ELEC2146 Project Report

The aim of this project is to provide a descriptive model of an unknown simulated system ('black box'), based on data collected from a simulation program. In the report three types of models will be used to describe the black box: (1) under damp second order circuit model, (2) AR model , (3) ARX model.

First of all we set the input as **impulse signal** and observe what the output looks like in Matlab:

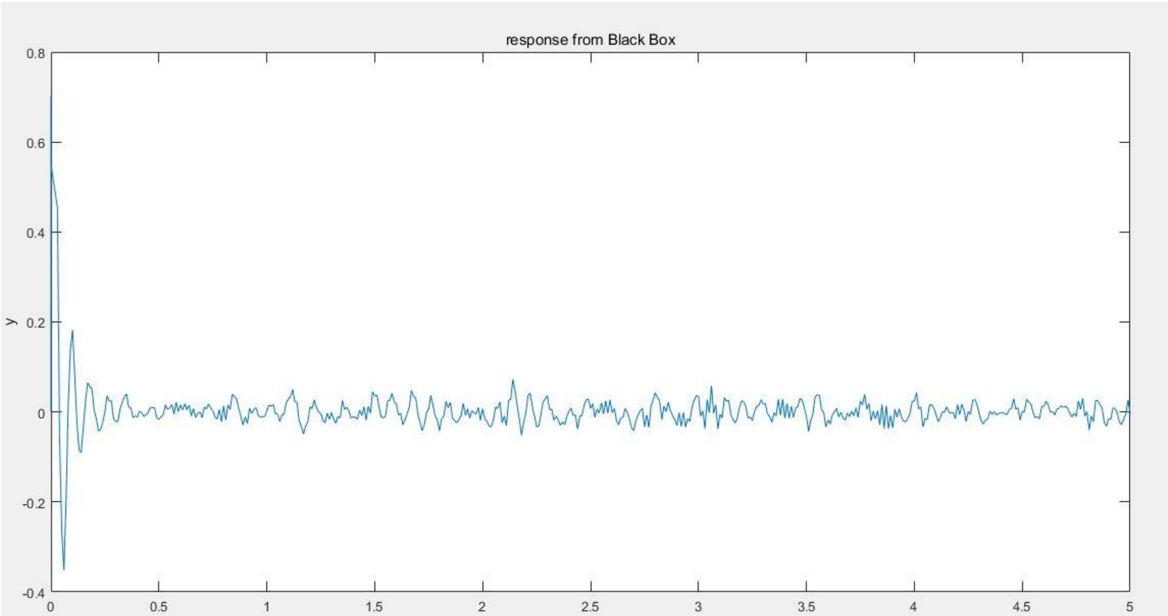


Figure 1

(1) Under damp model

As we can see the output is very similar to the under damp case second order circuit, so essentially the process of generating this model is to work out the parameters of **B*exp(-alpha*t)*cos(wd*t-theta)** by using the technique of least square.

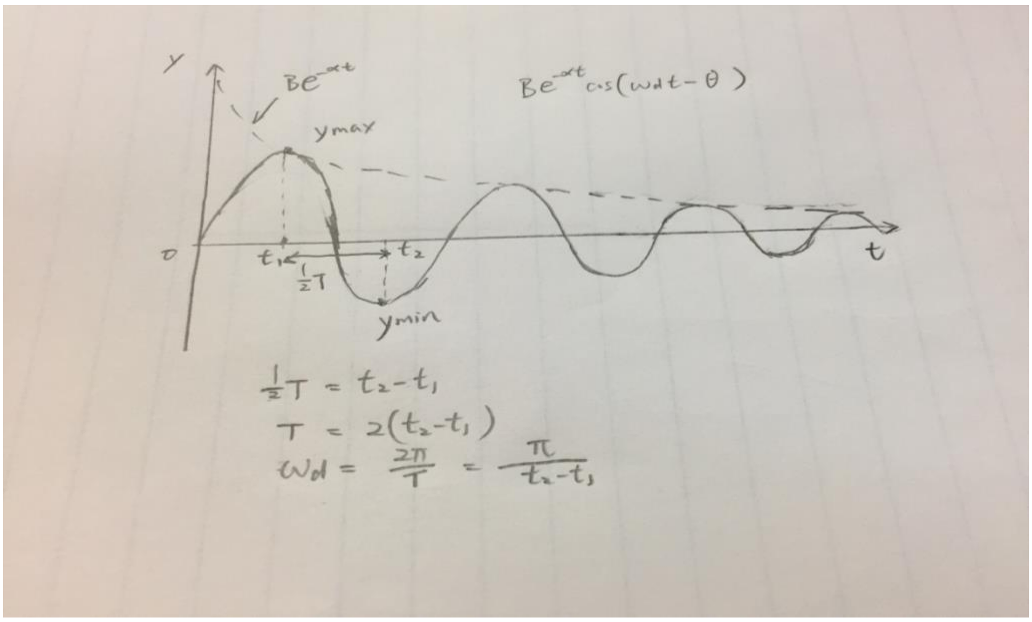


Figure 2

As shown in figure 2, we can find out the period T by subtract the t of max y (i.e. t 1) to the t of y min (i.e. t2): 0.5*T = t2 – t1 -> T = 2*(t2 – t1), **wd = 2pi/T = pi/(t2-t1)**.

If we pick out the peaks of the output then we obtain the **envelope** of the under damp model (i.e. B*exp(-alpha*t)). In Matlab code I only pick the first 7 peaks of y from the output because as time goes by the ‘peaks’ are actually just noise and won’t be helpful at all.

