System Identification Project Part II

GROUP NR 6

STUDENTS: MATEI CRISTINA

PERPELICI ALEXANDRA

ROKALY KRISZTA

Introduction/Problem statement

- We were given a dataset which contained one input and one output for a measured unknown dynamic system
- Our goal was to develop a black-box model for this system based on a polynomial, nonlinear ARX model
- We had to combine our knowledge from linear regression, together with the ARX identification technique in order to find the model for our system
- Lastly, we had to tune the model orders *na* and *nb* and also the polynomial degree *m* so we could find the best approximation

Methodology

Our main 2 goals were:

1. Finding the prediction output Y_hat based on the previous inputs and outputs.

2. Finding the simulation output Y_sim based on the previously simulated outputs, together with the previous inputs.

We did this for both identification and validation sets.

For calculating the regressors matrix PHI, we used a matrix of previous inputs and outputs, together with a matrix that contained all the combinations for the corresponding powers.

Using linear regression, we computed the coefficients vector THETA and after that we found our prediction output Y_hat.

For finding the simulation output Y_sim, we did the corresponding computations between our previously simulated outputs or given inputs, our matrix of powers and our THETA.

In the end, we calculated the errors with different values for the model orders and for the polynomial degree.

Algorithm

Computing the matrix of powers:

- one line represents the powers of one term in the polynomial
- we generated our matrix through a successive sum of vectors, adding 1 to a position until we fulfilled our condition
- our condition was that the maximum degree of a term must not be greater than our given polynomial degree : *m*
- after our condition was fulfilled, we moved on to the next position in the vector and repeated the procedure, while continuing to add the generated vectors to the ones that are already in the matrix to check if a new combination of powers can be added to the matrix
- through this algorithm, we generated our matrix of powers, having all the possible combinations that would fulfill our condition

Algorithm

Performing the ARX for the prediction output:

- to find our prediction output, we had to create a matrix that would contain the data used for generating the regressors matrix *PHI*. We called this matrix *data_matrix*
- in *data_matrix*, we performed an ARX model by taking the corresponding outputs and inputs according to our given model orders *na* and *nb*
- for the first *na* columns we had our previous outputs and for the last *nb* columns we had our previous inputs
- the elements with an index <= 0 are equal with 0

Algorithm

Calculating the prediction output:

- the next step was to compute our regressors matrix for the identification set, *PHI_id* term by term
- each line from this matrix would contain the terms of a polynomial equation which generates one output from the given data set
- for calculating the *x'th* term, we took the data from one line from the previously determined matrix: *data_matrix,* raised each value to the corresponding powers from the *x'th* line in the matrix of powers and create a product
- after that, we used the linear regression technique to find the parameters vector: *THETA= PHI_id\id.y,* where *id.y* is the output for the identification set
- lastly, we generated our prediction output for the validation set using the regressors matrix calculated for the validation set *PHI_val* and the parameters vector *THETA : Y_hat_val = PHI_val*THETA.*
- we did the same for identification: *Y_hat_id = PHI_id*THETA.*

Algorithm

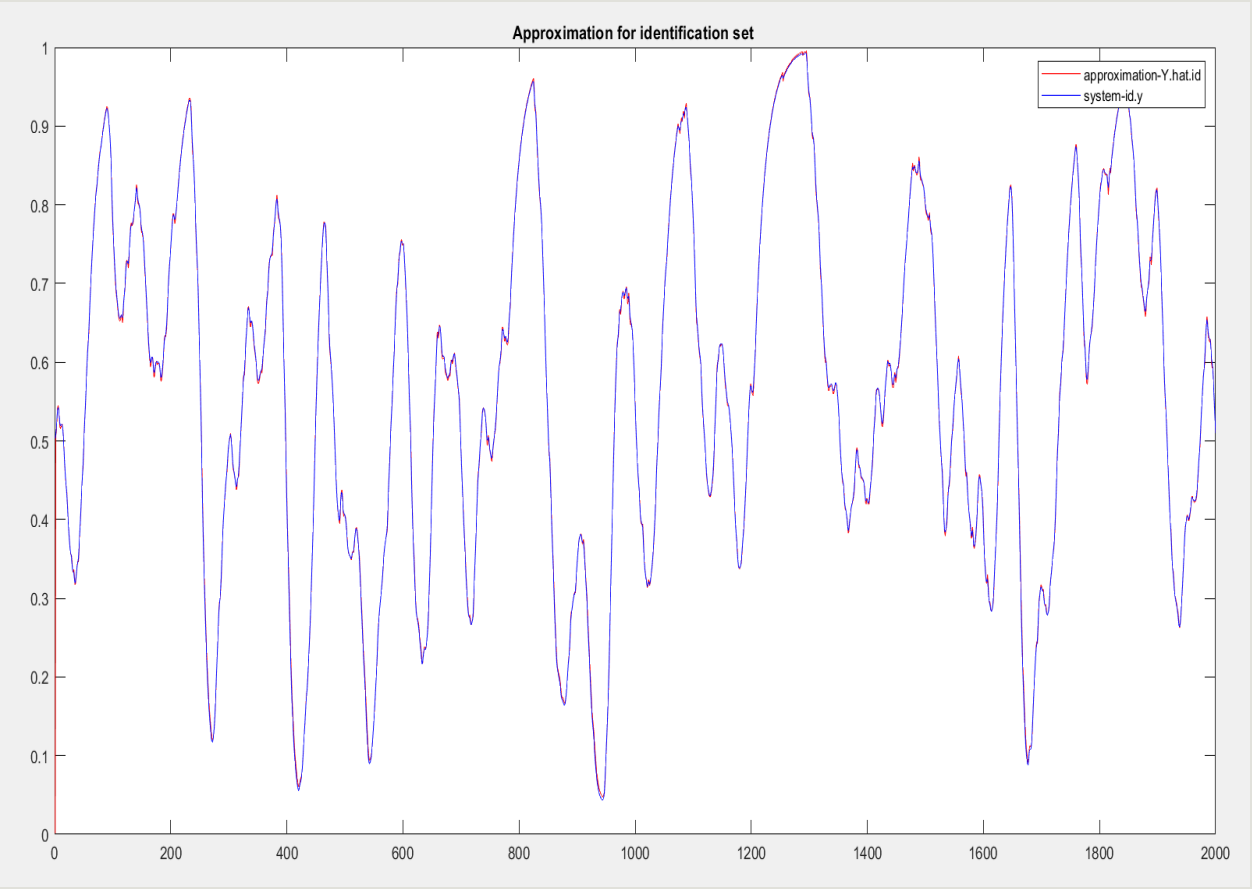
Calculating the simulation output:

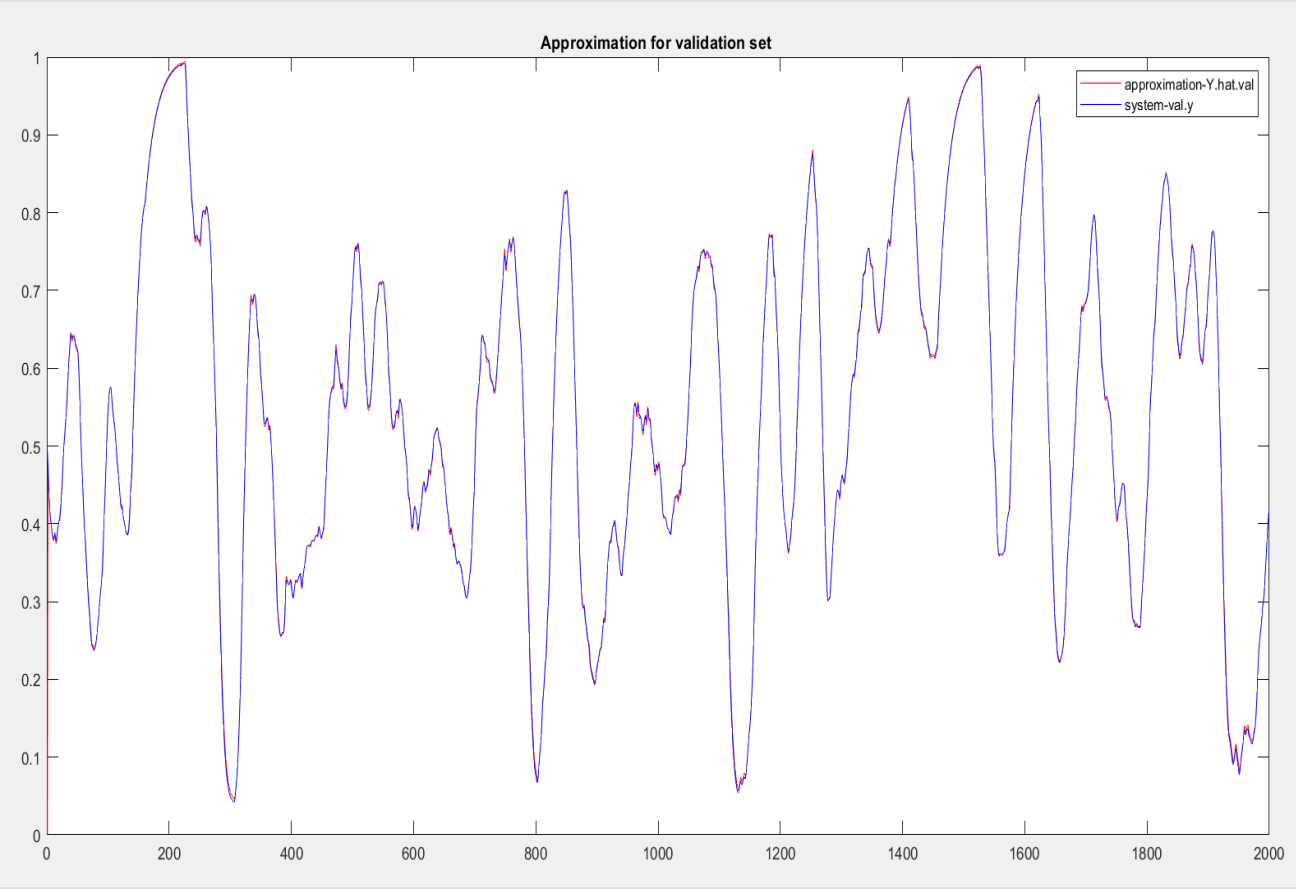
- for the first *na* columns we used the previously simulated outputs and for the last *nb* columns we used the previous inputs
- we raised this values to the corresponding powers and we created a product. This product would represent one term from the polynomial
- we then multiplied this term with it's associated value from the parameters vector THETA
- finally, we created a sum between all this terms to find our simulation for one value from the given data set
- we repeated this for all values for bot identification and validation sets

