System Identification

Project Assignment

Ambrus Bence-Lehel(as_lehel11x@yahoo.com)

December 17, 2017

Contents

1. Introduction

This MATLAB-based project assignment is a compulsory part of the System Identification course of the Technical University of Cluj-Napoca. The assignment consists of two problems, both using polynomial approximators. In the first problem the polynomial is used to model the behavior of an unknown function. The second problem consists of identifying a dynamic system with one input and one output.

The report is composed of the following sections:

- Fitting an unknown function
- Black-box system identification
- Appendix: Contains all written MATLAB .m files

2. Fitting an unknown function

A data set of input-output pairs is given; the outputs are generated by an unknown, nonlinear but static function f. In this assignment we will develop a model for this function, using a polynomial approximator. A second data sets are generated using the same function is provided for validation purposes.

2.1 The main script

This script will make use of the approximator function, which is used to approximate the unknown function given by the data structure and the grade of the polynomial. The two data sets are given as a MATLAB data file, one used for identification(id) and the other for validation puposes(val). Each of these structures contains the following field:

- A set of grid coordinates X for the inputs, where X is a cell array of two vectors, each vector *X{dim}* containing *n* grid points for input dimension *dim.*
- A set of corresponding outputs Y, a matrix of size n x n, where Y(I,j) is equal to the value of f at point (X{1} (i), X{2} (j)).
- The same data is provided in a different, 'flattened' format: the input Xflat, a wide matrix contains n^2 input points, one on each column, and the corresponding row vector of outputs Yflat.

```
3 -    load('proj_fit_14');
4      %load the input data
5 -    mesh(id.X{1},id.X{2},id.Y');
6      %mesh the input data
7 -    m=input('grade of polynomial m=');
8      %user choice given degree of the polynomial used for approximating the
9      %unknown function
```

This first part of the main script consists of loading the given data sets assigned by the lecturer, represent the unknown function based on these data sets and choosing the grade of the polynomial approximator, which is based on the user's choice, because the degree should be configurable in order to obtain better approximations.

For example, for the first few values of m, the approximator has the form:

m = 1, $g\hat{}(x) = [1, x1, x2] \cdot \theta = \theta1 + \theta2x1 + \theta3x2$

m = 2, $g\hat{}(x) = [1, x1, x2, x2\ 1\ , x2\ 2\ , x1x2] \cdot \theta = \theta1 + \theta2x1 + \theta3x2 + \theta4x\ 2\ 1 + \theta5x\ 2\ 2 + \theta6x1x2$

m = 3, $g\hat{}(x) = [1, x1, x2, x2\ 1\ , x2\ 2\ , x3\ 1\ , x3\ 2\ , x1x2, x2\ 1x2, x1x\ 2\ 2\ ] \cdot \theta$

Once the degree m has been chosen, model fitting consists of finding the optimal parameter vector θ so that g(x) best matches f(x) on the identification dataset, in a least-squares sense. That computation is made in the next section of code consisting in a loop which calls the '*proj_approx*' function for every polynomial degree from 1 to the inputted number and returns the Mean Squared Errors and the approximated outputs. After that we select the best match for our unknown function, based on the errors.

```
for i=1:m
     %computation of the mean sqaured error for both datasets(id,val)
    [MSEi(i,1) yhati{i}]=proj_approx(i,id);
    [MSEv(i,1) yhatv{i}]=proj_approx(i,val);
end
 [mMsev poz]=min(MSEv);%choose the best mean squared error and the grade of polynomial for this MSE
```

2.2 Function for getting the errors and the approximations

```
function [ MSE yhat ] = proj_approx(Grade,data)
%this function is used for computing the MSE and the approximated output of
%the system based on the datasets and the polinomial degree
```

This function is used to generate the polynomial approximator of any degree for any set of data given as input and returns the approximated output of the unknown function and its mean squared error. The way we generate the polynomial approximator is the following:

1. We generate the powers of the $x(x_1^1, x_2^1, x_1^2, x_2^2, \ldots, x_1^m, x_2^m)$

```
while n<=Grade
%generating the x^1...x^n powers
    for i=1:N
        if mod(j,2)==0%choose the position for x1 or x2 depending on the index of the column
        phi(i,j)=data.Xflat(1,i)^n;
        else
        phi(i,j)=data.Xflat(2,i)^n;
        end
    end
    if mod(j,2)==1
        n=n+1;
    end
    j=j+1;
end
```