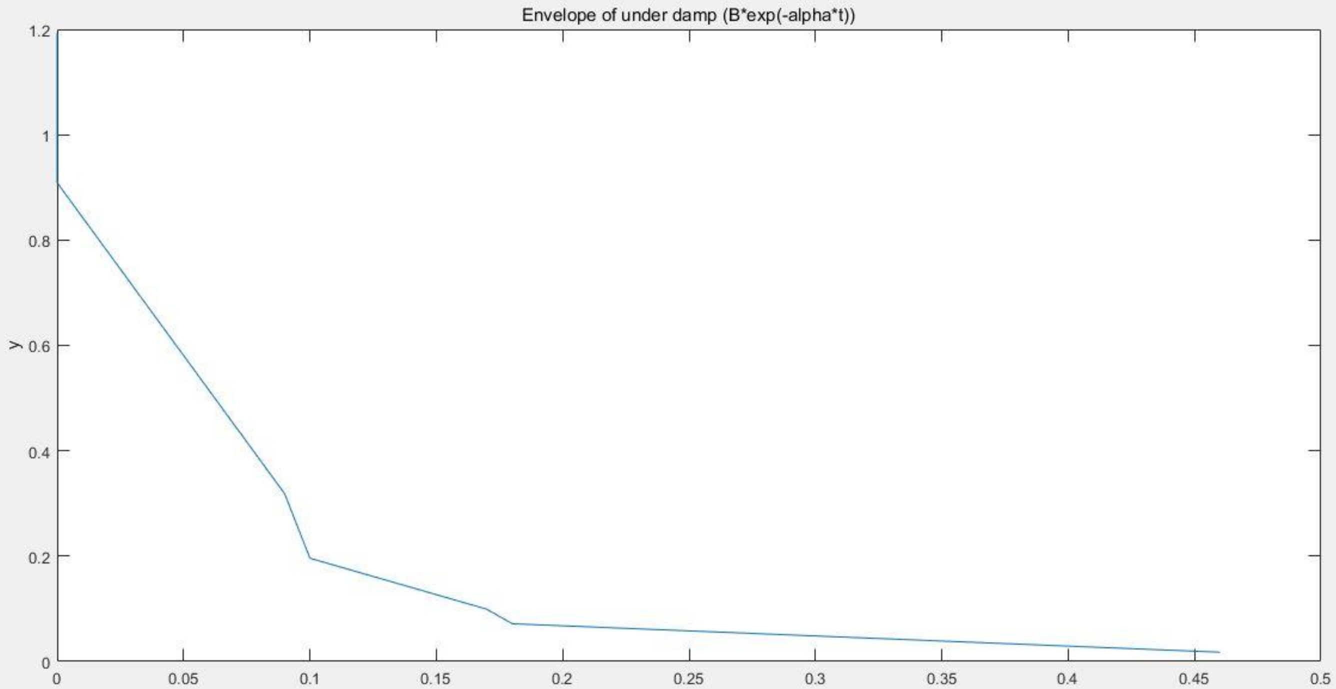


Figure 3 (envelope of under damp)

The envelope can be treated as the impulse response of first order circuit. Next step is try to linearise the equation y = B*exp (-alpha*t). Denote y1 = B*exp(-alpha*t1), y2 = B*exp(-alpha*t2), so y1/y2 = exp(-alpha(t1-t2)), log(y1/y2) = -alpha(t1-t2). Hence we let Y = log (y1/y2), X = (t1-t2), c = -alpha, c = inv (X’*X)*X’*Y. From Matlab we find out **alpha is around 11.5**, note that alpha might varies every time we run Matlab owing to the random error term inside Matlab. After we find out alpha B will be easy to figure out. Let Y = y , X = exp(-alpha*t), c = B. B = c = inv (X’*X)*X’*Y. From Matlab we find out **B is around 1.03.** The last step is to find out theta. We select an arbitrary point from output y, say the forth point (not four seconds) of output y, y (4) = B*exp (-alpha*t(4))*cos (wd*t(4)- theta), theta = wd*t(4) – cos-1 (y(4)/(B*exp(-alpha*t(4)))), we have theta is around 7.

Graph be low is the output we simulate using under damp model, with **MSE = 1.757945e-03**, which fits very well indeed.

Figure(4)

Plot output from black box and output from under damp model together to see how close they are:

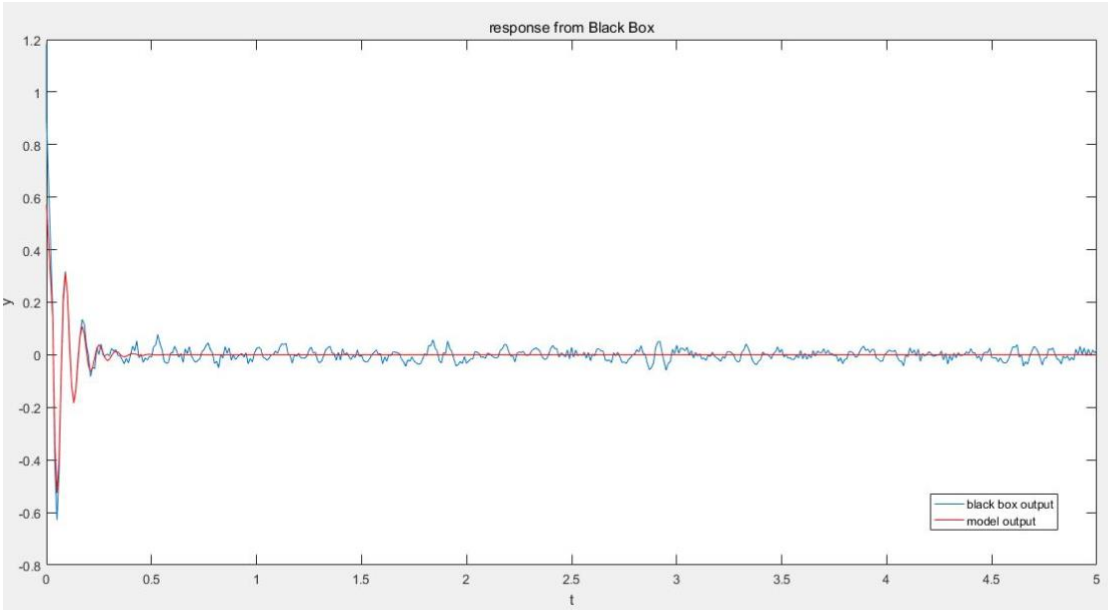
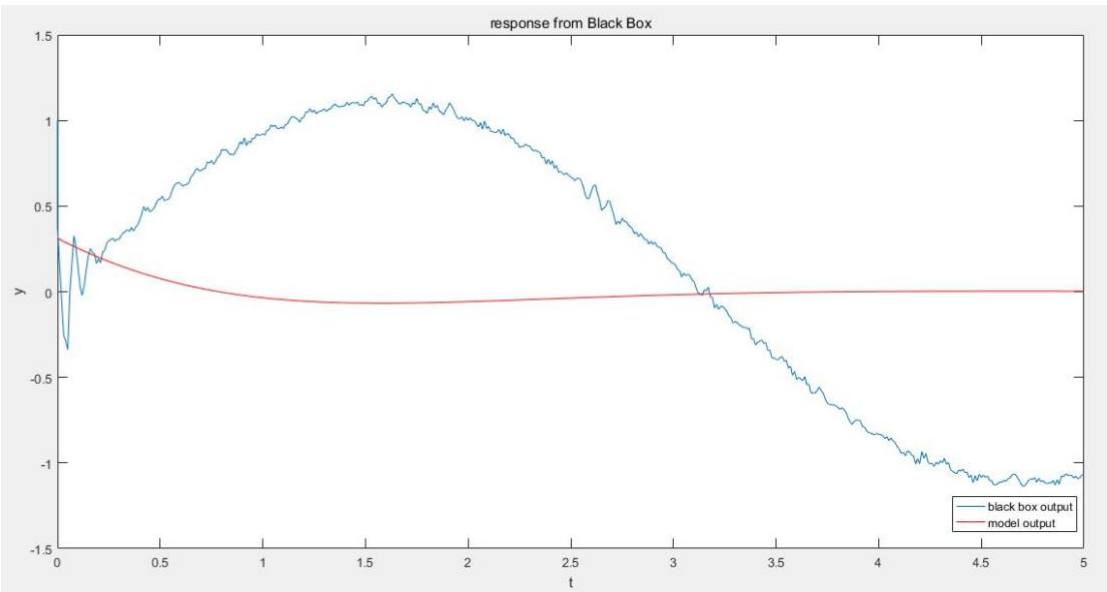


