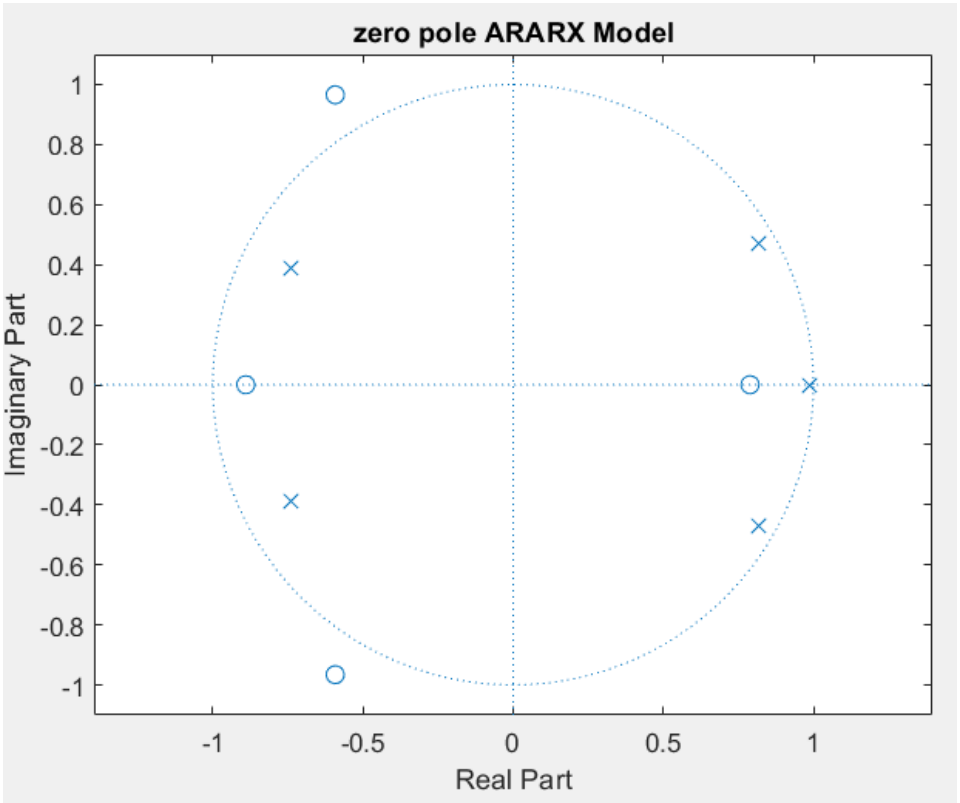

Figure 64 poles and zeros of ARMAX model

```
y_ararx =
Discrete-time ARARX model: A(z)y(t) = B(z)u(t) + [1/D(z)]e(t)
  A(z) = 1 - 1.14 z^-1 - 0.6868 z^-2 + z^-3 + 0.4559 z^-4 - 0.6179 z^-5

  B(z) = 0.6669 z^-1 + 0.8573 z^-2 + 0.4672 z^-3 - 0.4679 z^-4 - 0.6007 z^-5


  D(z) = 1 + 0.9841 z^-1 + 1.662 z^-2 + 1.501 z^-3 + 1.301 z^-4 + 1.043 z^-5 + 0.472 z^-6 + 0.3474 z^-7

Sample time: 0.5 seconds

Parameterization:
   Polynomial orders:   na=5   nb=5   nd=7   nk=1
   Number of free coefficients: 17
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using PEM on time domain data.
Fit to estimation data: 70.36% (prediction focus)
FPE: 7.78, MSE: 7.351
```

BJ:

$$n_F = 3; n_B = 3; n_C = 1; n_D = 0; n_K = 1$$

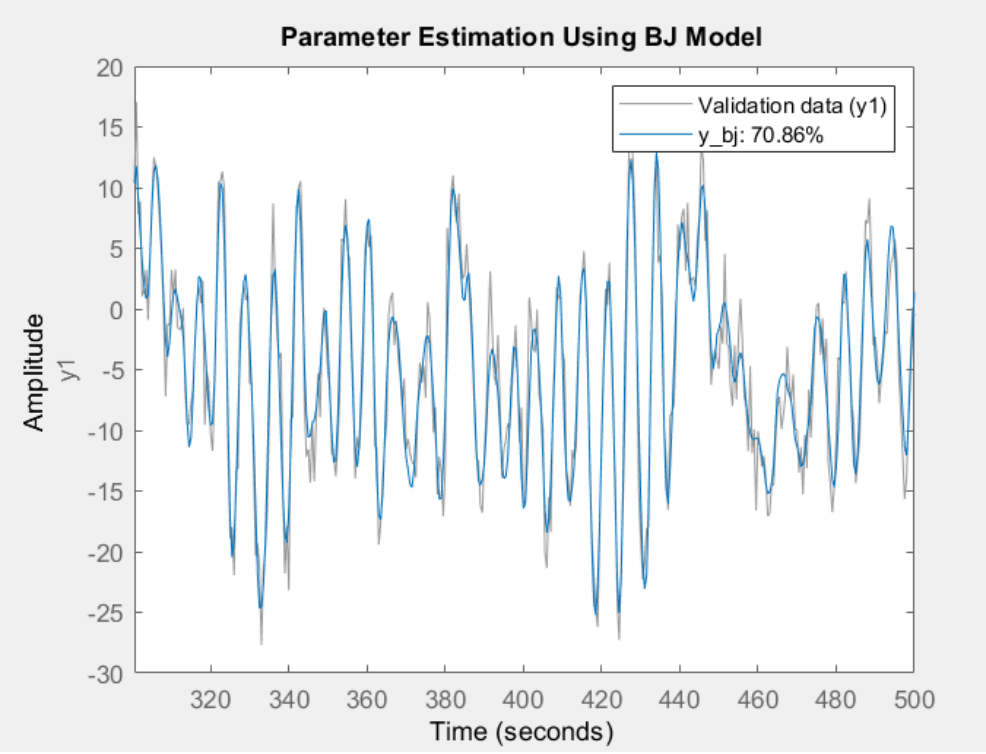

Figure 65 the output of the algorithm using the test data

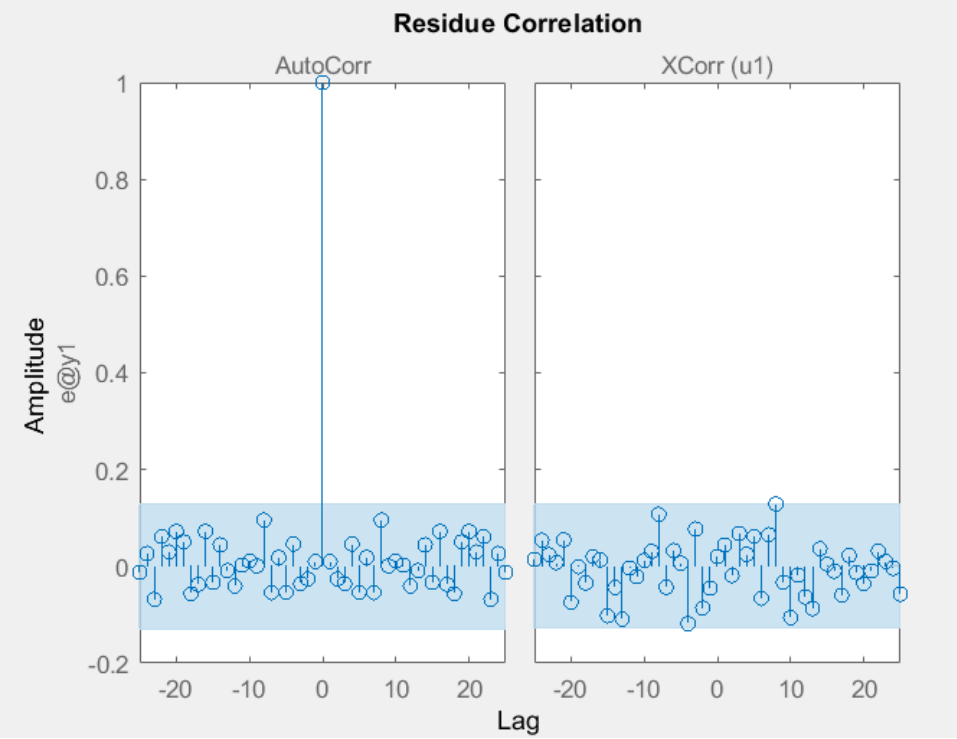

Figure 66 residuals of the ARARX algorithm

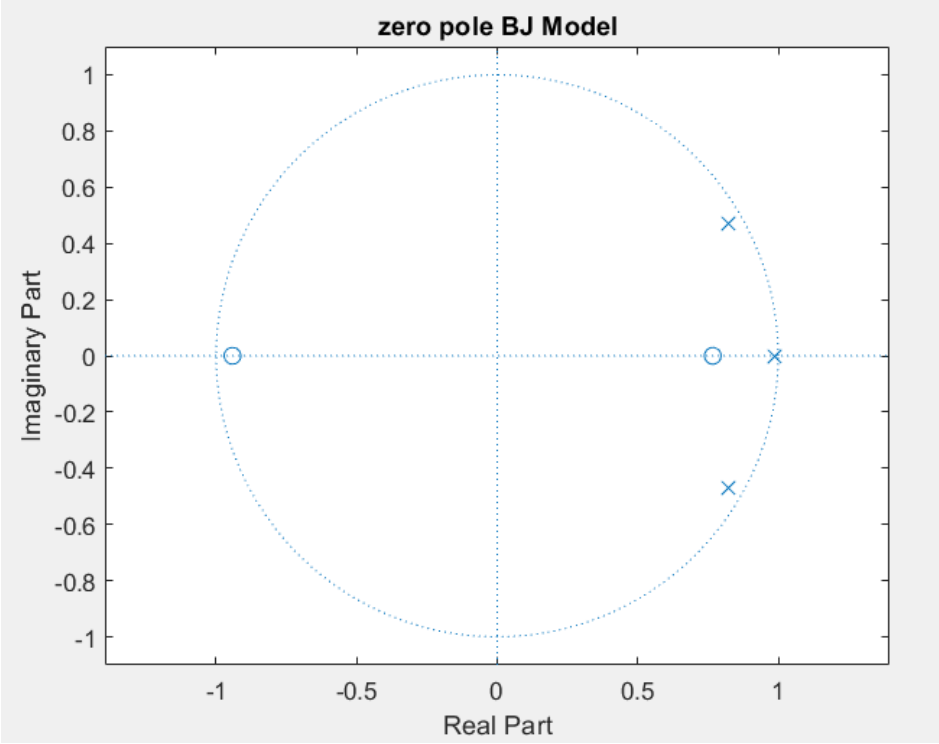

Figure 67 poles and zeros of ARMAX model

```
y_bj =
Discrete-time Polynomial model: y(t) = [B(z)/F(z)]u(t) + C(z)e(t)
  B(z) = 0.6669 z^-1 + 0.1163 z^-2 - 0.4817 z^-3

  C(z) = 1 + 0.02117 z^-1

  F(z) = 1 - 2.637 z^-1 + 2.528 z^-2 - 0.888 z^-3

Sample time: 0.5 seconds

Parameterization:
   Polynomial orders:   nb=3   nc=1   nf=3   nk=1
   Number of free coefficients: 7
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using BJ on time domain data.
Fit to estimation data: 71.75% (prediction focus)
FPE: 6.835, MSE: 6.677
```