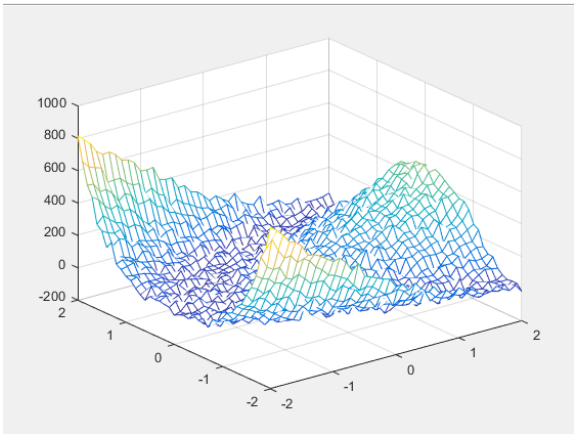
2. We generate the multiplications between these powers using the columns we generated at point 1.

```
a=2;b=3;%a and b represents the x1 and x2 and their grades used below for multiplication
for i=j:sum(1:n)
 %generating the multiplications between the x1 and x2 variation based on
 %the degree(x1*x2,x1*x2^2...)
    for k=1:N
        phi(k,i)=phi(k,a)*phi(k,b);
    end
    if a+b<2*n-1%if the grade of the two x is smaller then the grade n than raise the x2 grade by one
            b=b+2;
    else%starts over and raise the x1 grade by one
            b=3;
            a=a+2;
    end
end
```

In the last part of our function we make the computation for the coefficients column vector using backslash and finally computing the output and MSE.

2.3 Validation of the function

In order to check if our generated model is correct we need to test it on a different dataset, which is why in the main for we call the function for id and val.



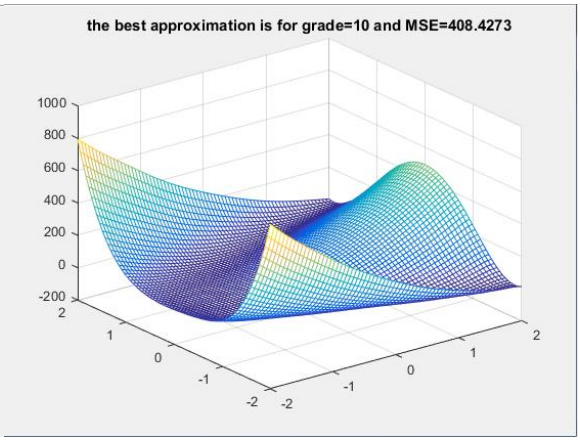


Fig.a Fig.b

As we can see above we have fig.a which is the unknown data with noise and at fig.b we have the best fit approximation for m=10.

2.4 Results and discussions

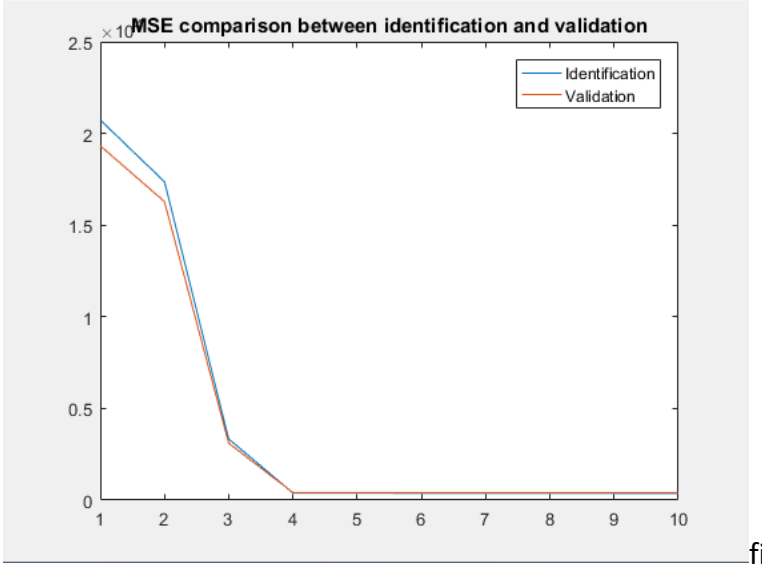


fig.c

In fig.c we can observe the evolution of the approximation because we plotted the MSE values for each grade of the polynomial till the inputted one and we can observe that until the polynomial of grade m=4 the approxiamtor is really different from the given data and after m=4 the approximator varies very little. Of course if we check for a higher order than 10 will get some values for which the approximated output will be over fitted.