Figure (5)

Hence is that a good representation of the black box? What we were doing above is to build a model based on a assumed input (i.e. impulse), and the input of the black box is in fact arbitrary, then **if the model is a good representation of the black box then it shall fit other types of inputs as well** although we didn't build our model based on those inputs ! So let's see how our first model performs when u (t) now becomes sin (t):



Now the model doesn't fit the black box at all, so **we assert the under damp model is not an ideal model for the black box although it fits well when input is impulse signal, however it fails to keep its good performance when we feed in other types of input.**

(2) AR model

Now we consider another type of model: AR. The process is on the lecture slide:

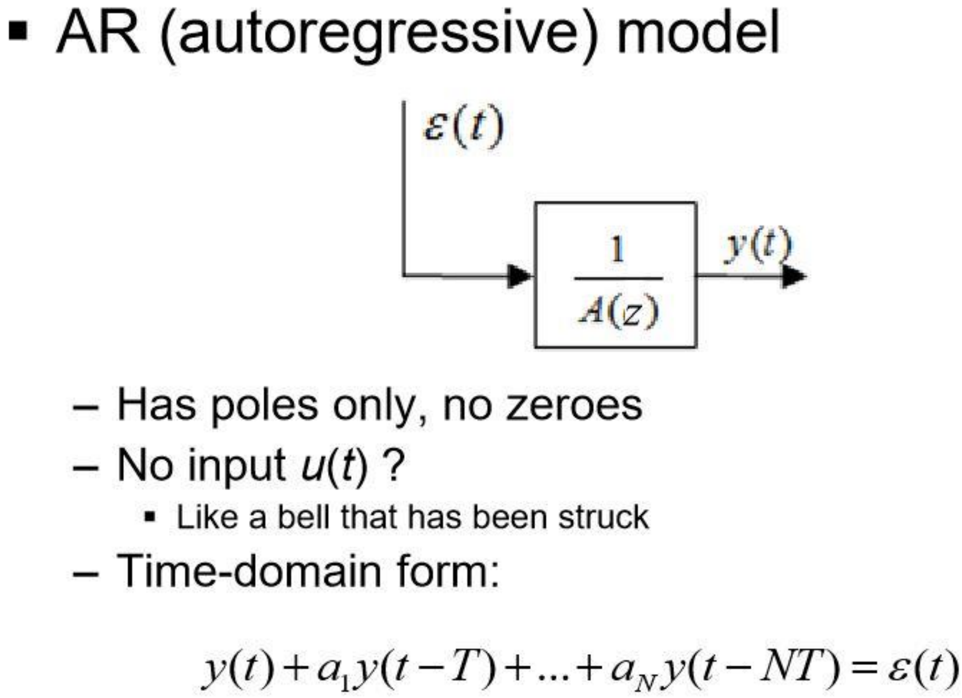
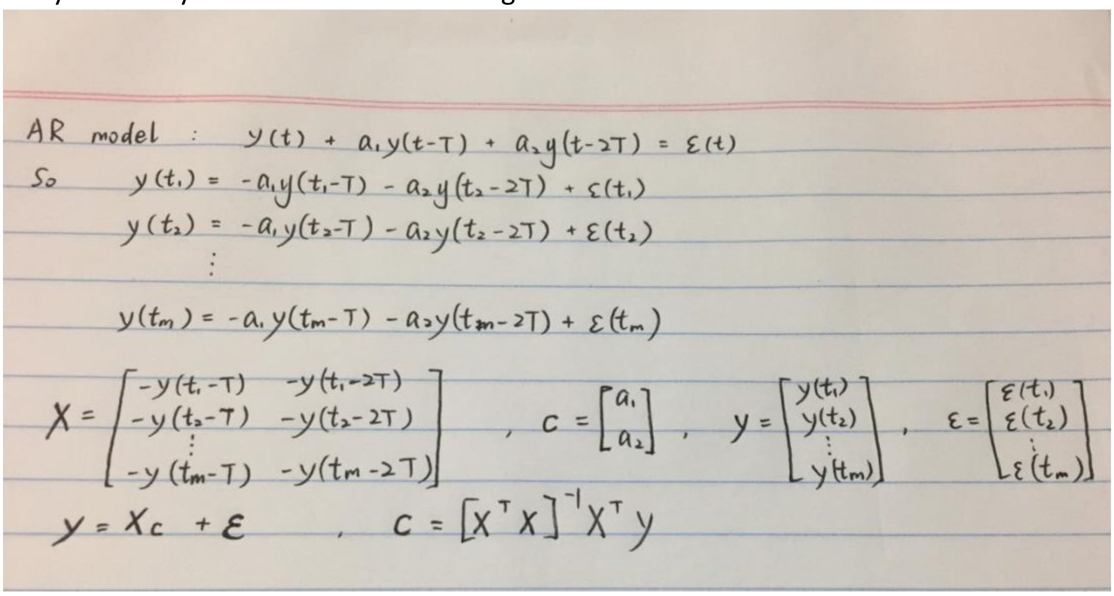