OE

$$n_F = 3; n_B = 3; n_K = 1$$

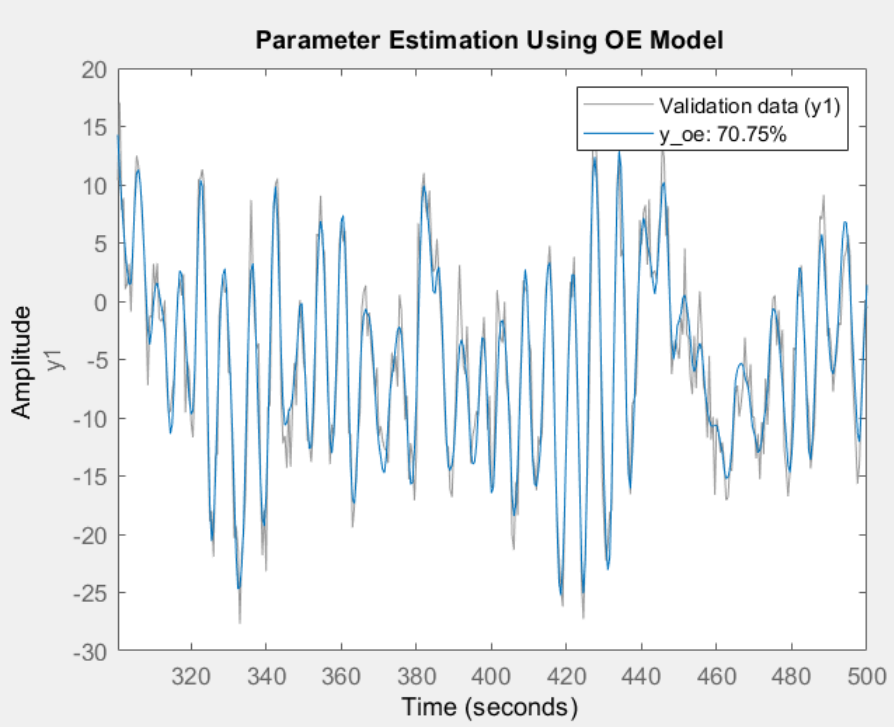

Figure 68 the output of the algorithm using the test data

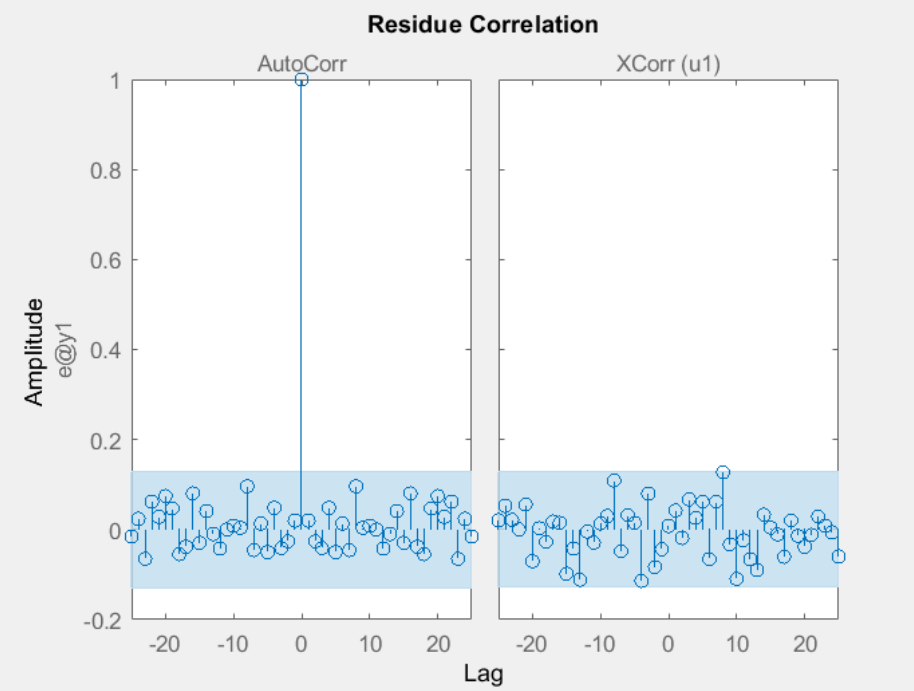

Figure 69 residuals of the ARARX algorithm

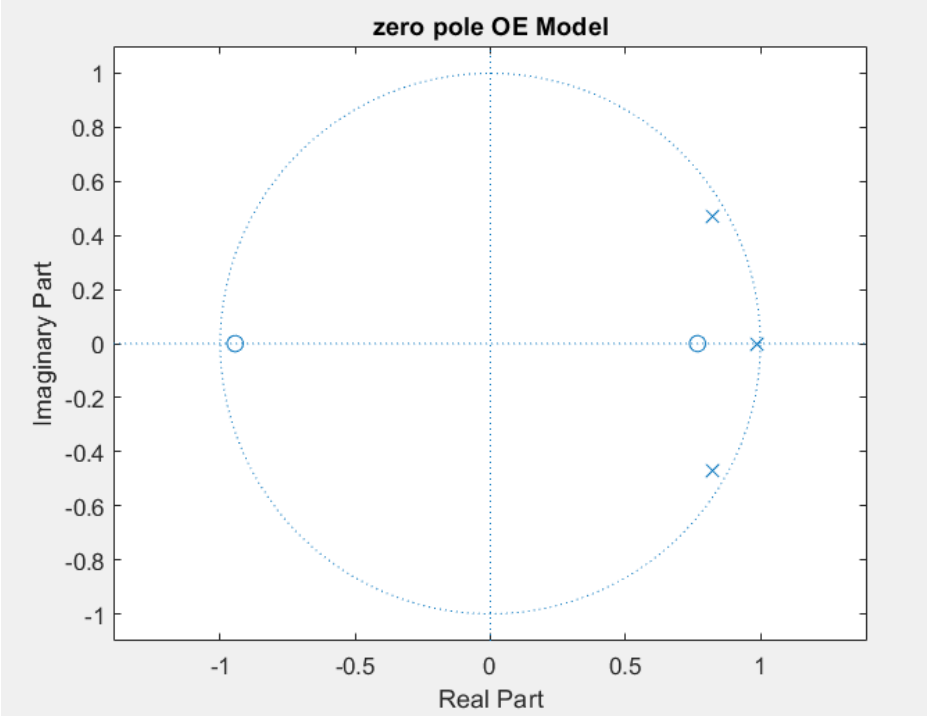

Figure 70 poles and zeros of ARMAX model

```
y_oe =
Discrete-time OE model: y(t) = [B(z)/F(z)]u(t) + e(t)
  B(z) = 0.6661 z^-1 + 0.118 z^-2 - 0.4827 z^-3

  F(z) = 1 - 2.637 z^-1 + 2.528 z^-2 - 0.888 z^-3

Sample time: 0.5 seconds

Parameterization:
   Polynomial orders:   nb=3   nf=3   nk=1
   Number of free coefficients: 6
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using OE on time domain data.
Fit to estimation data: 71.74%
FPE: 6.815, MSE: 6.68
```

The simplest model that gives us the minimum requirements, is the OE method. ARX algorithm has big coefficients and other models are more complex than OE. And most off the dynamics are inside the bound.

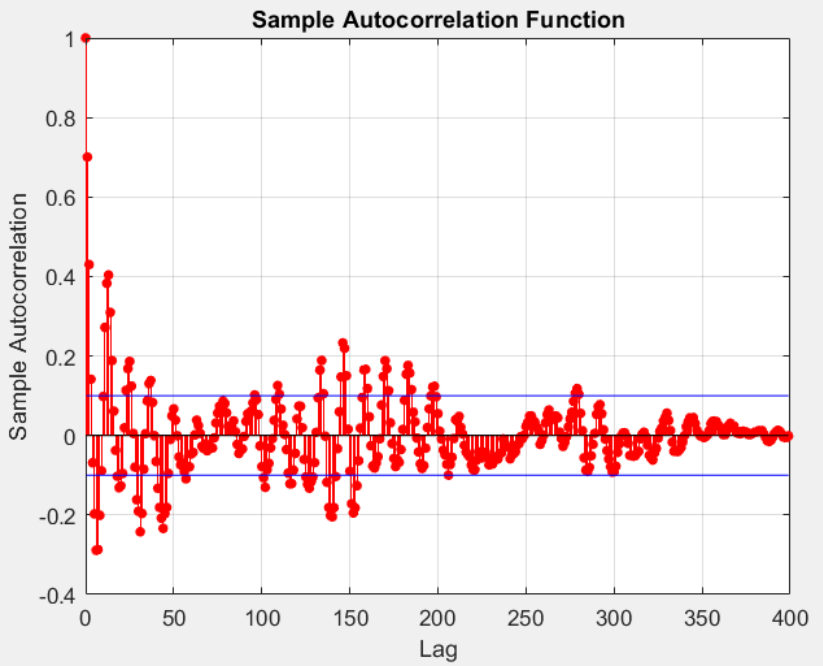

Figure 71 Autocorrelation plot of error on test data in OE algorithm

As time goes on and the estimation runs for a longer time, the information in error disappears and error gets more like white noise.

2)

The closed loop transfer function of the system is:

$$T_{cl} = \frac{\sum}{1 + k\sum}$$

K is a constant value, and according to the first part and OE model we know that the open loop transfer function of the system is:

$$B(z^{-1}) = 0.6661z^{-1} + 0.118z^{-2} - 0.4827z^{-3}$$

$$F(z^{-1}) = 1 - 2.637z^{-1} + 2.528z^{-2} - 0.888z^{-3}$$

$$\sum(z^{-1}) = \frac{B(z^{-1})}{F(z^{-1})}$$

And putting them together in the closed loop transfer function:

$$T_{cl} = \frac{B}{F + KB}$$

If we use ARX model for closed loop system:

$$n_A = 3; n_B = 3; n_k = 1$$

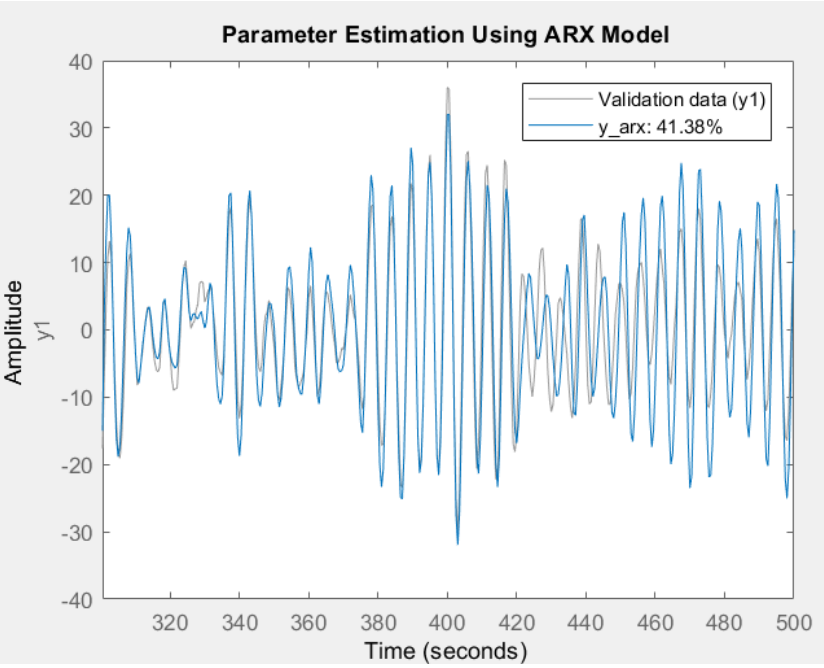

Figure 72 the output of the closed loop system using the test data

It is seen that the output of the estimated model does not follow the actual output.

Now we are going to use ARMAX model to estimate the output:

$$n_A = 3; n_B = 3; n_c = 1; n_k = 1$$

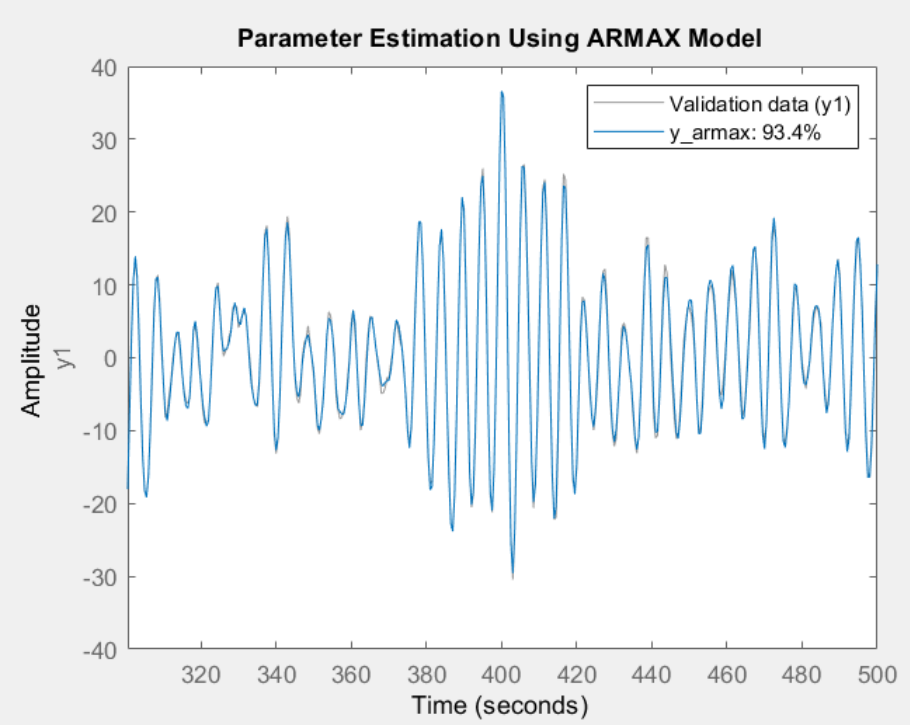

Figure 73 the output of the algorithm using the test data

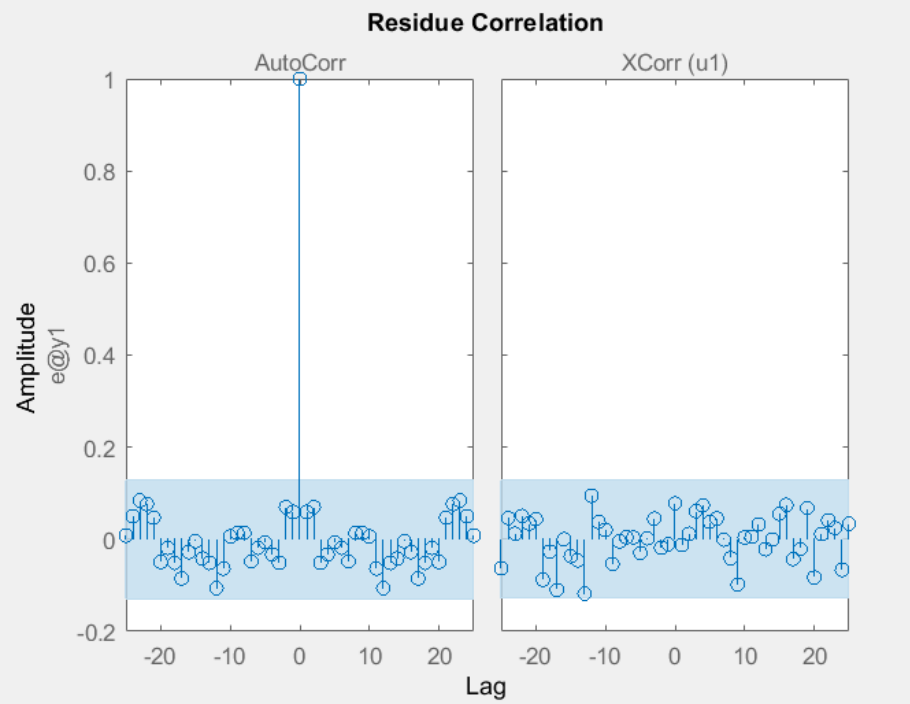

Figure 74 residuals of the ARARX algorithm

ARMAX has a good performance for closed loop systems. All the dynamics are inside the bound.

```
y_armax =
Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)
  A(z) = 1 - 2.595 z^-1 + 2.542 z^-2 - 0.924 z^-3

  B(z) = 0.6315 z^-1 + 0.188 z^-2 - 0.5226 z^-3

  C(z) = 1 - 2.657 z^-1 + 2.575 z^-2 - 0.9148 z^-3

Sample time: 0.5 seconds

Parameterization:
   Polynomial orders:   na=3   nb=3   nc=3   nk=1
   Number of free coefficients: 9
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARMAX on time domain data.
Fit to estimation data: 95.83% (prediction focus)
FPE: 0.2497, MSE: 0.2423
```