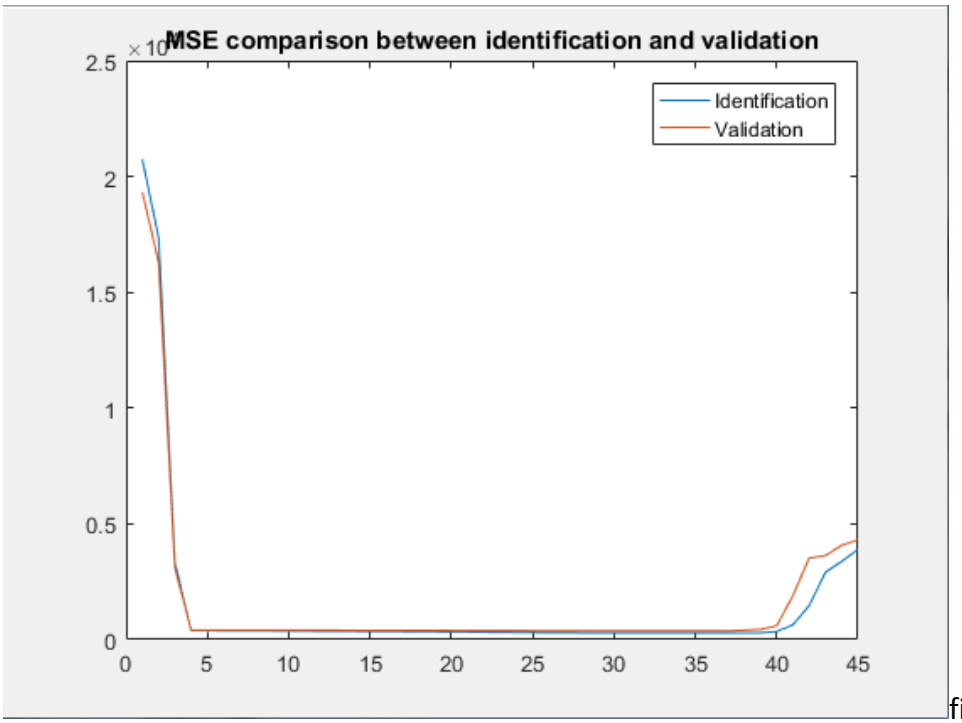


fig.d

In fig.d we can observe that for m=37 the output is getting over fitted, so that the MSE is getting bigger and in fig.e we can see the approximated output for m=37.