Figure (7)

Here for the sake of simplicity, we let N = 2 and T = 1, T here enables us to reference the delay value of y. The detail of the working is shown below:



AR model : $y(t) + a_1 y(t-T) + a_2 y(t-2T) = \varepsilon(t)$

So $y(t_1) = -a_1 y(t_1-T) - a_2 y(t_2-2T) + \varepsilon(t_1)$

$y(t_2) = -a_1 y(t_2-T) - a_2 y(t_2-2T) + \varepsilon(t_2)$

$\vdots$

$y(t_m) = -a_1 y(t_m-T) - a_2 y(t_m-2T) + \varepsilon(t_m)$

$$X = \begin{bmatrix} -y(t_1-T) & -y(t_1-2T) \\ -y(t_2-T) & -y(t_2-2T) \\ \vdots & \\ -y(t_m-T) & -y(t_m-2T) \end{bmatrix}, \quad c = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad y = \begin{bmatrix} y(t_1) \\ y(t_2) \\ \vdots \\ y(t_m) \end{bmatrix}, \quad \varepsilon = \begin{bmatrix} \varepsilon(t_1) \\ \varepsilon(t_2) \\ \vdots \\ \varepsilon(t_m) \end{bmatrix}$$

$$y = Xc + \varepsilon, \quad c = [X^T X]^{-1} X^T y$$

Figure (8)

Here we select the 10^{th},20^{th},30^{th},40^{th},50^{th},60^{th},70^{th},80^{th},90^{th},100^{th},200^{th} points of y (not seconds) to put into the matrix and generate c. From Matlab we got c = [-1.1038 0.3612]', which indicates a1 = -1.1038, a2 = 0.3612. For the estimated value of y, we then have y (t) = 1.1038*y (t-1) - 0.3612*y (t-2).

The graph below is the simulation of black box using AR model when input is impulse signal, the MSE is 1.067713e-03, which is pretty satisfactory:

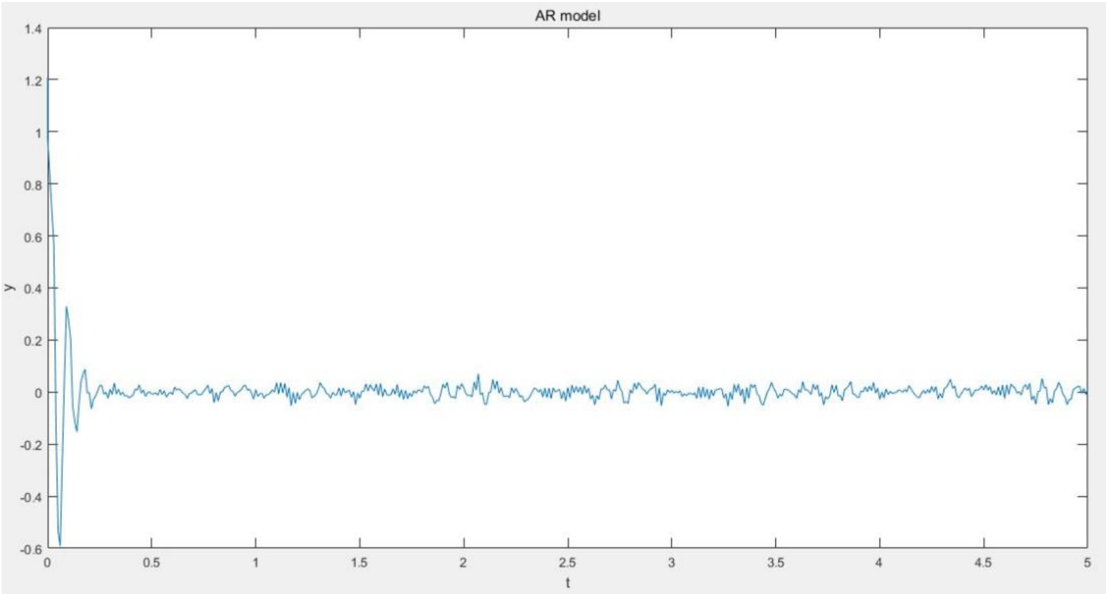
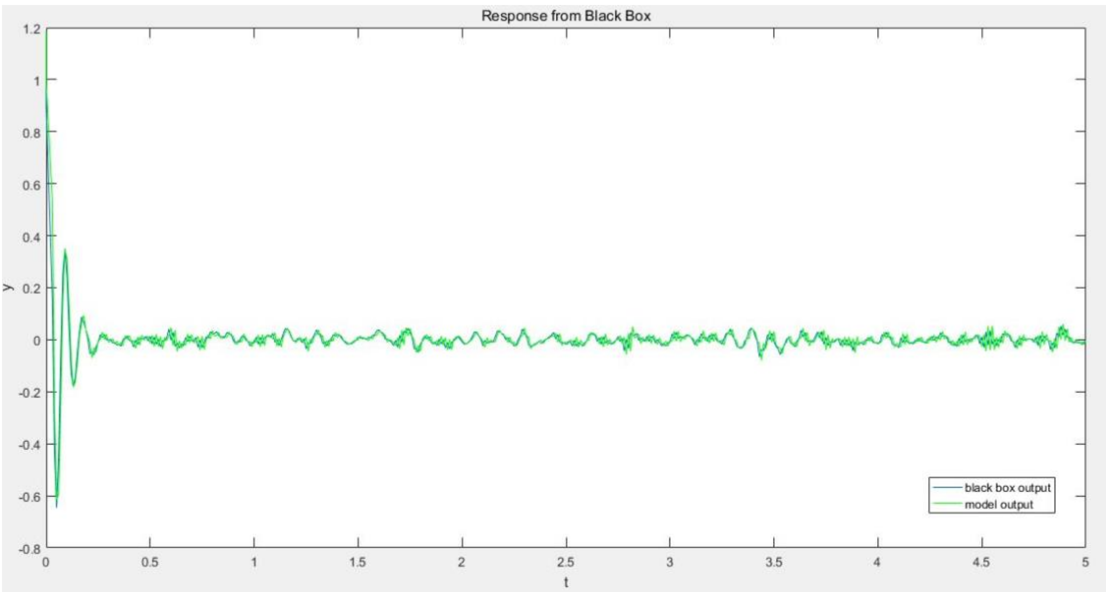
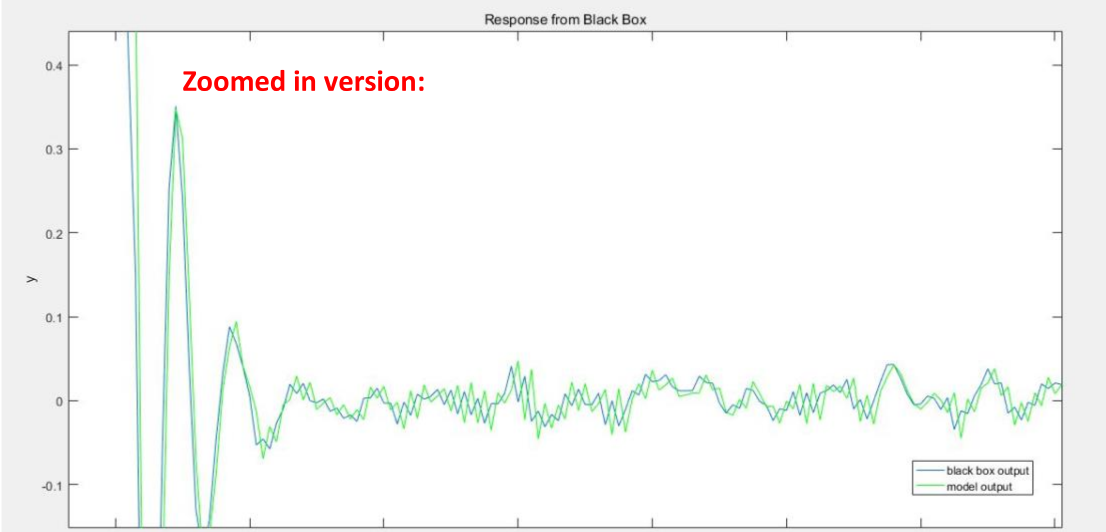