According to matlab, the closed loop system is:

$$Y(t) = \left(B'\left(z^{(-1)}\right)/\left(A'\left(z^{(-1)}\right) * U(t) + \left(C\left(z^{(-1)}\right)/\left(A\left(z^{(-1)}\right)\right)^{*} n(t)\right.\right.$$

$$Y(t) = \frac{0.6315z^{(-1)} + 0.188z^{(-2)} - 0.5226z^{(-3)}}{1 - 2.595z^{(-1)} + 2.542z^{(-2)} - 0.924z^{(-3)}} * U(t) + \frac{1 - 2.657z^{(-1)} + 2.575z^{(-2)} - 0.9148z^{(-3)}}{1 - 2.595z^{(-1)} + 2.542z^{(-2)} - 0.924z^{(-3)}} * n(t)$$

$$\Rightarrow Y(t) \simeq \frac{0.6315z^{(-1)} + 0.188z^{(-2)} - 0.5226z^{(-3)}}{1 - 2.595z^{(-1)} + 2.542z^{(-2)} - 0.924z^{(-3)}} * U(t) + n(t)$$

For closed loop system:

$$\mathrm{T_{cl}} = \frac{\mathrm{B}}{\mathrm{F} + \mathrm{KB}}$$

And, we will have:

$$\frac{B}{F + KB} = \frac{B'}{A'} = \frac{0.6315z^{(-1)} + 0.188z^{(-2)} - 0.5226z^{(-3)}}{1 - 2.595z^{(-1)} + 2.542z^{(-2)} - 0.924z^{(-3)}}$$

The OE open loop system, yields to a B as below:

$$\mathrm{B}(\mathrm{z}^{-1}) = 0.6661\mathrm{z}^{-1} + 0.118\mathrm{z}^{-2} - 0.4827\mathrm{z}^{-3}$$

Which is almost equal to the B' from the ARMAX model:

$$F + KB = A' \Rightarrow K = \frac{A' - F}{B} = \frac{0.042z^{(-1)} + 0.014z^{(-2)} - 0.036z^{(-3)}}{0.6661z^{(-1)} + 0.118z^{(-2)} - 0.4827z^{(-3)}}$$

Now the corresponding values of nominator and denominator z^{-1} are 0.042 and 0.6661

$$k = \frac{0.042}{0.6661} = 0.063$$

Question 2

a)

Classic methods are very useful in system identification. In this case using the step response of the system is wildly used. For example if the process has a behavior similar to that of a second order processes we can easily identify if the system is damped, critically damped or undamped (the condition for this is stability). The amount of delay, can be identified as well. In industrial processes step response method is widely used. We can estimate the process with first order delayed or undelayed, second order delayed or undelayed models and etc. For this purpose there are numerous methods available that we studied in Industrial control course.

In frequency response domain, which we mean Bode and Nyquist diagrams, it's sufficient to just determine the ultimate point. The ultimate point is minimum frequency that the frequency response of the system has a phase of -180 degrees.

Simplest model is the two parameter model with gain and time constant, which is determined using the tangent line:

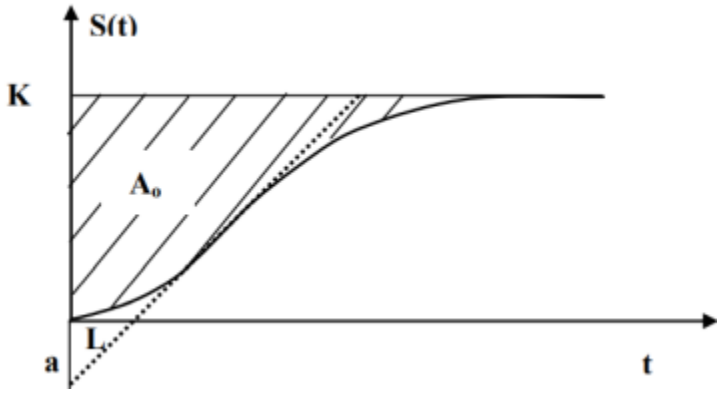

Figure 75 normalized response of he system

$$T_{ar} = \frac{A_\circ}{K}\ A_0 = \int_0^\infty (S(\infty) - S(t))dt = \int_0^\infty (K - S(t))dt$$

$$G_{2a}(s) = \frac{K}{1 + T_{ar}S}$$

The three-part model which contains gain, tine constant and delay is as follows:

$$G(s) = \frac{K}{1 + sT}e^{-sL}$$

$$\tau = \frac{L}{L+T} = \frac{L}{T_{ar}}$$

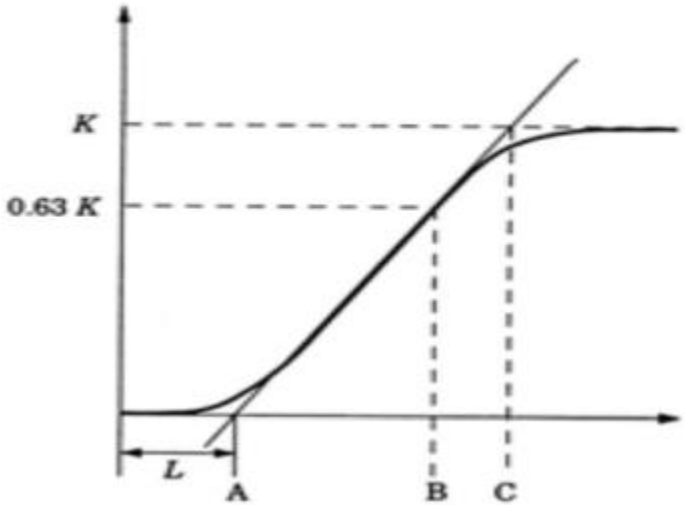

Figure 76 step response of the three component system

In Figure 76 it can be seen that suing a tangent line we can estimate different parameters of the model such as point B (63% of the final response).

Four-component models are estimated for more degree of freedom:

$$G(s) = \frac{Ke^{-sL}}{(1+sT_1)(1+sT_2)} \; S(t) = K\left(1 + \frac{\left(T_2 e^{-\frac{t-L}{T_2}} - T_1 e^{-\frac{t-L}{T_1}}\right)}{T_1 - T_2}\right) T_1 \neq T_2$$

In systems with integrator core, where the final response does not reach the steady state, there are still a few solutions:

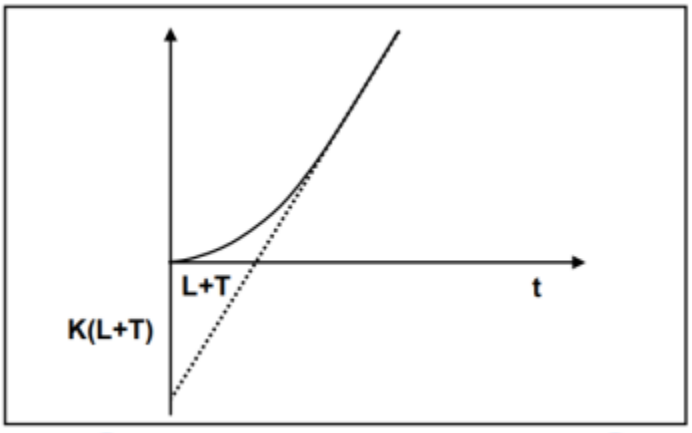

Figure 77 The response of an integrator system.

$$G(s) = \frac{Ke^{-sL}}{s(1+sT)} \; T = \frac{S(L+T)\cdot e^{1}}{K}$$

Gain K and T_{ar} will be determined from the drawings.

For oscillating systems:

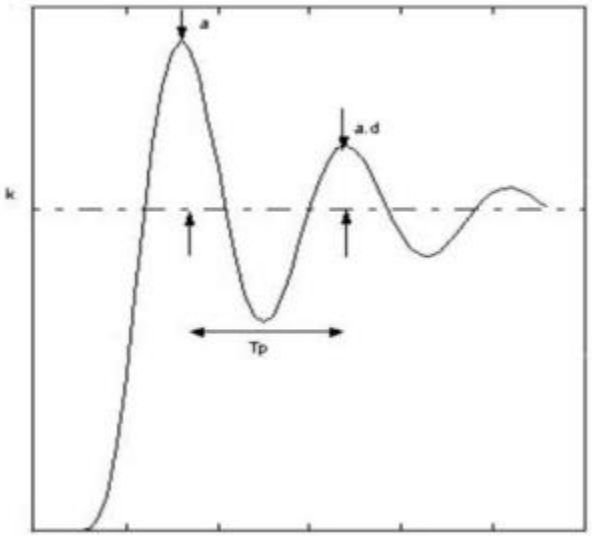

Figure 78 Step response of the oscillating system

$$G(s) = \frac{K\omega^2 e^{-sL}}{s^2 + 2\xi\omega s + \omega^2} \; \omega = \frac{2\pi}{T_p\sqrt{1-\xi^2}}, \xi = \frac{1}{\sqrt{1+\left(\frac{2\pi}{\ln d}\right)^2}}$$

Parameters K and T_p are obvious from the step response of Figure 78 the delay L is also calculated from the plot. For frequency response and Nyquist and Bode diagram, determining the ultimate point would suffice.

Using Ziegler-Nichols, when regulating the gain of the system to oscillate the response of the system. The input and the output of the system will have a -180 degree difference in phase. If the system is at the instability threshold, the frequency ultimate point is determined using the relationship below:

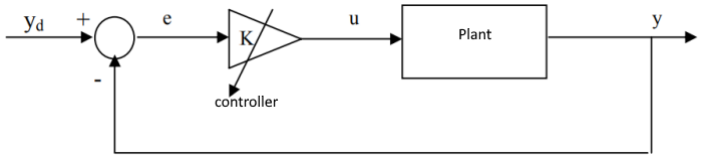

Figure 79 closed loop system with a proportional controller

$$K_u \cdot G(i\omega_u) = -1$$

Another control strategy could be feedback relay, which is a relay with an amplitude of d that is as below:

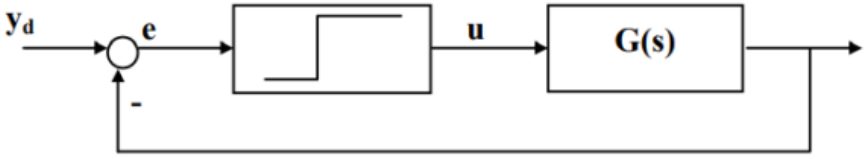

Figure 80 Relay feedback approach

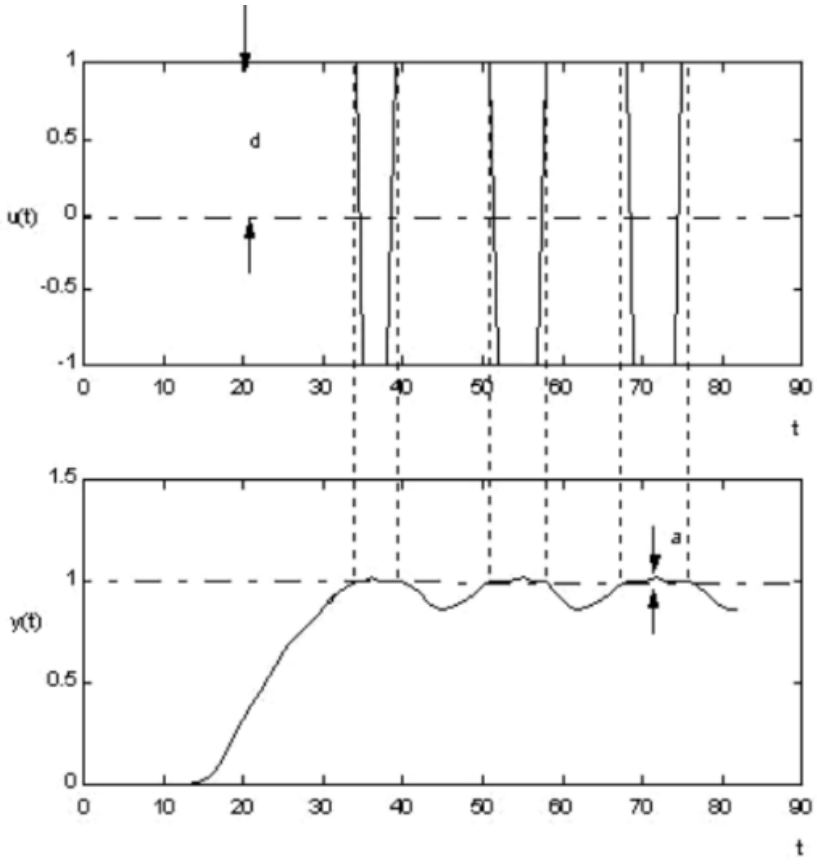

Figure 81 input and output of the relay feedback method

$$G(i\omega_u) = -\frac{\pi a}{4d}$$

ω_u is the ultimate point frequency.

b)

The step response of the system is acquired by giving the system the step input then using the *unknown_sys.p* file we get the step response:

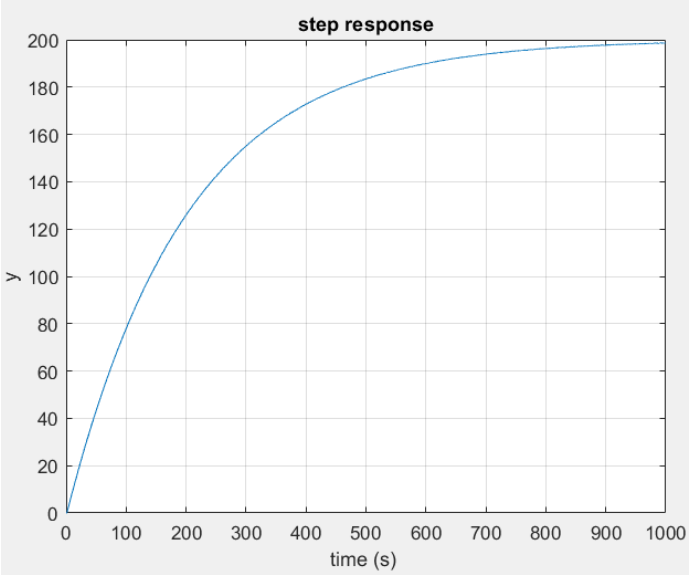

Figure 82 step response of the unknown system

The system is critically damped, and wear egging to use a first order transfer function to model the system. Now we carry out the modeling with two-parameter model $\frac{k}{\tau s+1}$ and three-parameter model $\frac{k}{\tau s+1}e^{-Ts}$.

