Advanced Techniques for System Identification and Validation in Complex Environments

Author: Tudor-Cristian Sîngerean

1. Project description

The objective of this project is to develop a nonlinear ARX model that captures the relationship between the system's input and output signals. This model, based on a linear-in-parameters structure with a polynomial form, is characterized by parameters ***na***, ***nb***, ***nk*** and ***m***.

The code utilizes MATLAB to approach, identify and approximate a dynamic system using a model with a polynomial structure. The choice of parameters ***na***, ***nb***, ***nk***, and ***m*** allows for flexibility in sketching the model to specific system characteristics.

The project focuses on building and validating a system model based on input-output data. The identified model is then used for prediction and simulation, and the performance is evaluated through mean squared errors. The system could be a dynamic system in a control or industrial process, and the project aims to develop a mathematical representation of its behavior.

2. Implementation

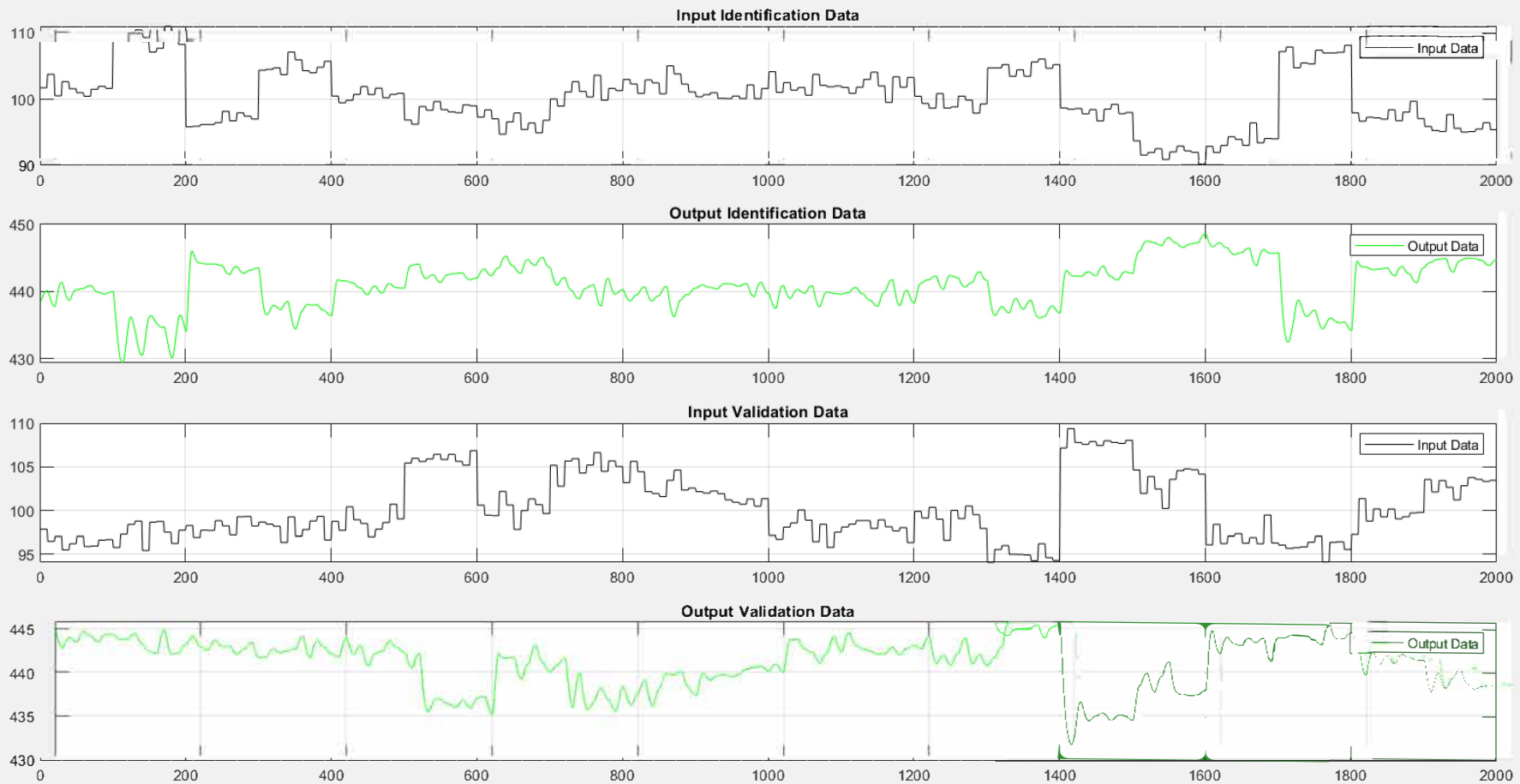
For developing a nonlinear ARX model using the identification data, constructing the regressor line based on past input and output values and also the design matrix '**ϕ**' for identification and simulation is considered to be an optimal approach.