Figure (9)

Plot the output from black box and the output from AR model together to see how close they are:

Figure (10)

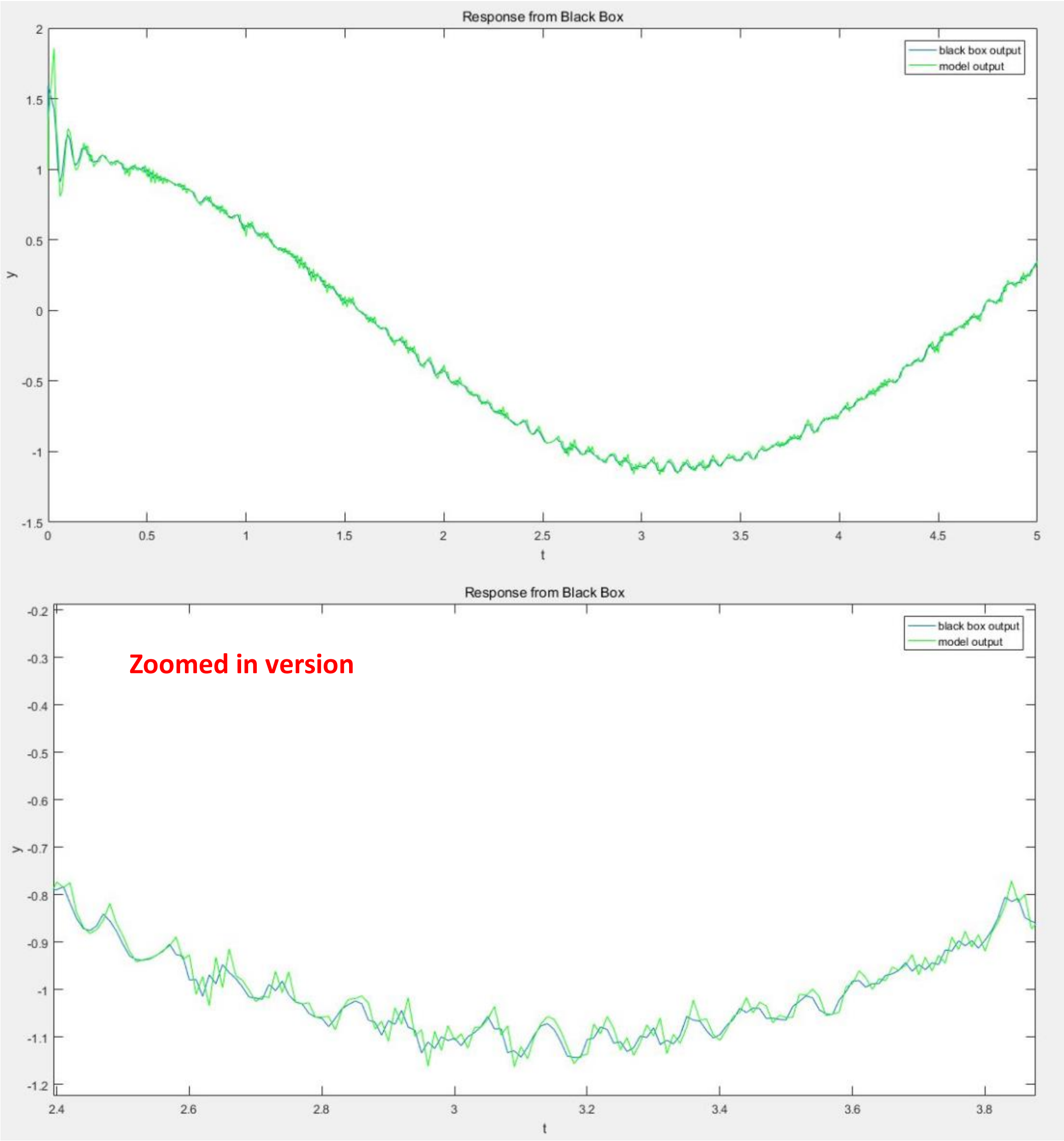




MSE = 1.405740e-03, we can see two outputs are so close to each other that they almost overlap. Hence at least when input is impulse we can see AR model works perfectly.