The final output of the system to step input is 200. So the DC gain of the system is 200 ($k = 200$).

Delay of the system can be calculated by zooming on the origin, as can be seen in Figure 83 the point where the curve starts to take off is (1,0). So the delay of the system is one second.

$$T = 1$$

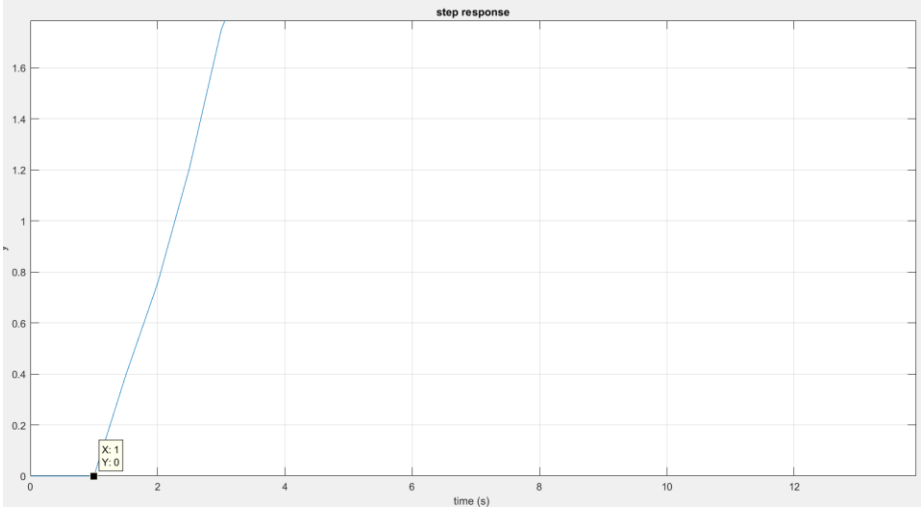

Figure 83 the delay of the system

The rise time or the time required for the system to reach 63% of the final response is:

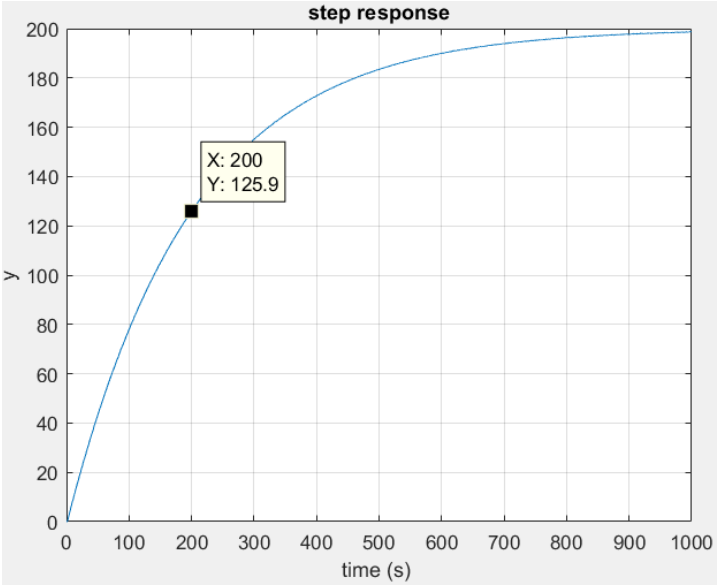

Figure 84 the rise time is almost 200 seconds

So the time constant of the system is:

$$\tau = 200 - 1$$

The above time is calculated by subtracting the delay from the rise time.

So for the three-parameter model we will have;

$$\frac{200}{199s + 1} e^{-s}$$

The two-parameter model is:

$$A_0 = \frac{200*400}{2} = 40000, T_{ar} = \frac{A}{k} = \frac{40000}{200} = 200$$

$$G = \frac{200}{200s + 1}$$

And for testing the accuracy of the two-parameter model, we use the compare command of matlab, and the test data:

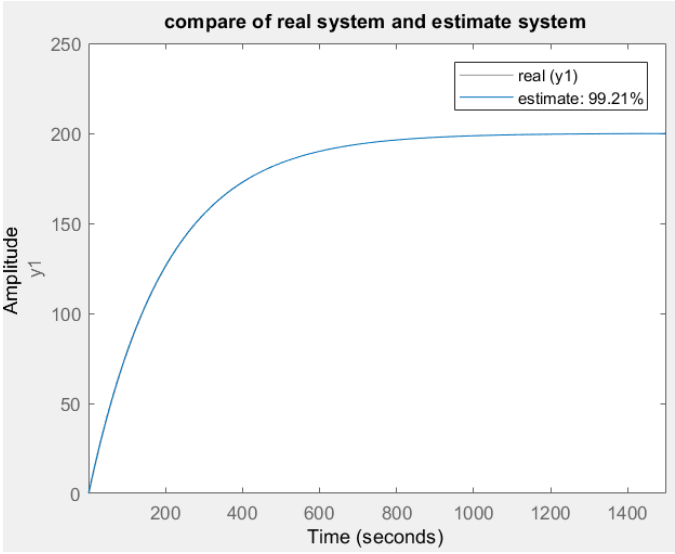

Figure 85 the output of the two-parameter model

And for the three-parameter model, we act accordingly:

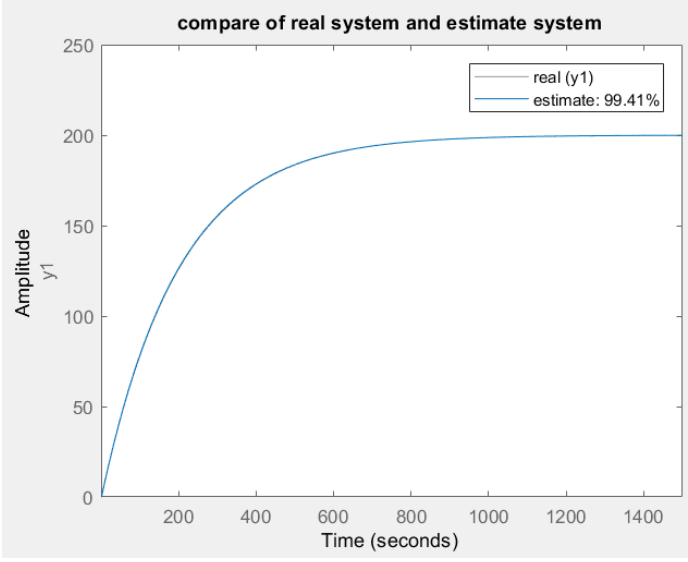

Figure 86 the output of the three-parameter model

With the three-parameter model the accuracy has increased slightly, but overly the accuracy is good for both models.

But if we use the ***procest*** model, the accuracy increases further:

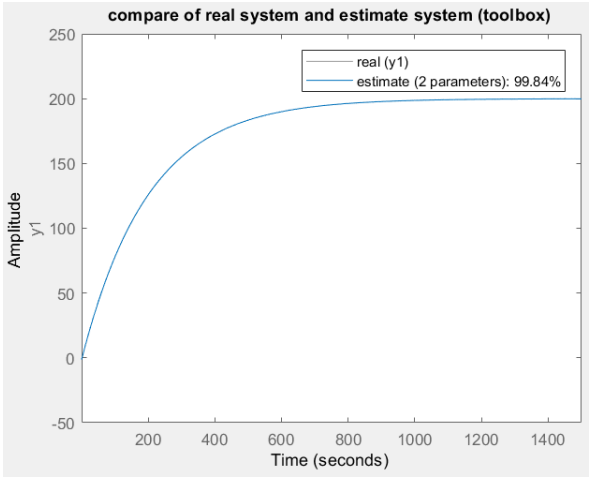

Figure 87 matlab toolbox procest command two-parameter model

```
p1 =
Process model with transfer function:
             Kp
  G(s) = ----------
          1+Tp1*s

        Kp = 200
       Tp1 = 200.03

Parameterization:
    'P1'
   Number of free coefficients: 2
   Use "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using PROCEST on time domain data "data3".
Fit to estimation data: 99.84%
FPE: 0.005407, MSE: 0.005396
```

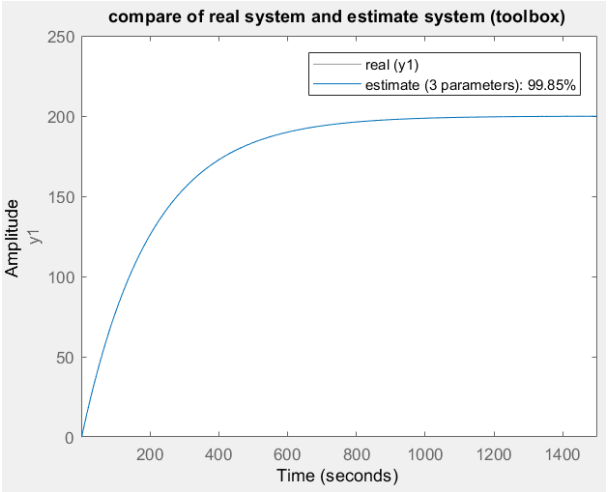

Figure 88 matlab toolbox procest command for three-parameter model

```
p1d =
Process model with transfer function:
             Kp
  G(s) = ---------- * exp(-Td*s)
         1+Tp1*s

        Kp = 200
       Tp1 = 200
        Td = 1.328

Parameterization:
    'P1D'
   Number of free coefficients: 3
   Use "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using PROCEST on time domain data "data3".
Fit to estimation data: 99.85%
FPE: 0.004573, MSE: 0.004561
```

If we compare the results of manual modeling with the matlab toolbox, we realize that both models are pretty much alike.

Now we examine the two-parameter model on the noisy data and chirp data. First we quantize the transfer function we derived from the last step. After quantization, we compare the actual results and the results from the quantized transfer function using ***compare*** command of matlab.

For input and gassian noise with variance 0.05:

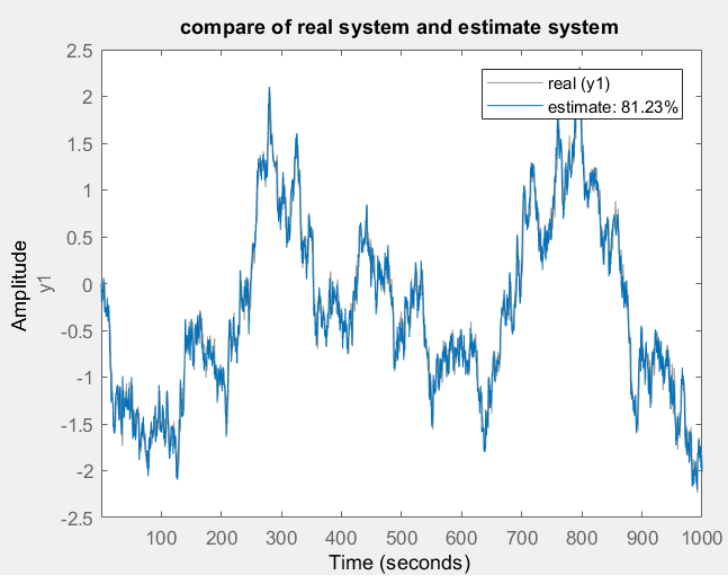

Figure 89 the actual system and the two-parameter model output with noise

The accuracy has reached 81.23%, which is acceptable with this amount of noise.

For chirp input:

The input is created with the lines below:

```
t=0:1:1999;
input=chirp(t, 0.01, 2000, 500)';
```

The output is:

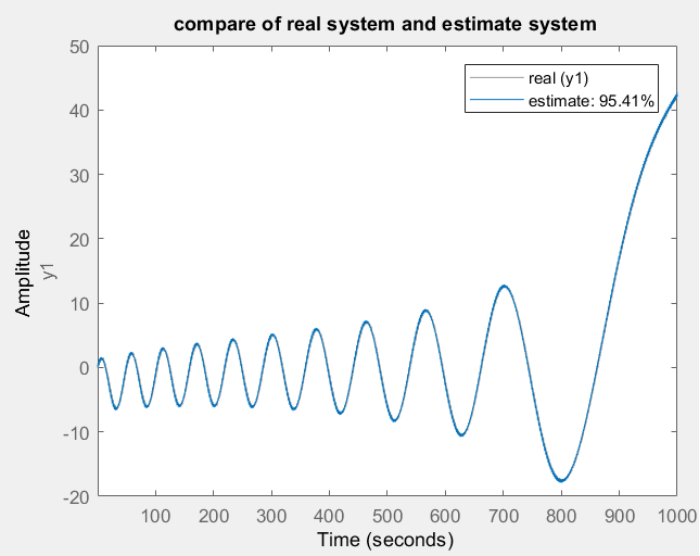

Figure 90 the output of the system with chirp input to the actual system and the model

The estimated model has managed to give us the output of the system to inputs other than step input. So this model shows a very good behavior of the actual system. We can say the model is a good representation of the actual system.

c)

We have the input and output data. And we want to use the linearity properties of the system, and use ARMAX to estimate the parameters of the system.