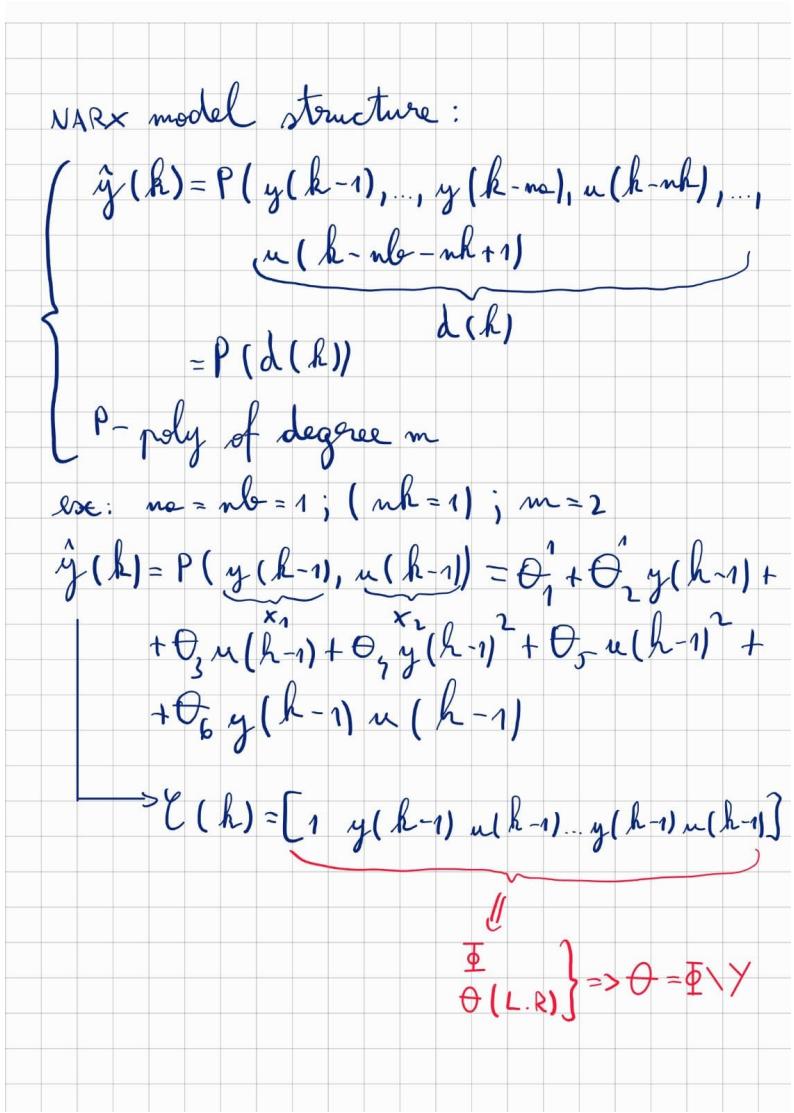
The model is based on two main functions which will be used in order to create the design matrices.

The '**phi_line_function**' is a recursive function that constructs a single row of the design matrix '**ϕ**'. This function is called by other functions (ex: '**phi_id**' or '**phi_val**') in order to build the rows of the design matrices.

The other function, '**regline**', is used to construct the regressor line for each data point based on past input and output values.

The other functions (**'phi_function_id'**, **'phi_function_id_sim'**, **'phi_function_val'**, **'phi_function_val_sim'**) construct the design matrix **'ϕ'** for identification, identification simulation, validation, and validation simulation, respectively. These matrices are essential for parameter estimation in the ARX model.