How about feed in other kinds of input, such as **u (t) = cos (t).**

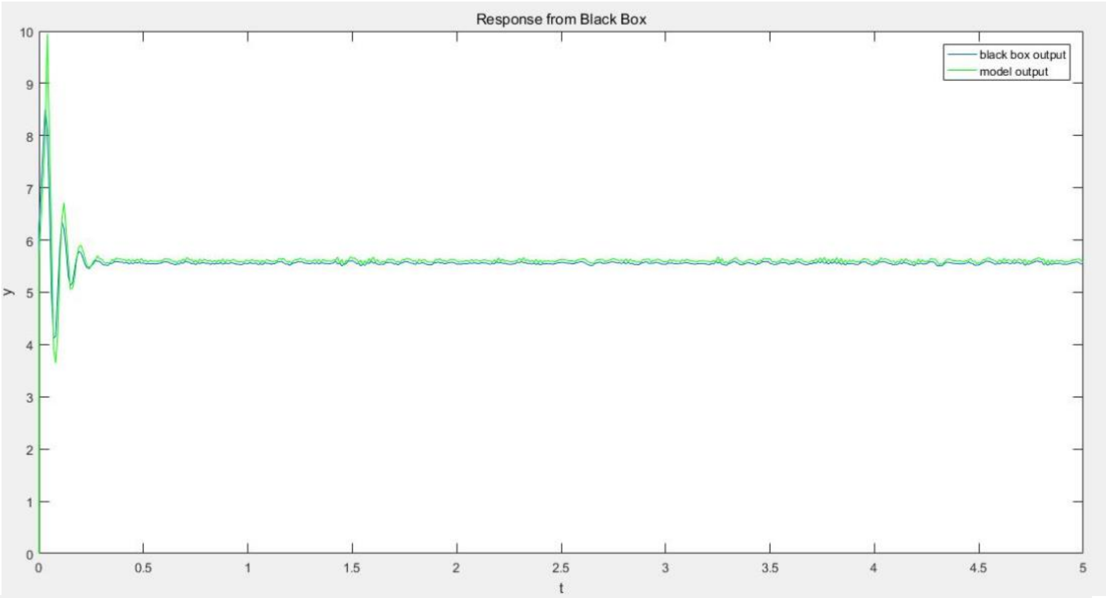
Figure (11)

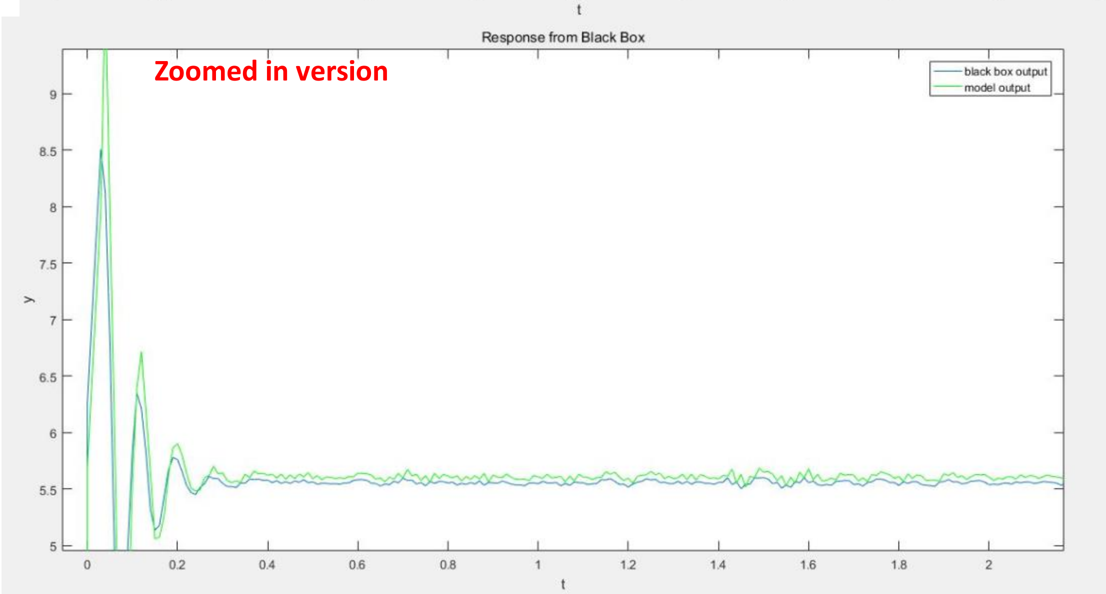


MSE = 4.091122e-03, AR model still works perfectly!

How about we feed in a constant value, say u (t) = 5, (expressed as `u = [5*ones(length(t))];`),and set y(1) = 0 in Matlab:

Figure (12)





MSE = 1.940940e-02, which is still not too bad.