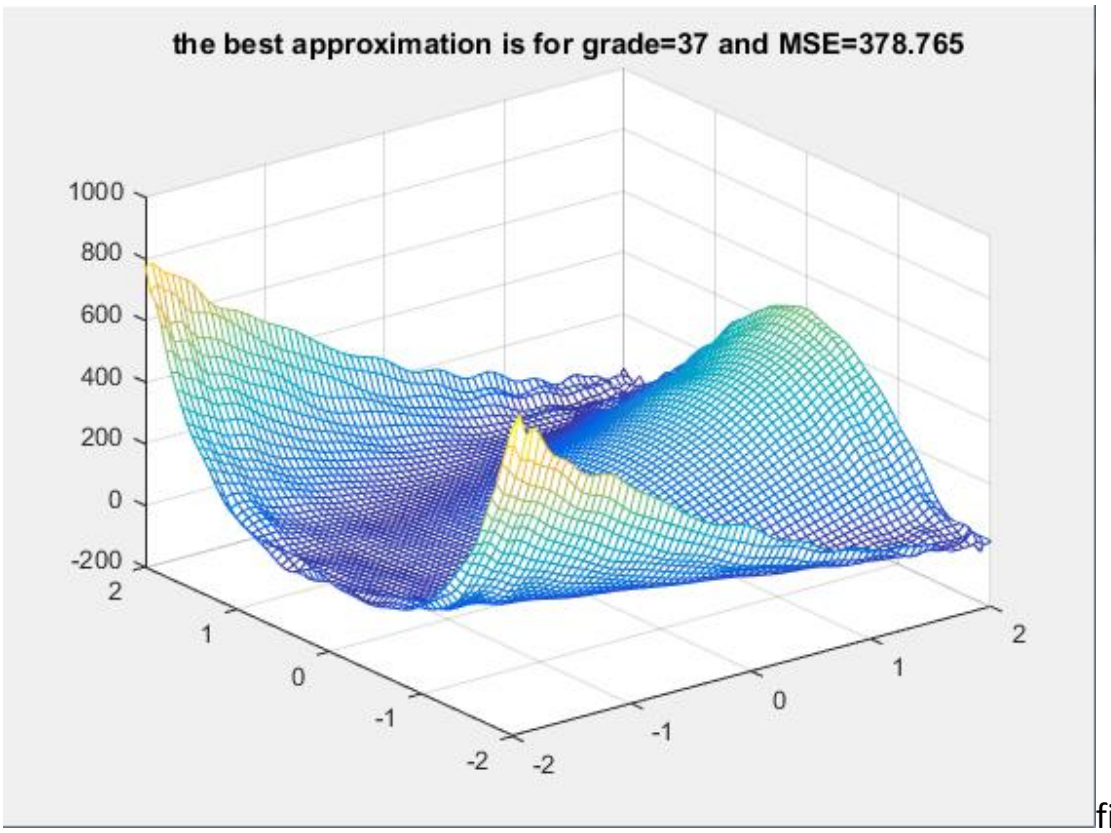


fig.e

3. Black-box system identification

The problem consists in finding a model for an unknown dynamic system with one input and one output. As in the previous problem we have a dataset given which is the input and the output of the unknown system. The task is to develop a black-box model for this system, using a polynomial, nonlinear ARX model.

3.1. The main script

```
load('iddata-19');
plot(id.u);hold on
plot(id.y,'r');
legend('Input signal','Output');
m=input('Grade of the polynomial:');
na=input('na=');
nb=input('nb=');
```

The first steps made in the script we load the data and plot it in order to see the behavior of the system and then request the user to input the grade of the polynomial and the na,nb constants used in the ARX method.

```
[yhat, yhats, msep, mses]=my_arx(na,nb,m,id,val);
```

The next step is calling the function for the inputted values and the dataset and the output parameters represent the approximated outputs for both prediction and simulation and their errors.

Finally, the last steps are representing the approximated outputs of the system with the real outputs.

```
figure
plot(yhat);hold
plot(val.y,'r');
title('One-step-ahed prediction');
legend('Approximated output','Real Output');
figure
plot(yhats);hold
plot(val.y,'r');
title('Simulation');
legend('Approximated output','Real Output');
```

3.2. The function which creates a model

The function works based on an algorithm presented by the lecturer; in the first part we implement a 'power' matrix of dimensions (m+1)^(na+nb)X(na+nb) with all the possible variations from 0 to the grade of the polynomial.

```
b=m+1;
dimr=b^(na+nb);
dimc=na+nb;
W=zeros(dimr,dimc);
for j=dimc:-1:1
    i=2;
    d=1;
    while i<=dimr
        if W(i-1,j)<m && j==dimc
            W(i,j)=W(i-1,j)+1;
        elseif mod(d,b^(dimc-j))>0 && j<dimc
            W(i,j)=W(i-1,j);
            d=d+1;
        elseif mod(d,b^(dimc-j))==0 && j<dimc && W(i-1,j)<m
            W(i,j)=W(i-1,j)+1;
            d=1;
        elseif mod(d,b^(dimc-j))==0 && j<dimc && W(i-1,j)==m
            W(i,j)=0;
            d=d+1;
        end
        i=i+1;
    end
end
```

My method of implementing this matrix was to have basically three phases: 1. When the element in a position needs to increase from the previous element; 2. When the element needs to remain at the same value as the previous one; 3.When the elements reach the grade of the polynomial they need to start over.