Input Identification Data
Input Data
Output Identification Data
Output Data
Input Validation Data
Input Data
Output Validation Data
Output Data



The NARX model structure is implemented through the construction of a design matrix ϕ.

Upon achieving the coefficients θ , the generation of the predictive model for both the identification and validation datasets results through the matrix multiplication of ϕ and θ.

The model parameters are estimated through linear regression, and the performance is evaluated using MSE values for prediction and simulation errors.

3. Tuning parameters

One can observe that by taking ***na*** = ***nb*** , the structure of the model is simplified by reducing the number of independent parametric combinations.

By increasing the ***na*** = ***nb*** , the model will capture more history and it could potentially make the model much more detailed and by resulting in a more wide-ranging representation of the system dynamics.

By decreasing the ***na*** = ***nb*** , the model will capture simplifies the model interpretation and could possibly result in a straightforward interpretation.

By increasing ***nk*** , a time delay is added and also applied to both the past input and output terms.

By decreasing ***nk*** , the time delay doesn't affect the system in such a manner and it leads to a faster response in the model.

Increasing ***m*** allows the model to capture more complex patterns while decreasing ***m*** simplifies the model, making it more linear.

4. Plots

To provide the precision of our model on the validation data, we calculate the Mean Squared Error (MSE). This way we can measure how close our model's predictions are to the actual values from the validation set.

By calculating the MSE for different values with the below formula, the understanding of how well our model is doing under certain conditions is easier.

$$MSE = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$, where $\hat{y}_i$ is the approximated value, and y_i is the given value.

The most straightforward way to visualize the accuracy of the model is by using MATLAB in order to calculate and display the Mean Square Errors.

```
Warning: Updating objects saved with previous MATLAB version...
Resave your MAT files to improve loading speed.
The mse for identification prediction is: 0.033495 for na=1, nb=1, nk=1, m=3.
The mse for identification simulation is: 0.105467 for na=1, nb=1, nk=1, m=3.
The mse for validation prediction is: 0.000046 for na=1, nb=1, nk=1, m=3.
The mse for validation simulation is: 0.000134 for na=1, nb=1, nk=1, m=3.

Warning: Updating objects saved with previous MATLAB version...
Resave your MAT files to improve loading speed.
The mse for identification prediction is: 0.233169 for na=3, nb=3, nk=1, m=1.
The mse for identification simulation is: 0.282454 for na=3, nb=3, nk=1, m=1.
The mse for validation prediction is: 0.000271 for na=3, nb=3, nk=1, m=1.
The mse for validation simulation is: 0.000922 for na=3, nb=3, nk=1, m=1.

Warning: Updating objects saved with previous MATLAB version...
Resave your MAT files to improve loading speed.
The mse for identification prediction is: 0.200920 for na=4, nb=4, nk=1, m=1.
The mse for identification simulation is: 0.247679 for na=4, nb=4, nk=1, m=1.
The mse for validation prediction is: 0.000178 for na=4, nb=4, nk=1, m=1.
The mse for validation simulation is: 0.001048 for na=4, nb=4, nk=1, m=1.
```

<- best case

mse	66.9903
mse2	210.9343
mse3	0.0919
mse4	0.2680
mse_id	0.0335
mse_id_sim	0.1055
mse_val	4.5932e-05
mse_val_sim	1.3402e-04

<- random case

mse	466.3384
mse2	564.9077
mse3	0.5412
mse4	1.8442
mse_id	0.2332
mse_id_sim	0.2825
mse_val	2.7059e-04
mse_val_sim	9.2211e-04

<- random case

mse	401.8392
mse2	495.3584
mse3	0.3564
mse4	2.0954
mse_id	0.2009
mse_id_sim	0.2477
mse_val	1.7821e-04
mse_val_sim	0.0010