After testing different types of data and witness AR model's good performance we can conclude that the **AR model we built is a good descriptive model of the black box.**

(3) ARX model

The basic logic of building ARX model is provided in lecture slide:

- ARX model
  - Suppose we take *m*+*N*-1 samples of the output and *m*+*M*-1 samples of the input, spaced by *T* (the sampling period)
    - $\rightarrow$ *m* equations

$$y(t_1)+a_1y(t_1-T)+\ldots+a_Ny(t_1-NT)=b_1u(t_1-T)+\ldots+b_Mu(t_1-MT)+\varepsilon(t_1)$$
$$y(t_2)+a_1y(t_2-T)+\ldots+a_Ny(t_2-NT)=b_1u(t_2-T)+\ldots+b_Mu(t_2-MT)+\varepsilon(t_2)$$
$$\vdots$$
$$y(t_m)+a_1y(t_m-T)+\ldots+a_Ny(t_m-NT)=b_1u(t_m-T)+\ldots+b_Mu(t_m-MT)+\varepsilon(t_m)$$

- ARX model

$$\mathbf{y} = \mathbf{X}\mathbf{c} + \boldsymbol{\varepsilon}$$

– where

$$\mathbf{X} = \begin{bmatrix} -y(t_1-T) & -y(t_1-2T) & \cdots & -y(t_1-NT) & u(t_1-T) & u(t_1-2T) & \cdots & u(t_1-MT) \\ -y(t_2-T) & -y(t_2-2T) & \cdots & -y(t_2-NT) & u(t_2-T) & u(t_2-2T) & \cdots & u(t_2-MT) \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ -y(t_m-T) & -y(t_m-2T) & \cdots & -y(t_m-NT) & u(t_m-T) & u(t_m-2T) & \cdots & u(t_m-MT) \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} y(t_1) \\ y(t_2) \\ \vdots \\ y(t_m) \end{bmatrix} \qquad \mathbf{c} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \\ b_1 \\ b_2 \\ \vdots \\ b_M \end{bmatrix} \qquad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon(t_1) \\ \varepsilon(t_2) \\ \vdots \\ \varepsilon(t_m) \end{bmatrix}$$

Figure (13)

Here impulse signal is still assumed to be the input signal. One important thing to keep in mind is u (t) = 0 when t>0, and **inv (X'*X)*X*Y has a solution if and only if there's no zero column in X.** For example, **let T = 1,**

Case 1: assume y(t) + a1*y(t-T) + a2*y(t-2*T) = b1*u(t-T)+b2*u(t-2*T) + e(t)

y(t) = -a1*y(t) – a2*y(t-2*T) + b1*u(t-T) + b2*u(t-2*T) + e(t)

Y = [y(20) y(30) y(40)]';

% 1st column 2nd column 3rd column 4th column

```
X1 = [-y(20-T) -y(20-2*T) u(20-T) u(20-2*T)];
X2 = [-y(30-T) -y(30-2*T) u(30-T) u(30-2*T)];
X3 = [-y(40-T) -y(40-2*T) u(40-T) u(40-2*T)];
X = [X1; X2; X3;];
```

Inv (X'X)*X'*Y will have no solution as the third and forth columns are zero columns. Intuitively this still make a lot of sense because if we do the matrix like that then from the matrix what we see is just the same as there's no input fed into the black box! This is to say the form and the points we choose have to represent the characteristic of the input. Generally we don't have to worry too much about that, but for impulse signal input if we don't include the first point then we fail to express its characteristic.

Case 2: assume y(t) + a1*y(t-T) + a2*y(t-2*T) = b1*u(t-2*T) + e(t)

y(t) = -a1*y(t-T) - a2*y(t-2*T) + b1*u(t-2*T) + e(t)

Y = [y(3) y(30) y(40)]';

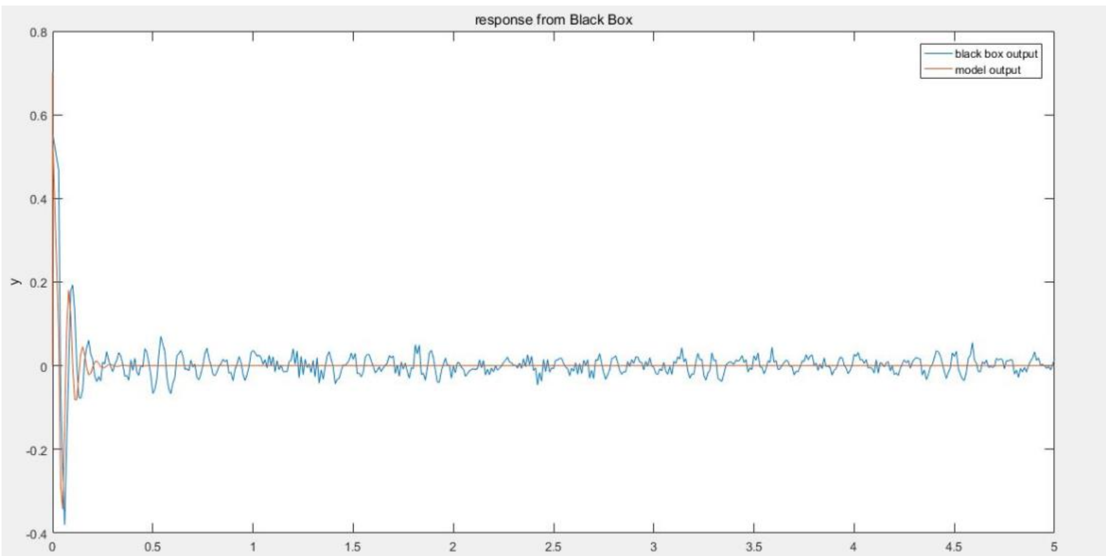
% 1st column 2nd column 3rd column

```
X1 = [-y(3-T)  -y(3-2*T)   u(3-2*T)];
X2 = [-y(30-T) -y(30-2*T) u(30-2*T)];
X3 = [-y(40-T) -y(40-2*T) u(40-2*T)];
X = [X1; X2; X3;];
```

In this case inv (X'X)*X'*Y will have a solution because the 3rd column now becomes [1 0 0]' from which we can tell it is a impulse signal, now all columns are non zero columns.
In Matlab code we may want to include more points into the matrix to make the model more accurate:

```
Y = [y(3) y(20) y(30) y(40) y(50) y(60) y(70) y(80) y(90) y(100) y(200)]';
X1 = [-y(3-T) -y(3-2*T) u(3-2*T)];
X2 = [-y(20-T) -y(20-2*T) u(20-2*T)];
X3 = [-y(30-T) -y(30-2*T) u(30-2*T)];
X4 = [-y(40-T) -y(40-2*T) u(40-2*T)];
X5 = [-y(50-T) -y(50-2*T) u(50-2*T)];
X6 = [-y(60-T) -y(60-2*T) u(60-2*T)];
X7 = [-y(70-T) -y(70-2*T) u(70-2*T)];
X8 = [-y(80-T) -y(80-2*T) u(80-2*T)];
X9 = [-y(90-T) -y(90-2*T) u(90-2*T)];
X10 = [-y(100-T) -y(100-2*T) u(100-2*T)];
X20 = [-y(200-T) -y(200-2*T) u(200-2*T)];
X = [X1;X2;X3;X4;X5;X6;X7;X8;X9;X10;X20];
c = inv(X'*X)*X'*Y;
```

c(1) = a1, c(2) = a2, c(3) = b1. As y(t) = -a1*y(t-T) - a2*y(t-2*T) + b1*u(t-2*T), we can get the estimated value of y(t) based on the previous value and the input. Finding all estimated values of y is an easy thing with the help of a for loop.
Let's see the performance of ARX model:



Figure(14)

MSE = 1.383484e-03, ARX works perfectly when input is impulse, although it can't model the noise of the black box as AR model do.
How about we change the input to u (t) = cos(t) ?