The steps for this purpose are:

1. Using ARX, give an initial estimation for A and B($\hat{A}, \hat{B}$)
2. Grant that $c = 1$ ϵ or prediction error of the next step and use it as the prediction error for ARMAX.

$$e_{ARX}(k) = \hat{A}(q)y(k) - \hat{B}(q)u(k)$$

3. Now we carry out LS estimation using the matrix below:

$$\begin{bmatrix} -y(t-1) \\ u(t-1) \\ \epsilon(t-1) \end{bmatrix}$$

4. Update A, B, C and prediction error
5. Continue till the stopping criterion is met

We undertake the estimation with 1200 training data points and 800 testing data points. After acquiring the parameters using ESL, we will have:

$$Y = \frac{B(z^{-1})}{A(z^{-1})} * U + \frac{C(z^{-1})}{A(z^{-1})} * n$$

$$A = 1 - 0.9951z^{-1}$$

$$B = 0.9712z^{-1}$$

$$C = 1 + 0.9185z^{-1}$$

Now we can compare ELS and ARMAX together:

```
ARMAX_Real =
Discrete-time ARMAX model:  A(z)y(t) = B(z)u(t) + C(z)e(t)
  A(z) = 1 - 0.9951 z^-1

  B(z) = 0.9876 z^-1

  C(z) = 1 - 0.594 z^-1

Sample time: 0.5 seconds

Parameterization:
   Polynomial orders:   na=1   nb=1   nc=1   nk=1
   Number of free coefficients: 3
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARMAX on time domain data "data".
Fit to estimation data: 99.79% (prediction focus)
FPE: 0.01031, MSE: 0.01026
```

$A_{ARMAX} = -0.9951$
$A_{ELS} = -0.9945$
$B_{ARMAX} = 0.9876$
$B_{ELS} = 1.0734$
$C_{ARMAX} = -0.594$
$C_{ELS} = 0.9105$
The parameters form both approaches area pretty close to each other.

Now it's time to examine the convergence of the outputs from the two methods.

ELS:

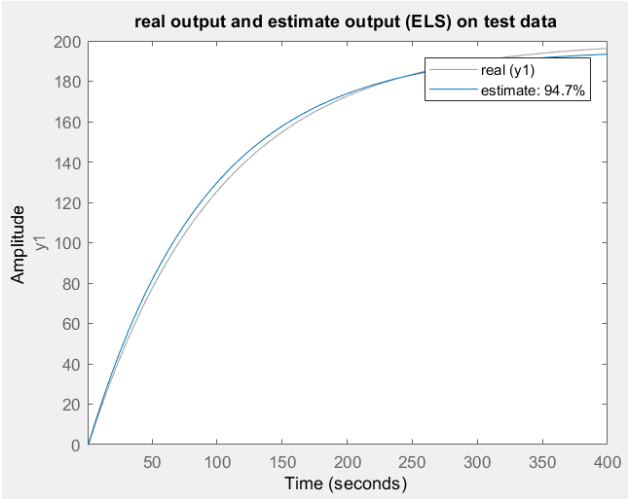

Figure 91 the ELS estimation approach and the actual system output

ARMAX:

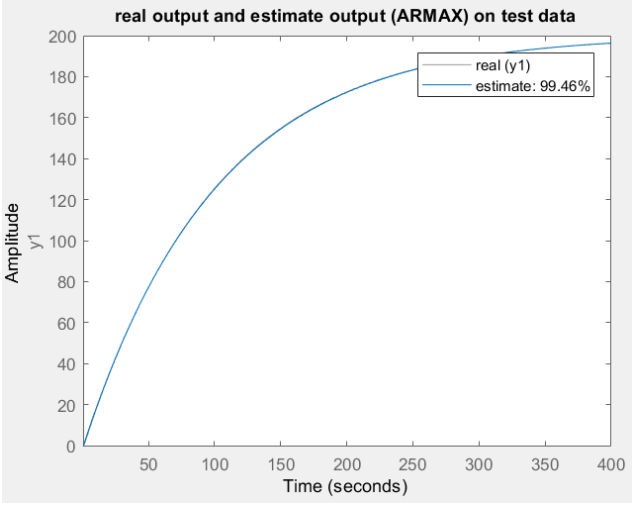

Figure 92 the ARMAX toolbox estimation approach and the actual system output

It is seen that ARMAX approach is far more superior to the ELS method.

The Auto correlation plot of the systems are as follows:

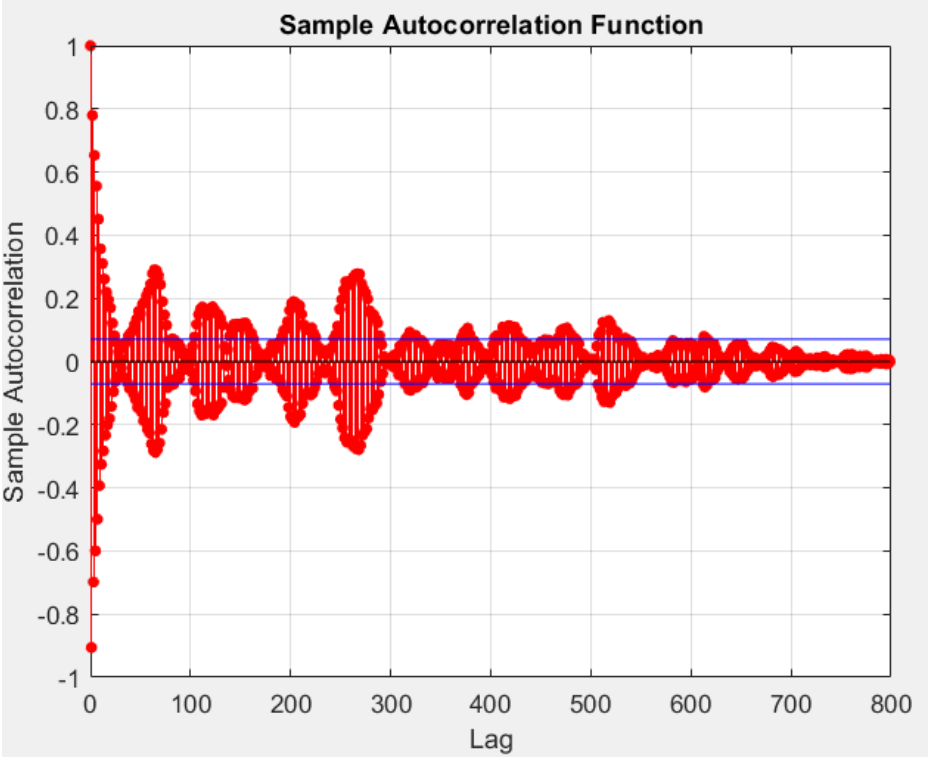

Figure 93 the auto correlation plot of the prediction error for step input

As the estimation perpetuates, the dependency in error decreases until it stays in the desired bound. And the error becomes the white noise.

Now we feed the model and system with chirp input:

chirp(t,0.01,1200,500)

we initialize the algorithm with values:

$$n_a = 3, n_b = 2, n_c = 1, n_k = 3$$

$A1_{ARMAX} = -0.9845$
$A1_{ELS} = -1.2714$
$A2_{ARMAX} = -0.1166$
$A2_{ELS} = 0.1554$
$A3_{ARMAX} = 0.1029$
$A3_{ELS} = 0.1174$
$B3_{ARMAX} = 0.3819$

$B3_{ELS} = 0.3879$
$B4_{ARMAX} = 0.004658$
$B4_{ELS} = -0.1068$
$C1_{ARMAX} = 0.618$
$C1_{ELS} = -0.0755$
As can be noticed, the parameters are close to each other.
Next case is the chirp input:
chirp(t,0.01,1200,500)
Let's see what the result of the estimation is:

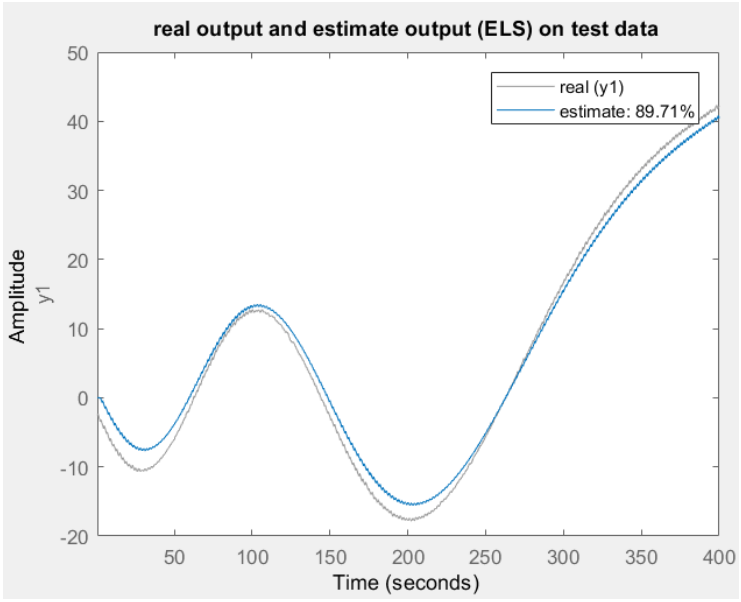

Figure 94 the output of ELS algorithm to step input

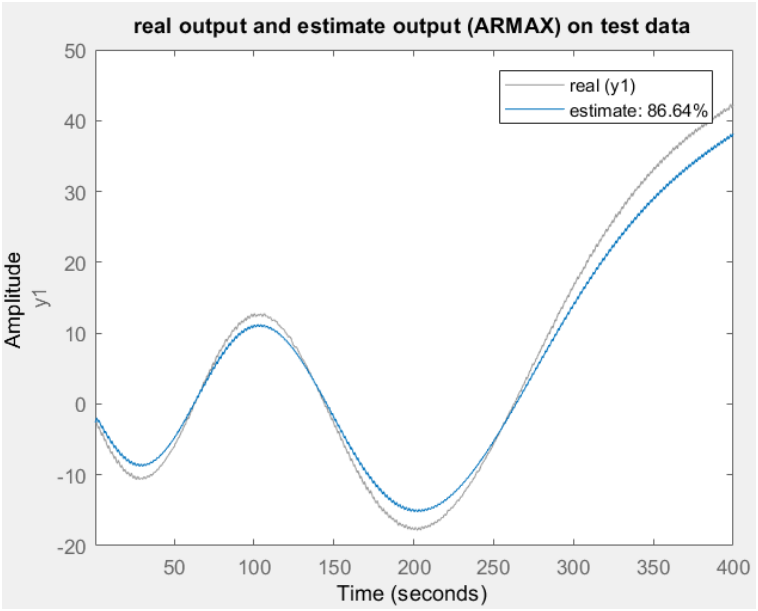

Figure 95 The output of ARMAX algorithm to step input

It's interesting that ELS algorithm has a better performance on the test data than the ARMAX algorithm.

The auto correlation of prediction error om the test data of ELS algorithm in

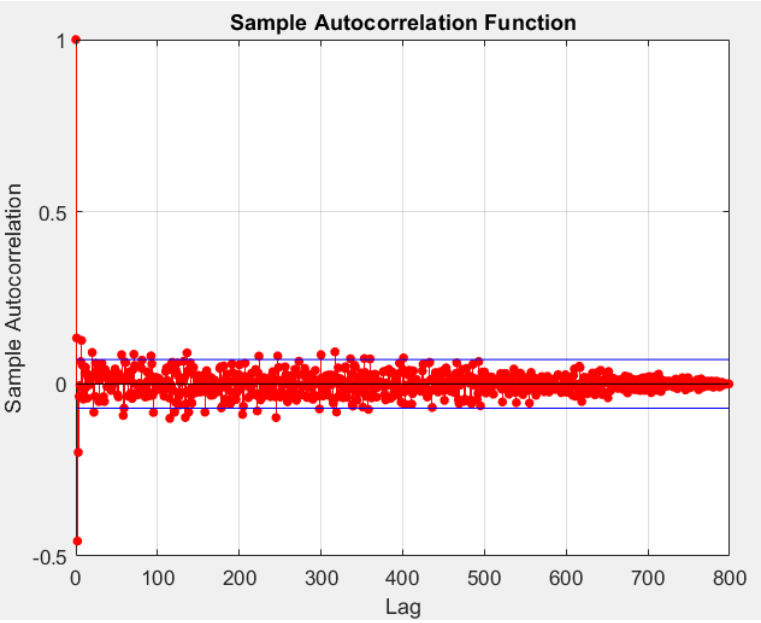

Figure 96 shows that the error is within the bound, and the error is similar to white noise.

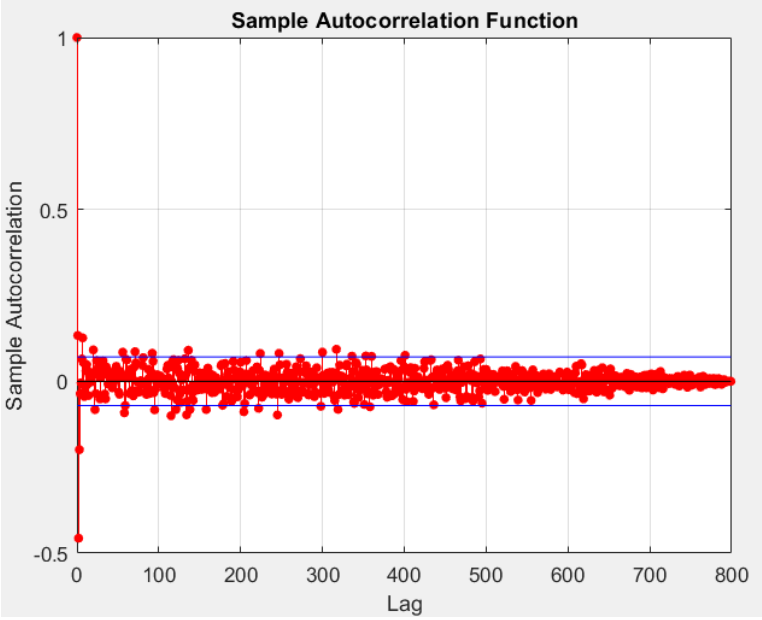

Figure 96 auto correlation of prediction error for chirp input

Question 3

The dataset is composed of two parts:

1. Train data
2. Test data

The train part has 300 data points.

The test part has 100 data points.