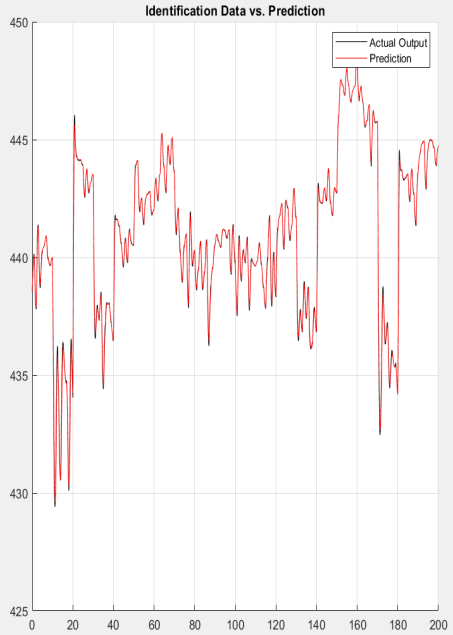
It can be observed that the identification errors slightly differs from the validation errors.

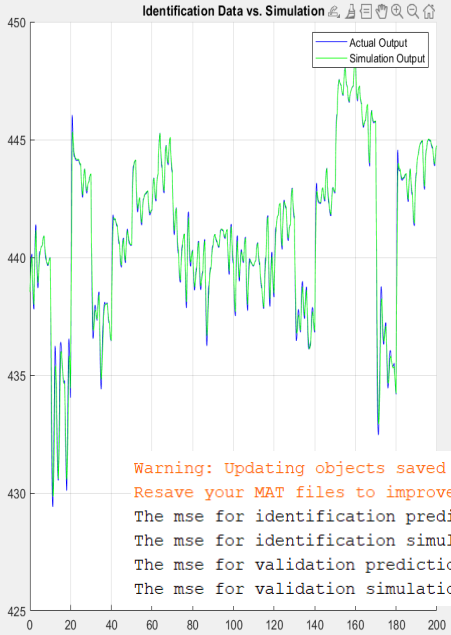
In our provided dataset, the optimal model for the validation data corresponds to ***na*** = ***nb*** = ***1***. ***nk*** will be taken as **1** by default and the best fitting degree will be ***m*** = ***3***.

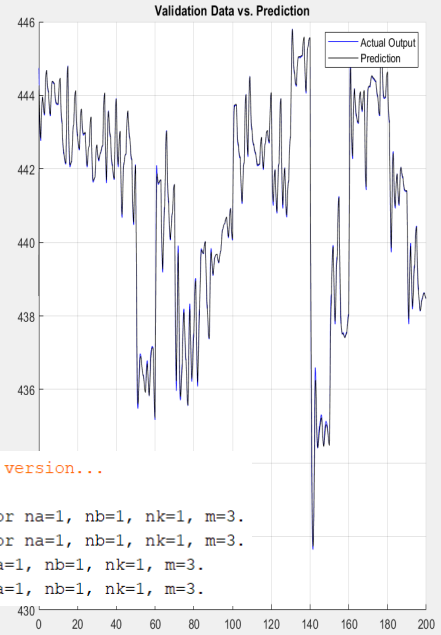
Additionally, it's noticeable that as ***m*** increases, the Mean Squared Error (MSE) for the identification and validation data decreases, indicating a better fit.

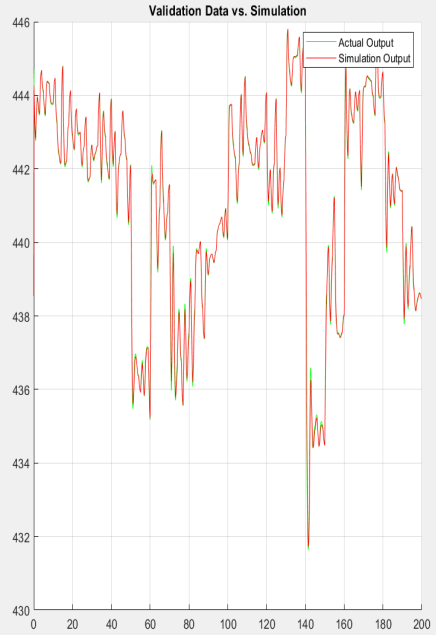
mse	66.9903
mse2	210.9343
mse3	31.4276
mse4	95.3475
mse_id	0.0335
mse_id_sim	0.1055
mse_val	0.0157
mse_val_sim	0.0477