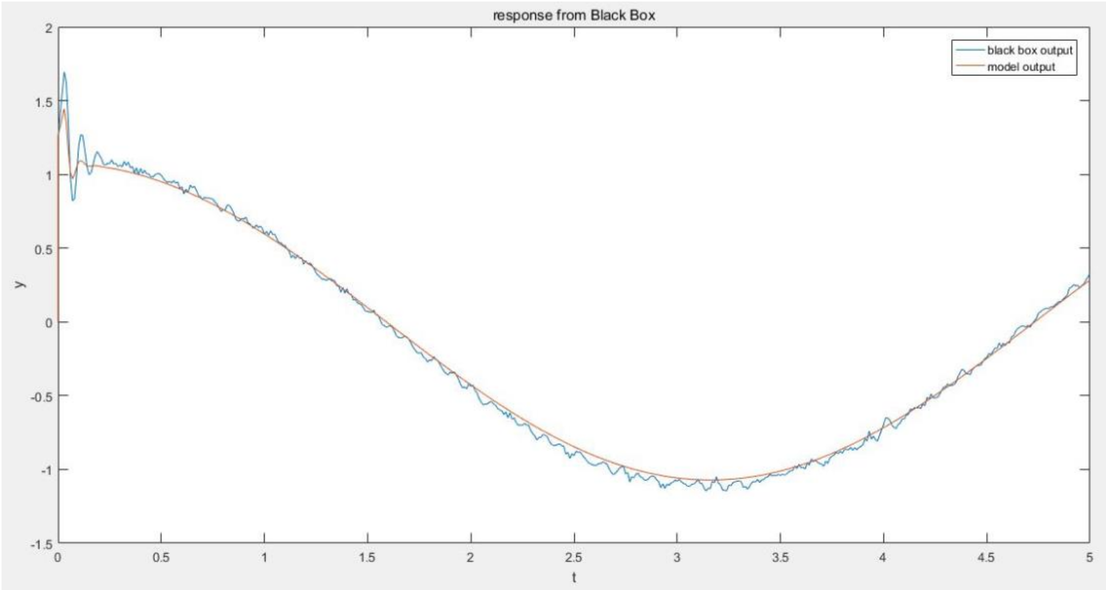


Figure (15)

MSE = 1.775922e-03 ARX model still works pretty welll.

Now let's feed in a constant input say u (t) = 5:

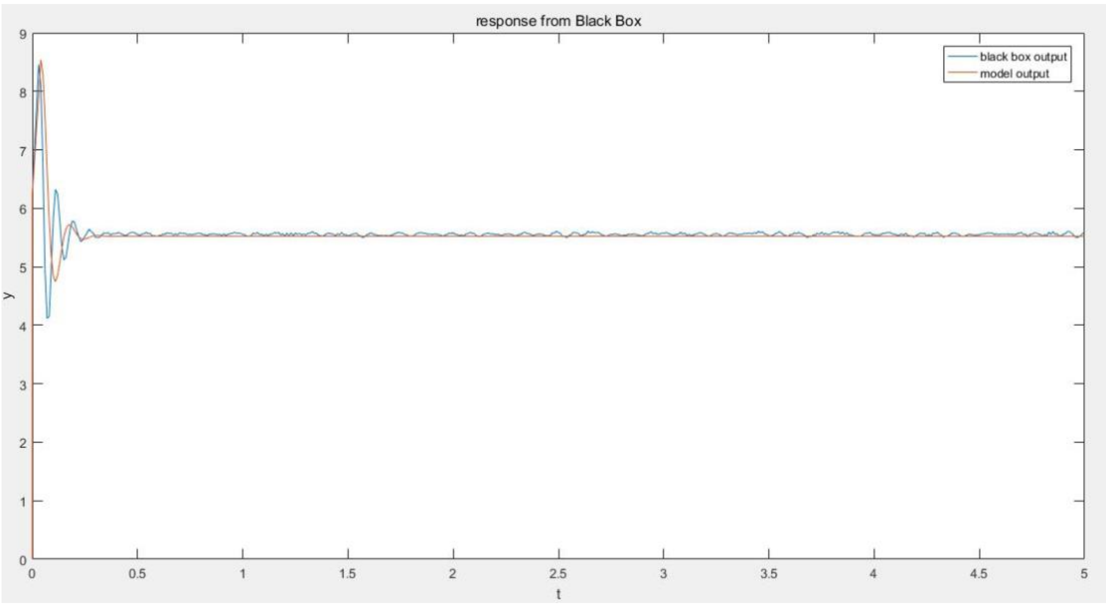


Figure (16)

MSE = 5.233388e-02, we can see some distinct deviation between two output but it is still acceptable.
After testing different types of data and witness ARX model's good performance we can conclude that the **ARX model we built is a good descriptive model of the black box.**

Conclusion

In conclusion, if we compare MSE of the models we built, we can see the AR model is the best descriptive model among the three as it has the smallest average MSE. ARX model is also a good model but not as good as AR model, however this is just talking about the general performance and doesn't mean AR model beats ARX in every kinds of input. **Also note that MSE might change every time we run Matlab owing to the random error inside the black box.** Under damp model is not a good model at all as it only works well when input is impulse signal, so we should not choose it as the descriptive model of the black box.

APPENDIX

project_underdamp.m
project_AR.m
project_ARX.m
blackbox.m
(matlab files will be uploaded)