We use this power matrix in order to get the ARX model based on a vector d=[-y(k-1)…-y(k-na) u(k-1)…u(k-nb)].After this we add 1 to each column separately which represents the multiplication with the y(k-1)…y(k-na) and u(k-1)…u(k-nb);

```
for j=1:dimc
    i=1;
    while i<=length(W)
        Wa(i+(j-1)*length(W),j)=W(i,j)+1;
        Wa(i+(j-1)*length(W),(j+1):dimc)=W(i,(j+1):dimc);
        if j>1
            Wa(i+(j-1)*length(W),1:j-1)=W(i,1:j-1);
        end
        i=i+1;
    end
end
```

The next step is to eliminate the rows which have a sum greater than the grade of the polynomial+1.

```
l=length(Wa);
i=1;
while i<=l
    if(sum(Wa(i,:))>m+1)
        Wa(i,:)=[];
        l=l-1;
    else
        i=i+1;
    end
end
```

The next step is to collect all the rows which are the same in order to obtain a well-conditioned model, without linearly dependent regressors.

```
Wb=unique(Wa,'rows','stable');
```

The last thing we use this matrix is to bring the d vector to the corresponding degree so we would obtain a polynomial ARX equation.

```
for i=2:length(id.u)
    for j=1:dimc
        if((i-j)<=0) && (i-j+na<=0)
            d(i,j)=0;
        elseif(j<=na) &&((i-j)>0)
            d(i,j)=-id.y(i-j);
        elseif(j>na)
            d(i,j)=id.u(i-j+na);
        end
    end
end
for i=1:length(id.u)
    for k=1:length(Wb)
            phi(i,k)=prod(d(i,:).^Wb(k,:));
    end
end
theta=phi\id.y;
```

This is the method for one-step-ahead prediction which consist basically in the knowledge of the real output. The other method is the simulation when we recursively use the approximated output in order to approximate the real one.

3.3. Results and discussions

First of all there is a big difference between the two methods mentioned earlier, so the expectations are that they are going to have different quality for the fit.

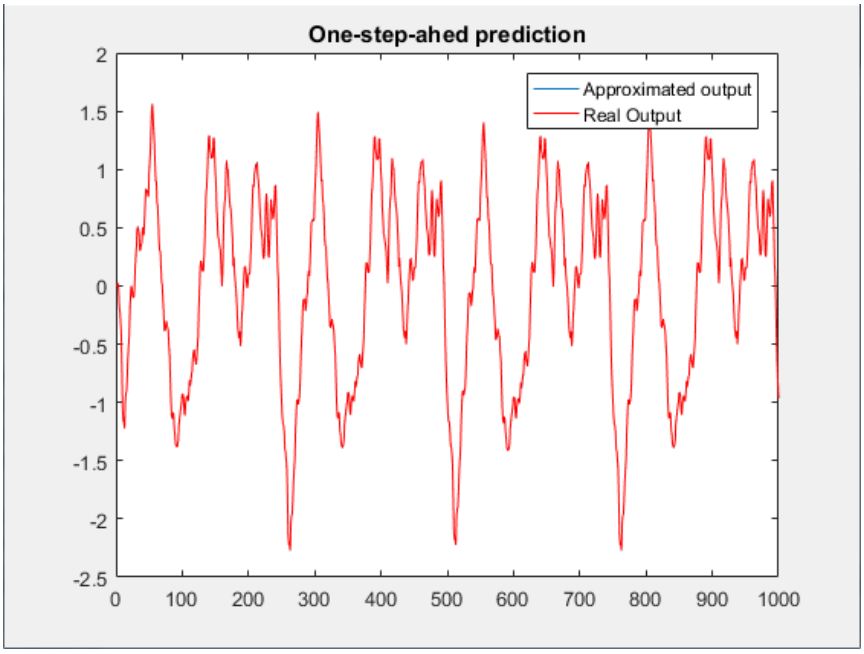


Fig.1

As we can see in Fig.1 the approximated output matches the real output, but looking at Fig.2 we can see that the first method gives us better fit.