Input output plot of the data is as below:

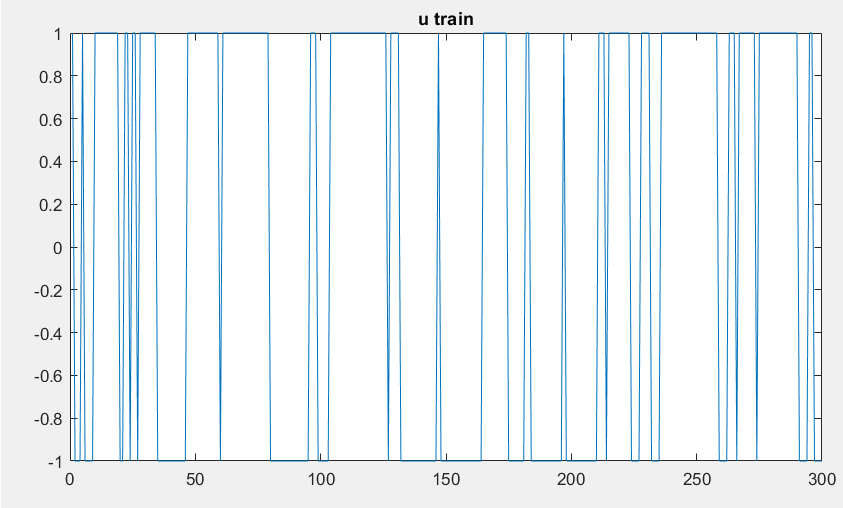

Figure 97 the training input

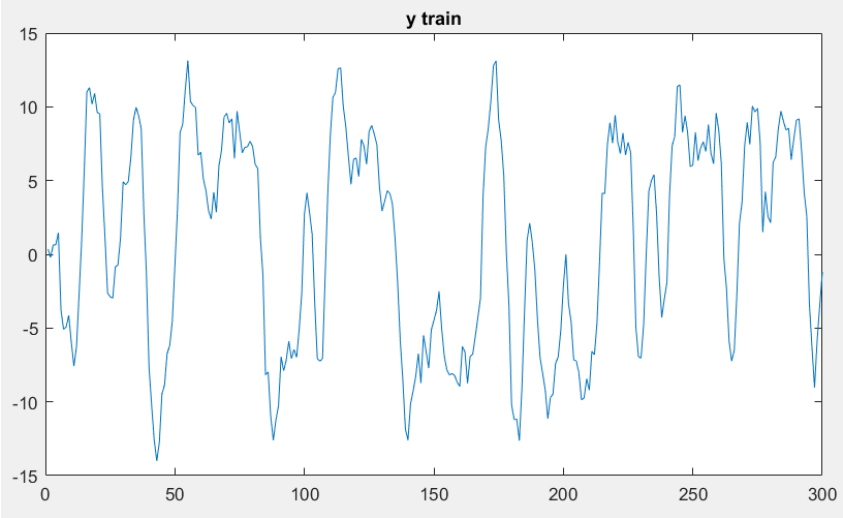

Figure 98 the training output

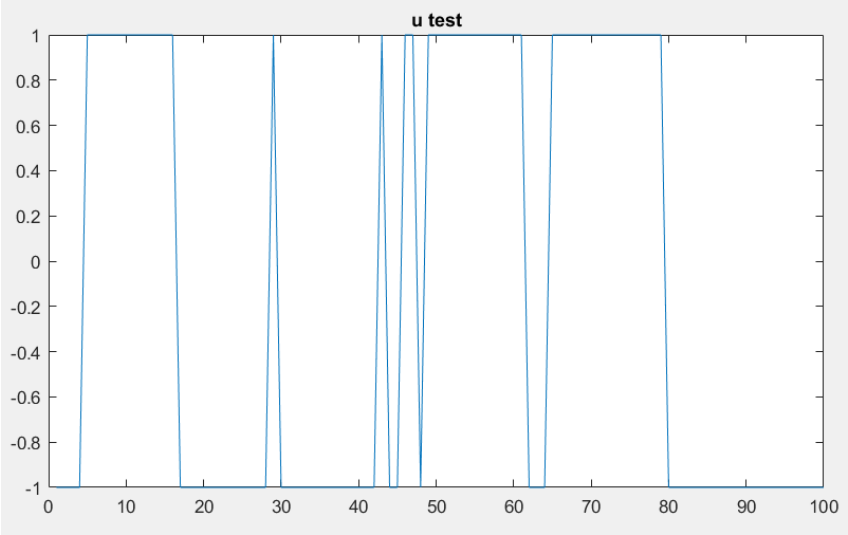

Figure 99 the testing input

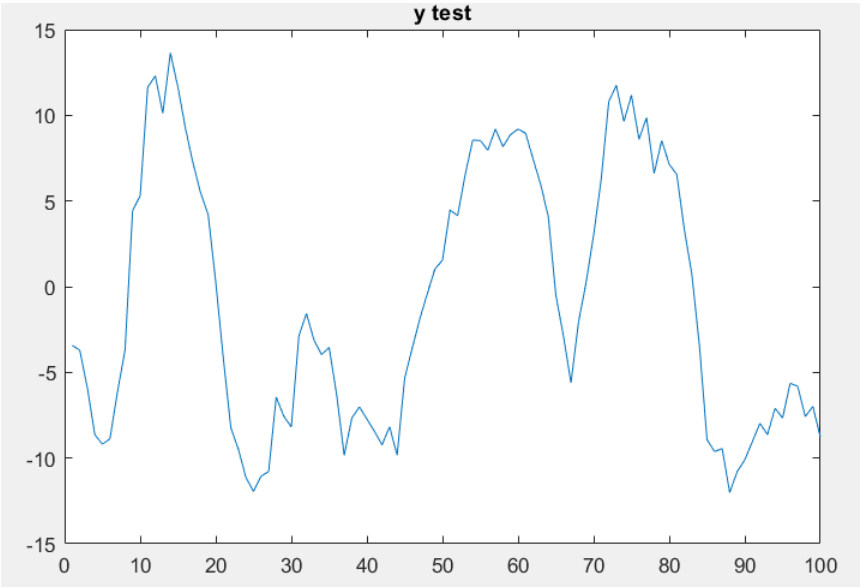

Figure 100 the testing output

Some other attributes of this data are the time constant of 0.1s and it is consist of two iddata objects.

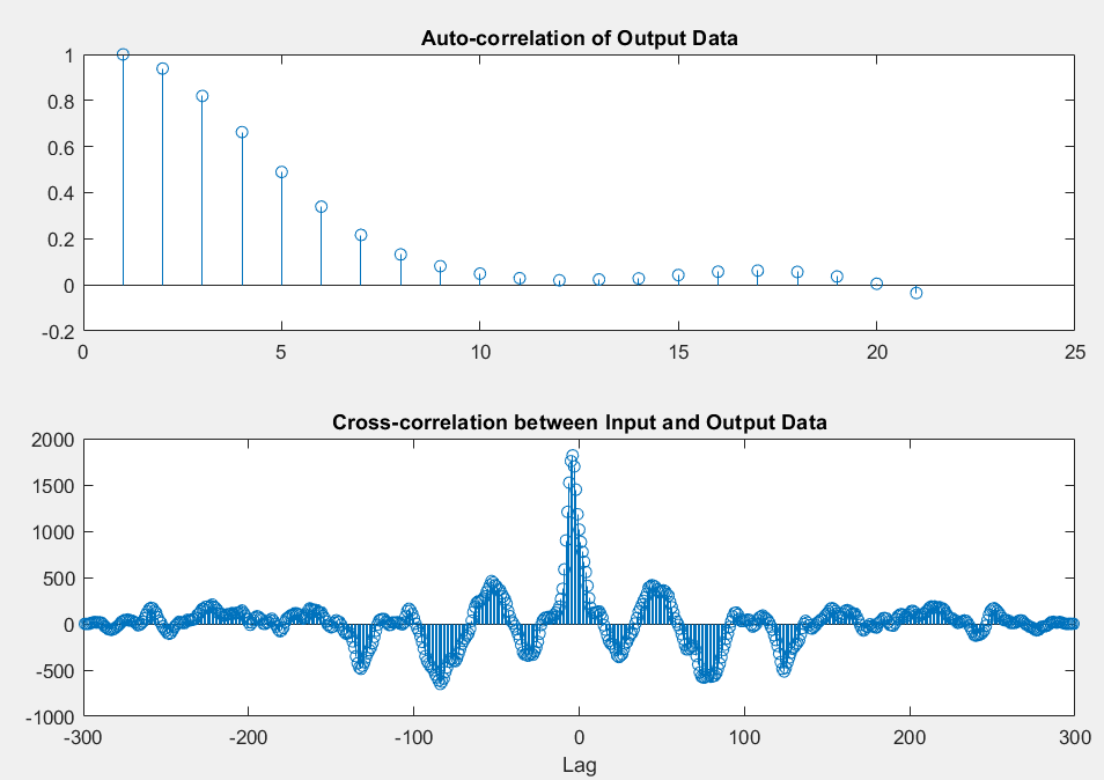

Figure 101 Auto correlation and cross correlation between input and output

First we try **with normalization** and the initial parameters:

$$n_a = 3; n_b = 3;\ n_k = 1;\ n_f = 1;\ n_c = 1$$

OE:

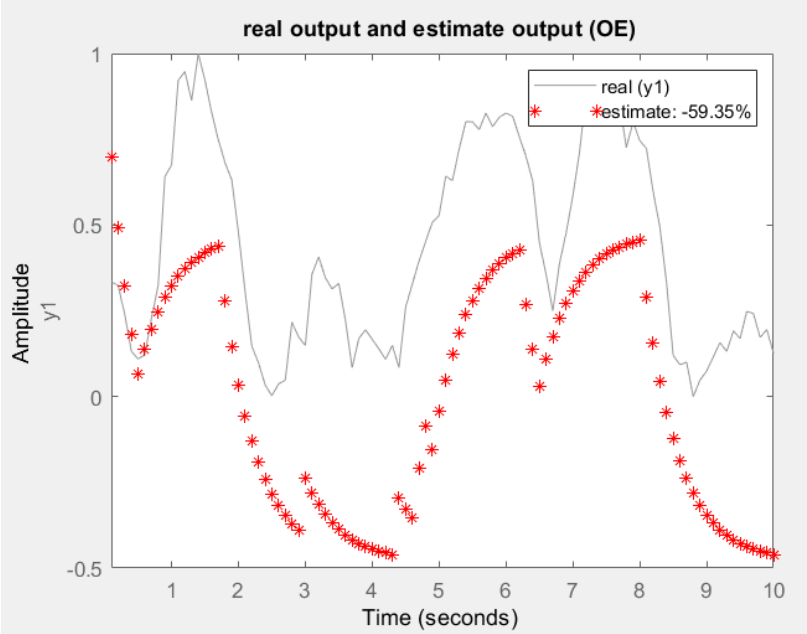

Figure 102 OE model of the data (normalized)

ARX:

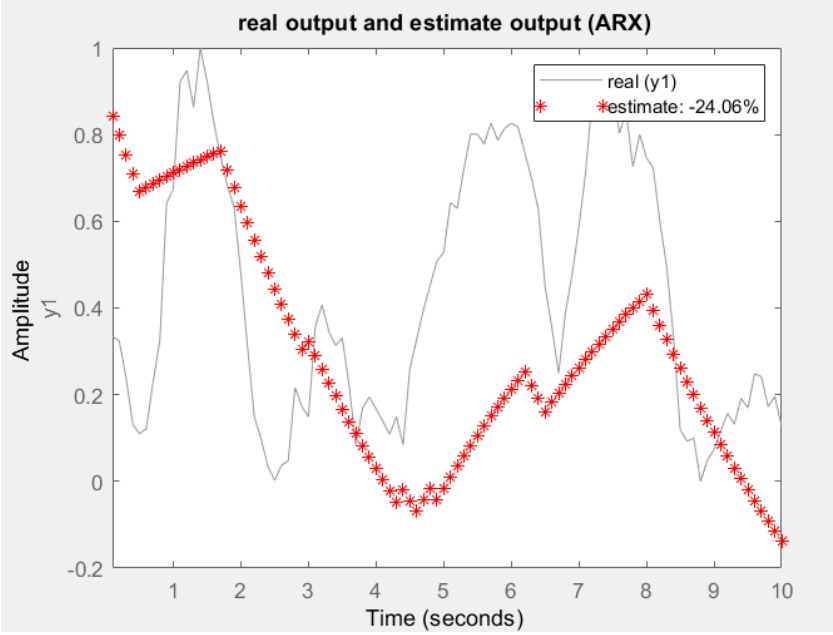

Figure 103 ARX model of the data (normalized)

ARMAX:

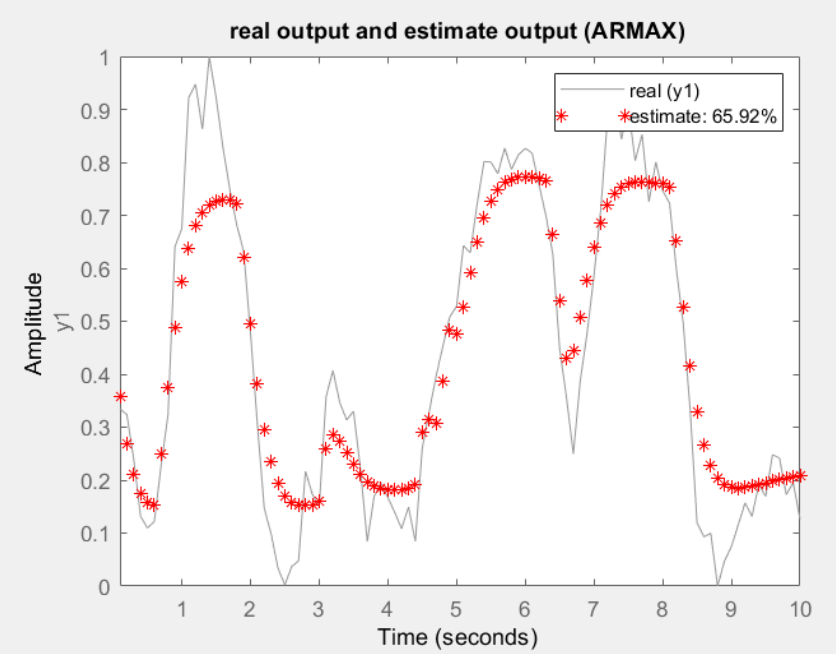

Figure 104 ARMAX model of the data (normalized)

The accuracy of the models are not acceptable, we have low accuracies with normalization and the initial parameters. Paying attention to the results we see the best performance is that of the ARMAX model.

So First we don't use normalization for the data and see how it affects our results:

Not Normalized and parameters as before:

Before proceeding to this part one thing Is worth mentioning, which is, because the system is a real system, it is more likely to be closed loop. So we are going to carry out the estimation task with the assumption that the system is closed loop.