Best case

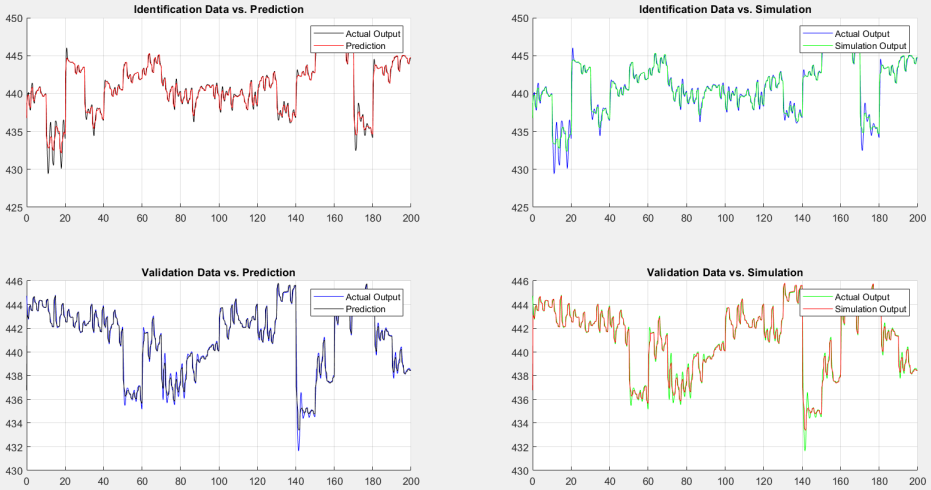




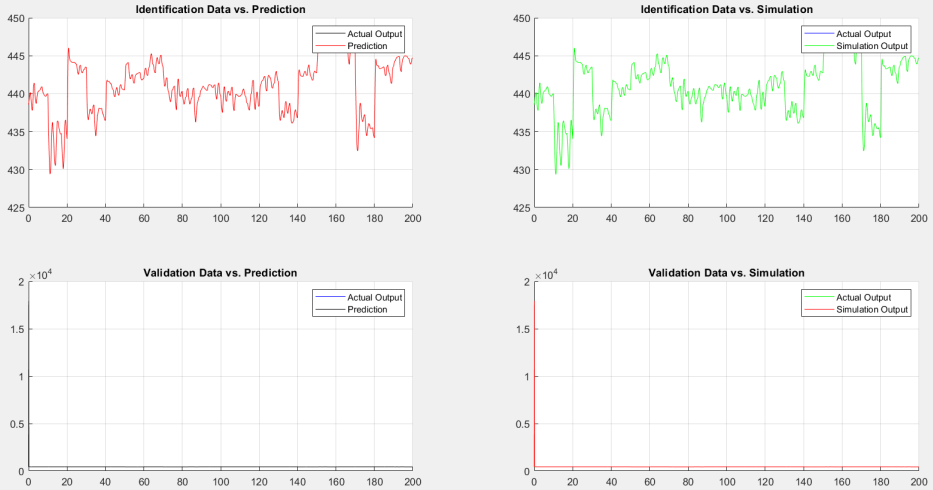





```
Warning: Updating objects saved with previous MATLAB version...
Resave your MAT files to improve loading speed.
The mse for identification prediction is: 0.033495 for na=1, nb=1, nk=1, m=3.
The mse for identification simulation is: 0.105467 for na=1, nb=1, nk=1, m=3.
The mse for validation prediction is: 0.015714 for na=1, nb=1, nk=1, m=3.
The mse for validation simulation is: 0.047674 for na=1, nb=1, nk=1, m=3.
```



Underfitting



Overfitting

```
Warning: Updating objects saved with previous MATLAB version...
Resave your MAT files to improve loading speed.
The mse for identification prediction is: 0.327566 for na=2, nb=2, nk=1, m=1.
The mse for identification simulation is: 0.397743 for na=2, nb=2, nk=1, m=1.
The mse for validation prediction is: 0.164251 for na=2, nb=2, nk=1, m=1.
The mse for validation simulation is: 0.194631 for na=2, nb=2, nk=1, m=1.
```

```
Resave your MAT files to improve loading speed.
Warning: Rank deficient, rank = 32, tol =  1.703130e-03.
The mse for identification prediction is: 0.000008 for na=2, nb=2, nk=1, m=3.
The mse for identification simulation is: 0.000038 for na=2, nb=2, nk=1, m=3.
The mse for validation prediction is: 0.000006 for na=2, nb=2, nk=1, m=3.
The mse for validation simulation is: 0.000027 for na=2, nb=2, nk=1, m=3.
```

5. Conclusions

- The provided MATLAB code offers a systematic approach to identify and approximate a dynamic system using a linear-in-parameters model with a polynomial structure.