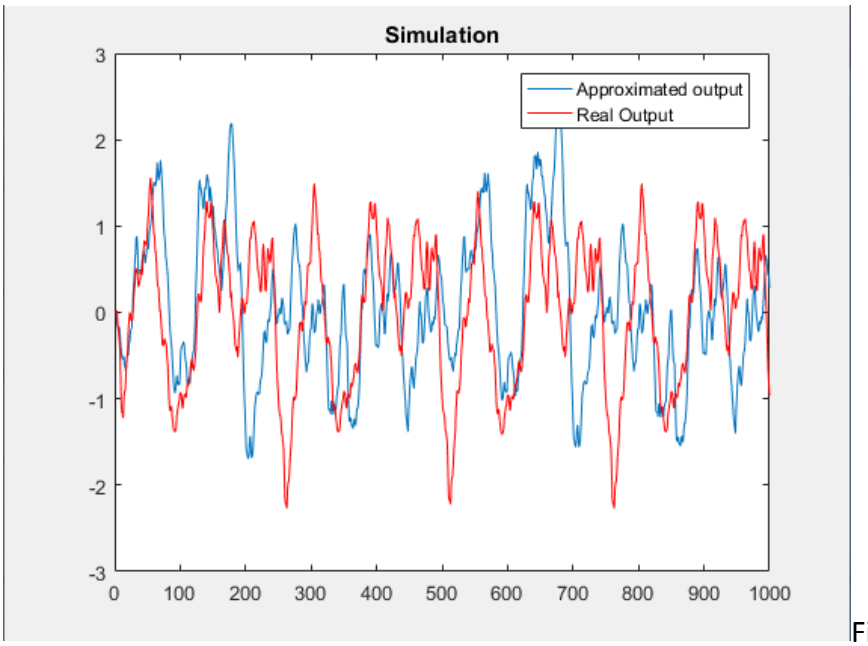


Fig.2

4. Appendices

For all Matlab-files Matlab version R2016b was used.

A.0 Main script

```
clc
clear all
load('proj_fit_14');
%load the input data
mesh(id.X{1},id.X{2},id.Y');
%mesh the input data
m=input('grade of polynomial m=');
%user choice given degree of the polynomial used for approximating the
%unknown function
for i=1:m
    %computation of the mean sqaured error for both datasets(id,val)
   [MSEi(i,1) yhati{i}]=proj_approx(i,id);
   [MSEv(i,1) yhatv{i}]=proj_approx(i,val);
end
[mMsev poz]=min(MSEv);%choose the best mean squared error and the grade of polynomial for this MSE
figure
plot(MSEi(:,1))
hold
plot(MSEv(:,1))
title('MSE comparison between identification and validation');
legend('Identification','Validation');
Yhat=reshape(yhatv{poz},val.dims);%Reshape the yhat vector in order to obtain the matrix for the mesh
figure
mesh(val.X{1},val.X{2},Yhat');
title(['the best approximation is for grade=',num2str(poz),' and MSE=',num2str(mMsev)]);
```

A.1 Function proj_approx

```
function [ MSE yhat ] = proj_approx(Grade,data)
%this function is used for computing the MSE and the approximated output of
%the system based on the datasets and the polinomial degree
N=length(data.Yflat);
%determine the number of rows of the phi matrix
phi(1:N,1)=1;%the first column is always filled with 1
j=2;n=1;%initializing the column and row
while n<=Grade
%generating the x^1...x^n powers
    for i=1:N
        if mod(j,2)==0%choose the position for x1 or x2 depending on the index of the column
        phi(i,j)=data.Xflat(1,i)^n;
        else
        phi(i,j)=data.Xflat(2,i)^n;
        end
    end
    if mod(j,2)==1
        n=n+1;
    end
    j=j+1;
end
a=2;b=3;%a and b represents the x1 and x2 and their grades used below for multiplication
for i=j:sum(1:n)
 %generating the multiplications between the x1 and x2 variation based on
 %the degree(x1*x2,x1*x2^2...)
    for k=1:N
        phi(k,i)=phi(k,a)*phi(k,b);
    end
    if a+b<2*n-1%if the grade of the two x is smaller then the grade n than raise the x2 grade by one
            b=b+2;
    else%starts over and raise the x1 grade by one
            b=3;
            a=a+2;
    end
end
theta=phi\data.Yflat';%create the coefficients column vector
yhat=phi*theta;%create the approximated column vector
MSE=sum((yhat'-data.Yflat).^2)/N;%calculating the mean squared error for the given dataset
end
```