OE:

```
Discrete-time OE model: y(t) = [B(z)/F(z)]u(t) + e(t)
  B(z) = 2.024 z^-1

  F(z) = 1 - 0.7689 z^-1

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   nb=1   nf=1   nk=1
   Number of free coefficients: 2
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using OE on time domain data "data1".
Fit to estimation data: 49.27%
FPE: 13.23, MSE: 13.05
```

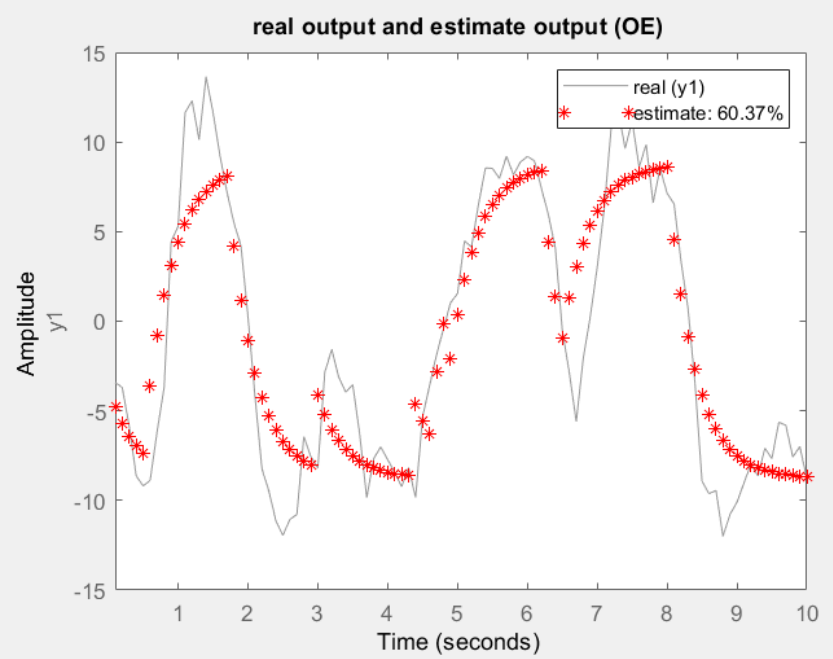

*Figure 105 OE model of the data (**not normalized**)*

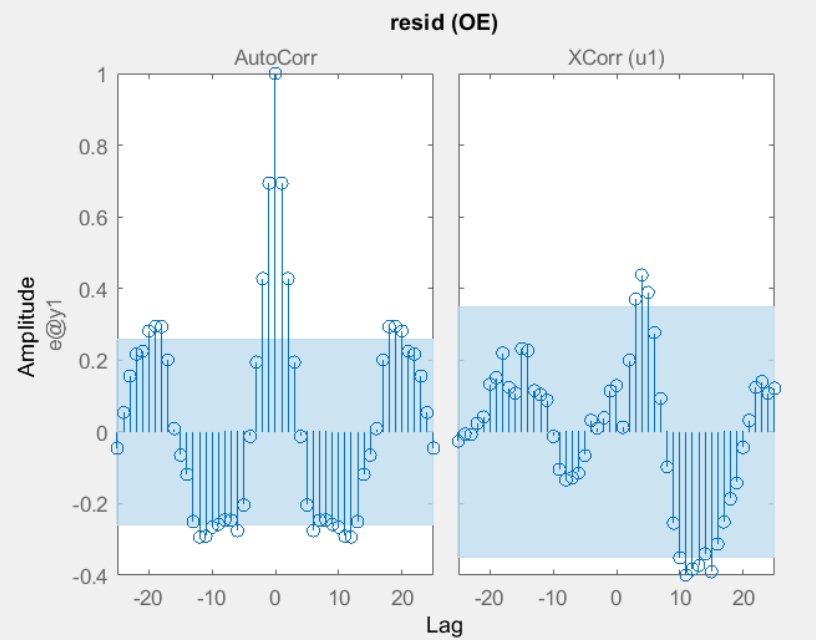

*Figure 106 residuals of OE model (**not normalized**)*

ARX:

```
estimated_y_1 =
Discrete-time ARX model: A(z)y(t) = B(z)u(t) + e(t)
  A(z) = 1 - 0.8746 z^-1

  B(z) = 0.9865 z^-1

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=1   nb=1   nk=1
   Number of free coefficients: 2
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARX on time domain data "data1".
Fit to estimation data: 67.69% (prediction focus)
FPE: 5.4, MSE: 5.293
```

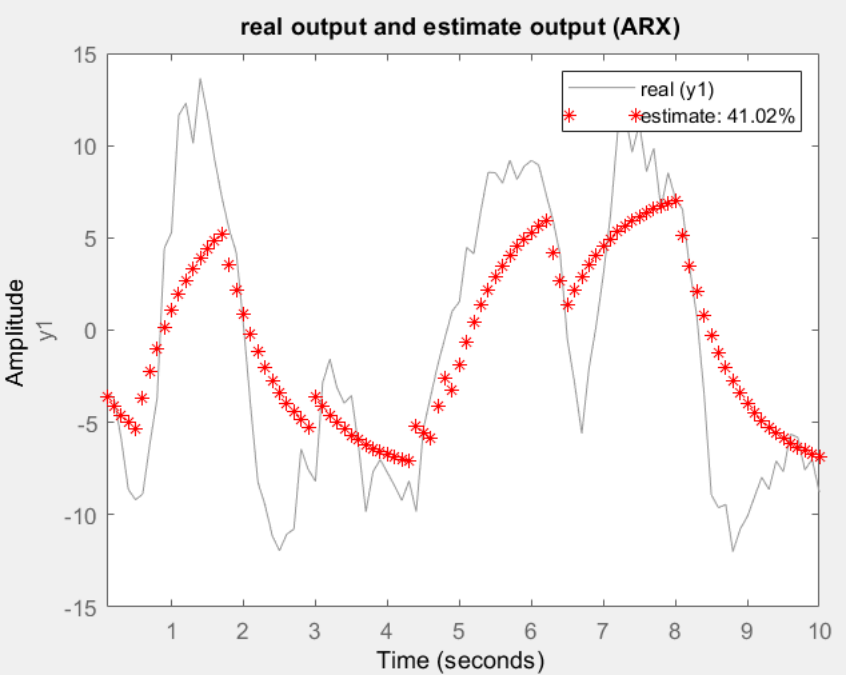

*Figure 107 ARX model of the data (**not normalized**)*

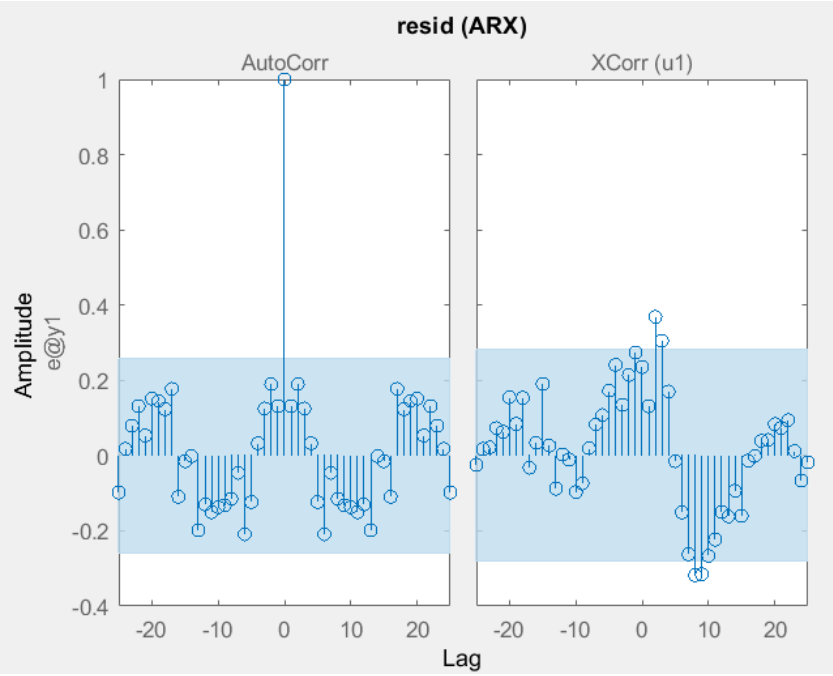

*Figure 108 residuals of ARX model (**not normalized**)*

ARMAX:

```
Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)
  A(z) = 1 - 1.13 z^-1 + 0.1487 z^-2 + 0.2492 z^-3

  B(z) = 0.0752 z^-1 + 0.7298 z^-2 + 1.204 z^-3

  C(z) = 1 - 0.8752 z^-1

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=3   nb=3   nc=1   nk=1
   Number of free coefficients: 7
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARMAX on time domain data "data1".
Fit to estimation data: 84.9% (prediction focus)
FPE: 1.212, MSE: 1.157
```

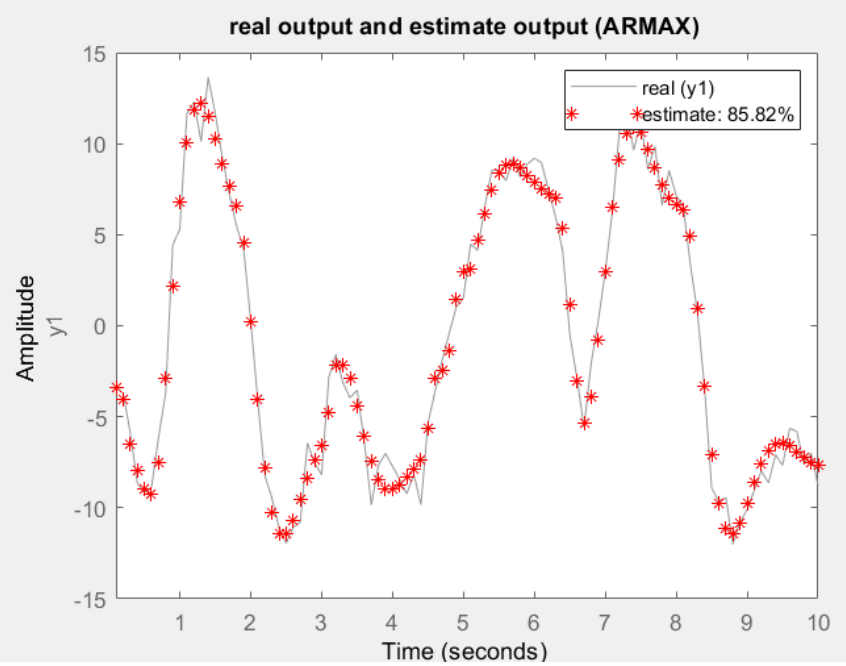

Figure 109 ARMAX model of the data ***(not normalized)***

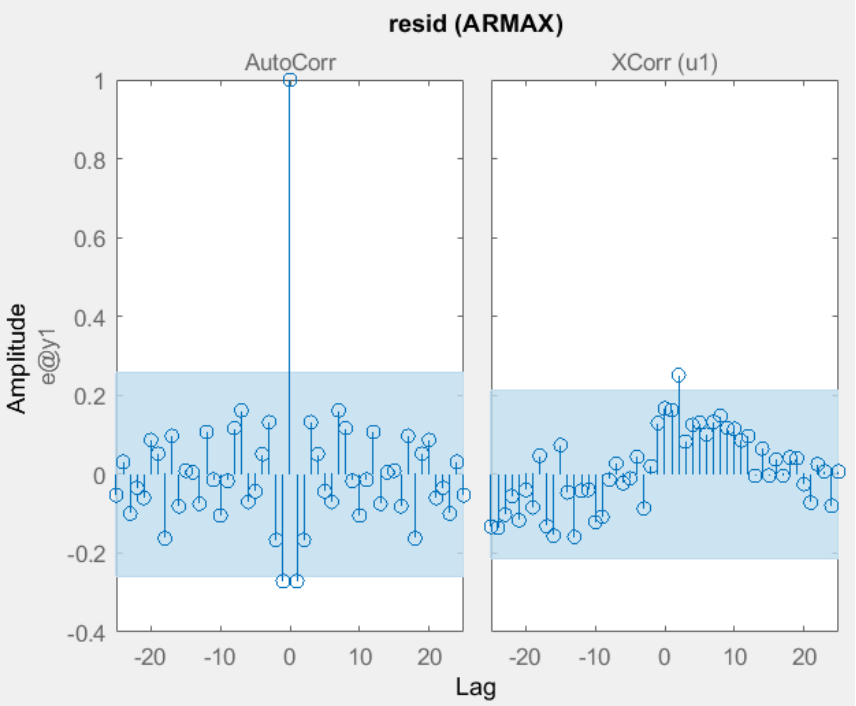

*Figure 110 residuals of ARMAX model (**not normalized**)*

We are better without normalization!!

Without normalization and initial parameters $n_a = 3; n_b = 3;\ n_k = 1;\ n_f = 1;\ n_c = 1$, **the best model that reaches the threshold of 85% is ARMAX model**.
Now, we have a few other options. We can play up with the parameters and see what the results would look like.