- The choice of parameters ***na***, ***nb***, ***nk***, and ***m*** allows for flexibility in tailoring the model to specific system characteristics.

- The effectiveness of the model is evaluated through MSE calculations, providing insights into the accuracy of both prediction and simulation results.

- This approach is valuable in various engineering applications where accurate system modeling is needed for successful control and optimization.

- For example, aircraft control systems must ensure stability, performance, and safety during various flight conditions. Accurate modeling is essential for designing control systems that respond predictably to pilot commands and external factors like turbulence.

6. **Appendix**

Loading, separating and visualizing and setting up parameters for the model

```
close all;
clear all;
clc;

load('iddata-12.mat');

Ts=id.Ts;

% Separate input and output data for identification
uid=id.InputData;
yid=id.OutputData;

% Separate input and output data for validation
uval=val.InputData;
yval=val.OutputData;

% Plotting the identification and validation data
figure;
subplot(411)
plot(uid,'k');
title('Input Identification Data');grid;shg;
legend('Input Data');
subplot(412)
plot(yid,'g');
title('Output Identification Data');grid;shg;
legend('Output Data');
subplot(413)
plot(uval,'k');
title('Input Validation Data');grid;shg;
legend('Input Data');
subplot(414)
plot(yval,'g');
title('Output Validation Data');grid;shg;
legend('Output Data');
```

```
% Model parameters
na=1; % Ideal cases for na=nb=1 and
m=3 and also for na=nb=3 and m=1.
nb=1;
nk=1; % Delay in the system

m=3; % Degree


% Time vectors for identification and
validation
timp_de_id=0:Ts:199.9;
timp_de_val=0:Ts:199.9;
```

'phi_function_id' for creating the identification phi matrix (analog, the script is modified in order to create the other phi matrices)

```
% Function to calculate the phi matrix for identification

function phi_id=phi_function_id(na,nb,nk,degr_m,uid,yid)

phi_id=[];

for i=1:length(yid)

vector=regline(na,nb,nk,i,uid,yid);
phi_lines=[];

for j=1:length(vector)
phi_lines=phi_line_function(phi_lines,vector,0,degr_m,1,j);
end

phi_id=[phi_id; phi_lines];
end
end
```

```
% Function to calculate the phi matrix for identification

function phi_id_sim=phi_function_id_sim(na,nb,nk,degr_m,uid,y_hatz_id)

phi_id_sim=[];

for i=1:length(y_hatz_id)

vector=regline(na,nb,nk,i,uid,y_hatz_id);
phi_lines=[];

for j=1:length(vector)
phi_lines=phi_line_function(phi_lines,vector,0,degr_m,1,j);
end

phi_id_sim=[phi_id_sim;phi_lines];
end
end
```

```
% Function to calculate the phi matrix for identification

function phi_val=phi_function_val(na,nb,nk,degr_m,uval,yval)

phi_val=[];

for i=1:length(yval)

vector=regline(na,nb,nk,i,uval,yval);
phi_lines=[];

for j=1:length(vector)
phi_lines=phi_line_function(phi_lines,vector,0,degr_m,1,j);
end

phi_val=[phi_val;phi_lines];
end
end
```

```
% Function to calculate the phi matrix for validation simulation

function phi_val_sim=phi_function_val_sim(na,nb,nk,degr_m,uval,y_hatz_val)

phi_val_sim=[];

for i=1:length(y_hatz_val)

vector=regline(na,nb,nk,i,uval,y_hatz_val);
phi_lines=[];

for j=1:length(vector)
phi_lines=phi_line_function(phi_lines,vector,0,degr_m,1,j);
end

phi_val_sim=[phi_val_sim;phi_lines];
end
end
```