A.2. Second part Script

```
clear all
clc
load('iddata-19');
plot(id.u);hold on
plot(id.y,'r');
legend('Input signal','Output');
m=input('Grade of the polynomial:');
na=input('na=');
nb=input('nb=');
[yhat, yhats, msep, mses]=my_arx(na,nb,m,id,val);
figure
plot(yhat);hold
plot(val.y,'r');
title('One-step-ahed prediction');
legend('Approximated output','Real Output');
figure
plot(yhats);hold
plot(val.y,'r');
title('Simulation');
legend('Approximated output','Real Output');
```

A.3. My_arx function

```
function [ yhat, yhats, MSEp, MSEs] = my_arx( na,nb,m,id,val)
b=m+1;
dimr=b^(na+nb);
dimc=na+nb;
W=zeros(dimr,dimc);
for j=dimc:-1:1
    i=2;
    d=1;
    while i<=dimr
        if W(i-1,j)<m && j==dimc
            W(i,j)=W(i-1,j)+1;
        elseif mod(d,b^(dimc-j))>0 && j<dimc
            W(i,j)=W(i-1,j);
            d=d+1;
        elseif mod(d,b^(dimc-j))==0 && j<dimc && W(i-1,j)<m
            W(i,j)=W(i-1,j)+1;
            d=1;
        elseif mod(d,b^(dimc-j))==0 && j<dimc && W(i-1,j)==m
            W(i,j)=0;
            d=d+1;
        end
        i=i+1;
    end
end
for j=1:dimc
    i=1;
    while i<=length(W)
        Wa(i+(j-1)*length(W),j)=W(i,j)+1;
        Wa(i+(j-1)*length(W),(j+1):dimc)=W(i,(j+1):dimc);
        if j>1
            Wa(i+(j-1)*length(W),1:j-1)=W(i,1:j-1);
        end
        i=i+1;
    end
end

l=length(Wa);
i=1;
while i<=l
    if(sum(Wa(i,:))>m+1)
        Wa(i,:)=[];
        l=l-1;
    else
        i=i+1;
    end
end
Wb=unique(Wa,'rows','stable');
d=zeros(length(id.u),dimc);
for i=2:length(id.u)
    for j=1:dimc
        if((i-j)<=0) && (i-j+na<=0)
            d(i,j)=0;
        elseif(j<=na) &&((i-j)>0)
            d(i,j)=-id.y(i-j);
        elseif(j>na)
            d(i,j)=id.u(i-j+na);
        end
    end
end
for i=1:length(id.u)
    for k=1:length(Wb)
            phi(i,k)=prod(d(i,:).^Wb(k,:));
```

```
    end
end
theta=phi\id.y;
for i=2:length(val.u)
    for j=1:dimc
        if((i-j)<=0) && (i-j+na<=0)
            dh(i,j)=0;
        elseif(j<=na) &&((i-j)>0)
            dh(i,j)=-val.y(i-j);
        elseif(j>na)
            dh(i,j)=val.u(i-j+na);
        end
    end
end
for i=1:length(val.u)
    for k=1:length(Wb)
            phiv(i,k)=prod(dh(i,:).^Wb(k,:));
    end
end
yhat=phiv*theta;
MSEp=(sum(yhat-val.y)^2)/length(val.y);
yhi=zeros(length(id.u),na+nb);
yhats(1)=0;
for i=2:length(id.u)
    for j=1:na+nb
        if((i-j)<=0) &&(i-j+na<=0)
            yhi(i,j)=0;
        elseif(j<=na) &&((i-j)>0)
            yhi(i,j)=yhats(i-j);
        elseif(j>na)
            yhi(i,j)=id.u(i-j+na);
```

```
        end
    end
    for l=1:length(id.u)
    for k=1:length(W)
            yhit(l,k)=prod(yhi(l,:).^W(k,:));
    end
    end
    yhats(i,1)=phi(i,:)*theta;
end
MSEs=(sum(yhats-val.y)^2)/length(val.y);
end
```