One option was the rule of thumb prof. Aliyari mentioned:

$$n_b = n_a - 1;\ n_c < n_d;\ n_a \cong n_f$$

$$n_d = n_a; n_d > n_f$$

So the parameters are initialized as follows:

$$n_a = 3; n_b = 2;\ n_k = 1;\ n_f = 3;\ n_c = 2$$

ARX:

```
Discrete-time ARX model: A(z)y(t) = B(z)u(t) + e(t)
  A(z) = 1 - 0.8746 z^-1

  B(z) = 0.9865 z^-1

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=1   nb=1   nk=1
   Number of free coefficients: 2
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARX on time domain data "data1".
Fit to estimation data: 67.69% (prediction focus)
FPE: 5.4, MSE: 5.293
```

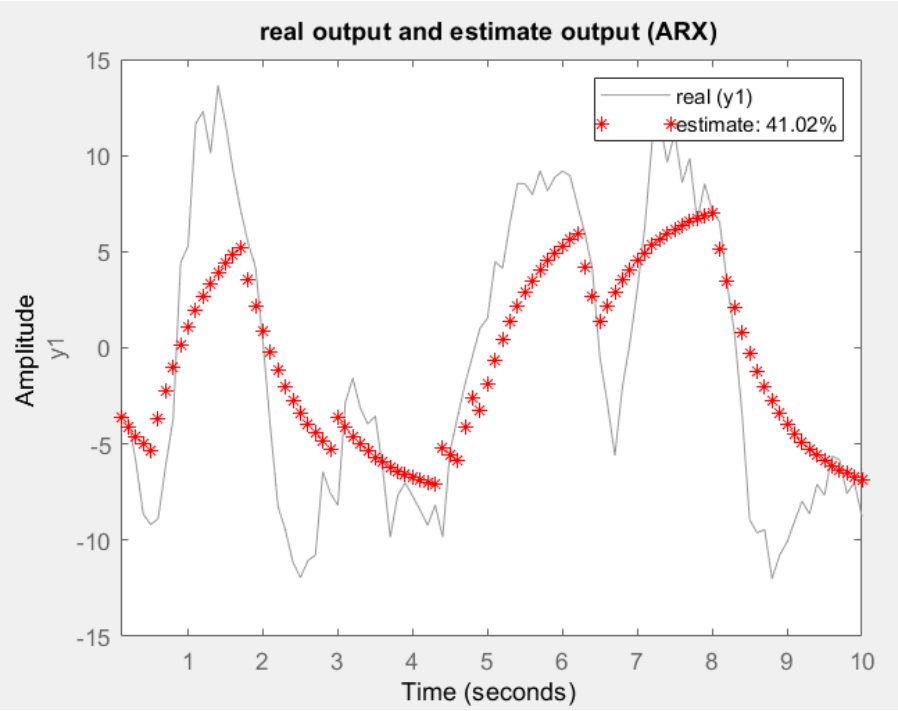

*Figure 111 ARX model of the data (**not normalized**)*

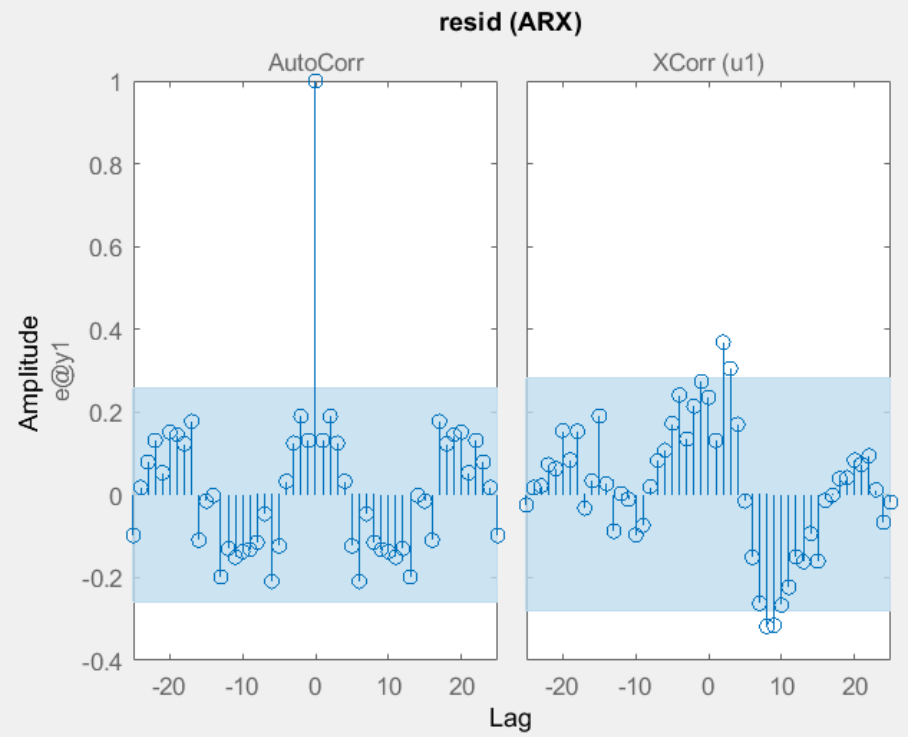

*Figure 112 residuals of ARX model (**not normalized**)*

OE:

```
Discrete-time OE model: y(t) = [B(z)/F(z)]u(t) + e(t)
  B(z) = 0.561 z^-1

  F(z) = 1 - 2.23 z^-1 + 1.813 z^-2 - 0.5125 z^-3

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   nb=1   nf=3   nk=1
   Number of free coefficients: 4
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using OE on time domain data "data1".
Fit to estimation data: 82.91%
FPE: 1.521, MSE: 1.481
```

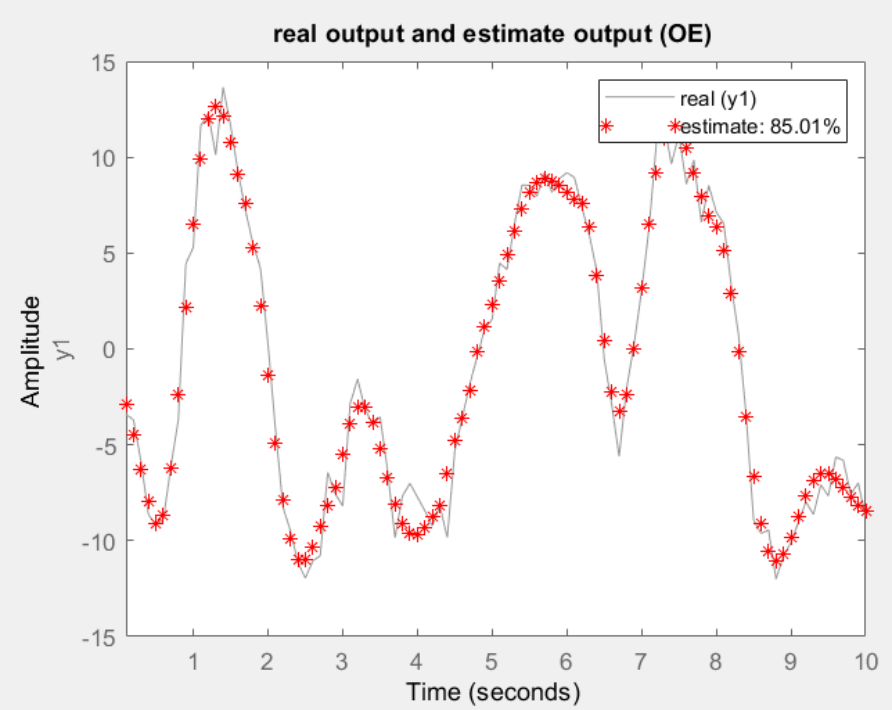

*Figure 113 OE model of the data (**not normalized**)*

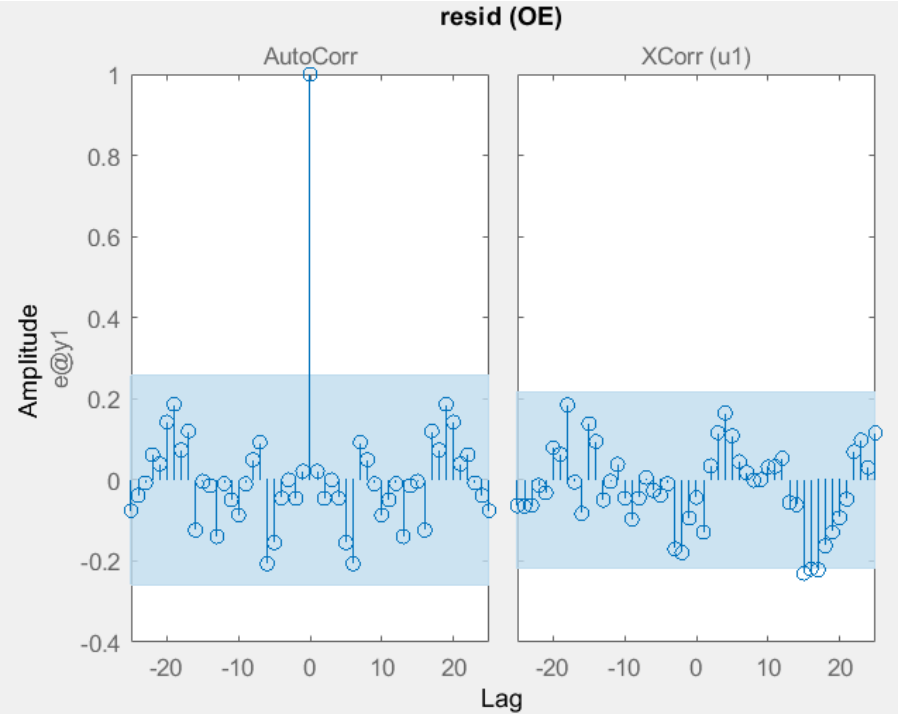

*Figure 114 residuals of OE model(**not normalized**)*

ARMAX:

```
Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)
  A(z) = 1 - 1.758 z^-1 + 1.091 z^-2 - 0.1807 z^-3

  B(z) = -0.09638 z^-1 + 1.236 z^-2

  C(z) = 1 - 1.689 z^-1 + 0.8603 z^-2

Sample time: 0.1 seconds

Parameterization:
   Polynomial orders:   na=3   nb=2   nc=2   nk=1
   Number of free coefficients: 7
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARMAX on time domain data "data1".
Fit to estimation data: 86.6% (prediction focus)
FPE: 0.9538, MSE: 0.9103
```

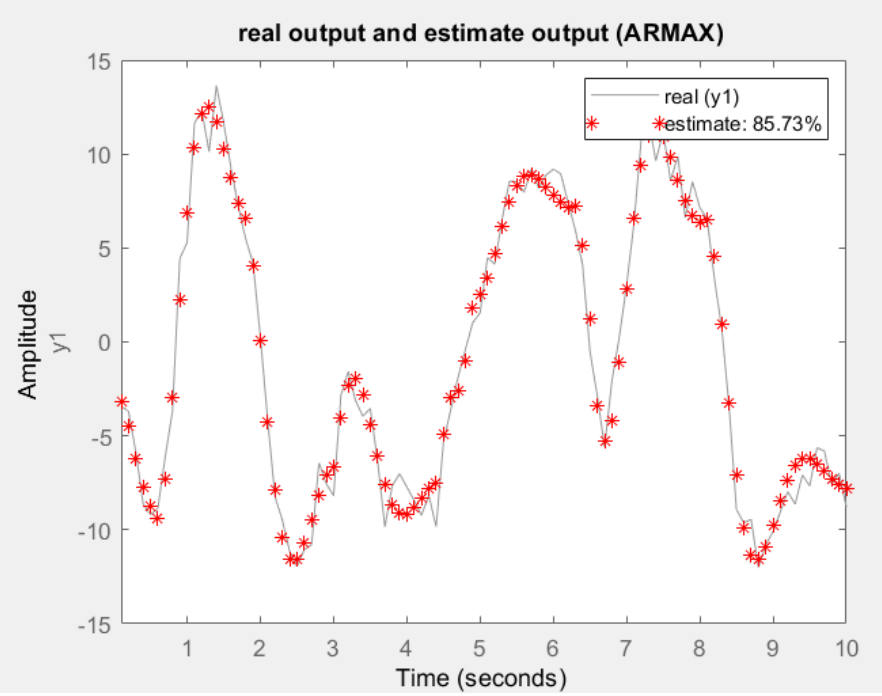

*Figure 115 OE model of the data (**not normalized**)*

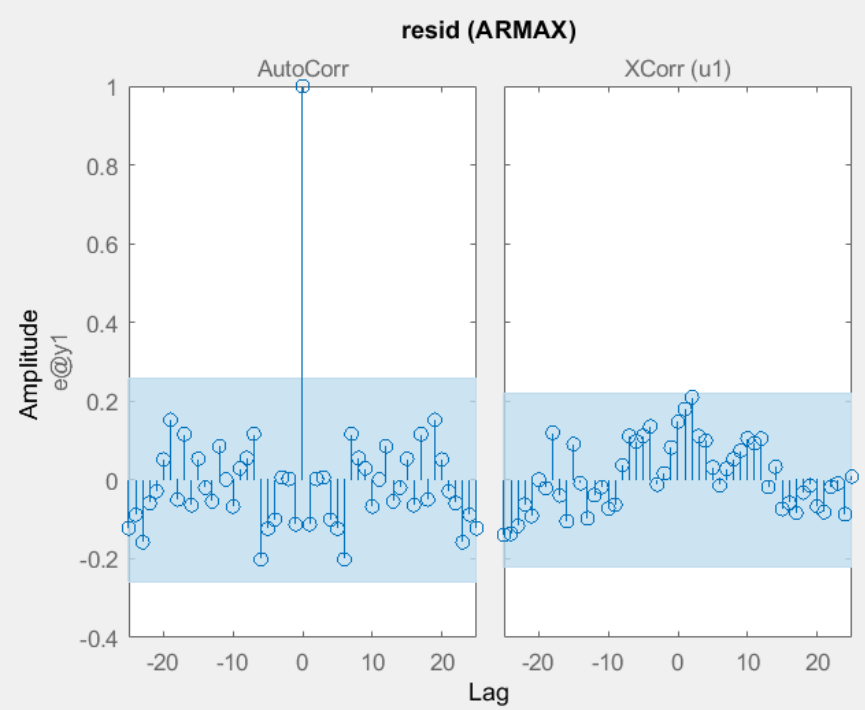

*Figure 116 residuals of ARMAX model (**not normalized**)*

The new results are much better, so the simplest model that satisfies the question's requirement is OE model. If we look at the residuals of OE method, it can be seen that almost all dynamics are within bound, and as the estimation perpetuates, the error is much more like the white noise, and it is empty of information.

The approach Dr. Aliyar invoked worked better with this problem☺

The initial parameters as:

$$n_a = 3; n_b = 2;\ n_k = 1;\ n_f = 3;\ n_c = 2$$

The assumption that the system is a closed loop (because the system is a real industrial system). **The simplest model with the best performance over 85% is OE model.**

Because the dataset is given with separated train-test components, we can manipulate the train test data. Therefor the train test ration is always constant.