Implementation of phi matrices for identification and validation, plotting and MSE evaluation

```
% Building the phi matrix for identification
phi_id=phi_function_id(na,nb,nk,m,uid,yid);

% Estimate model parameters
theta=phi_id\yid;
y_hatz_id=phi_id*theta;

% Plotting identification data against prediction and simulation
figure;

subplot(221);
hold on;
grid;shg;
plot(timp_de_id,yid,'k');
plot(timp_de_id,y_hatz_id,'r');
hold off;
title('Identification Data vs. Prediction');
legend('Actual Output','Prediction');

% Building the phi matrix for identification simulation
phi_id_sim=phi_function_id_sim(na,nb,nk,m,uid,y_hatz_id);
y_hatz_id_sim=phi_id_sim*theta;

subplot(222);

hold on;
grid;shg;
plot(timp_de_id,yid,'b');
plot(timp_de_id,y_hatz_id_sim,'g');
hold off;
title('Identification Data vs. Simulation');
legend('Actual Output','Simulation Output');
```

```
% Building the phi matrix for validation
phi_val=phi_function_val(na,nb,nk,m,uval,yval);
y_hatz_val=phi_val*theta;

% Plotting validation data for prediction

subplot(223);
hold on;
grid;shg;
plot(timp_de_val,yval,'b');
plot(timp_de_val,y_hatz_val,'k');
hold off;
title('Validation Data vs. Prediction');
legend('Actual Output','Prediction');

% Calculate mean squared errors for validation prediction and simulation
mse3=0;
mse4=0;

phi_val_sim=phi_function_val_sim(na,nb,nk,m,uval,y_hatz_val);
y_hatz_val_sim=phi_val_sim*theta;

for i=5:length(yval)
mse3= mse3+((y_hatz_val(i)-yval(i))^2);
mse4=mse4+((y_hatz_val_sim(i)-yval(i))^2);
end

mse_val=(1/length(yval)*mse3);
mse_val_sim=(1/length(yval)*mse4);

% Building the phi matrix for validation simulation
phi_val_sim=phi_function_val_sim(na,nb,nk,m,uval,yval);
y_hatz_val_sim=phi_val_sim*theta;
```

```
% Plot validation data for simulation
subplot(224);
hold on;
grid;shg;
plot(timp_de_val,yval,'g');
plot(timp_de_val,y_hatz_val_sim,'r');
hold off;

title('Validation Data vs. Simulation');
legend('Actual Output','Simulation Output');

% Calculate mean squared errors for identification
prediction and simulation
mse=0;
mse2=0;

for i=1:length(yid)
mse=mse+((y_hatz_id(i)-yid(i))^2);
mse2=mse2+((y_hatz_id_sim(i)-yid(i))^2);
end

mse_id=(1/length(yid)*mse);
mse_id_sim=(1/length(yid)*mse2);

% Displaying mean squared errors
fprintf('The mse for identification prediction is: %f for
na=%d, nb=%d, nk=%d, m=%d. \n',mse_id,na,nb,nk,m);
fprintf('The mse for identification simulation is: %f for
na=%d, nb=%d, nk=%d, m=%d.
\n',mse_id_sim,na,nb,nk,m);
fprintf('The mse for validation prediction is: %f for na=%d,
nb=%d, nk=%d, m=%d. \n',mse_val,na,nb,nk,m);
fprintf('The mse for validation simulation is: %f for na=%d,
nb=%d, nk=%d, m=%d. \n',mse_val_sim,na,nb,nk,m);
```

Function to calculate a phi matrix line

```
function philine=phi_line_function(philine,vector,deg,degr_m,value,position)

value=value*vector(position);
deg=deg+1;

if deg~=degr_m
        for i=position:length(vector)
        philine=phi_line_function(philine,vector,deg,degr_m,value,i);
        end
else
        philine=[philine value];
        end
end
```

Function to construct the regressor line

```
function phi_val=phi_function_val(na,nb,nk,degr_m,uval,yval)

phi_val=[];

for i=1:length(yval)

vector=regline(na,nb,nk,i,uval,yval);
phi_lines=[];

for j=1:length(vector)
        phi_lines=phi_line_function(phi_lines,vector,0,degr_m,1,j);
end

phi_val=[phi_val;phi_lines];
        end
end
```