Tuning and results

Through several tries, we found that our best fitting solution works for the model orders *na* and *nb* = 1 and the polynomial degree *m = 2.*

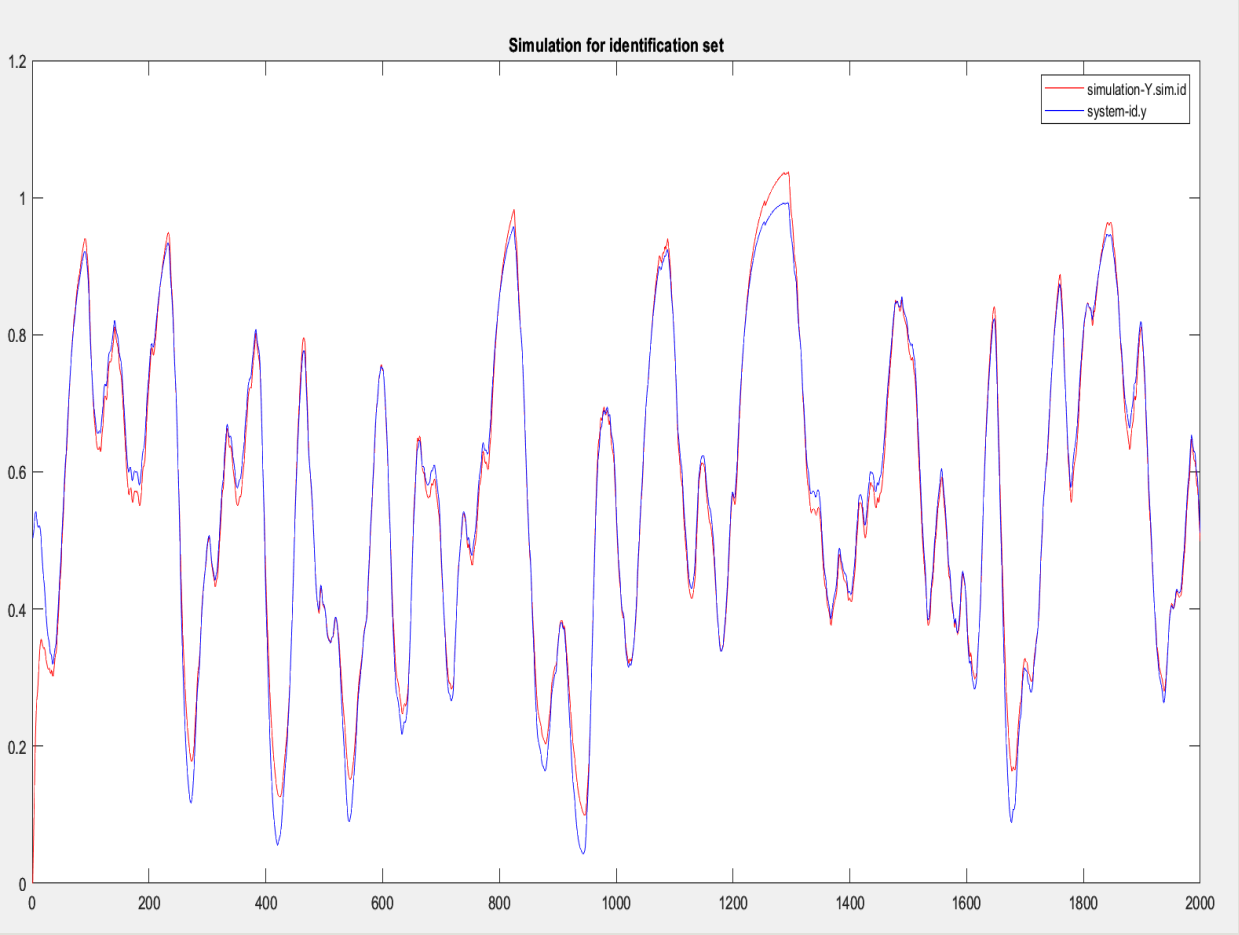
For prediction:

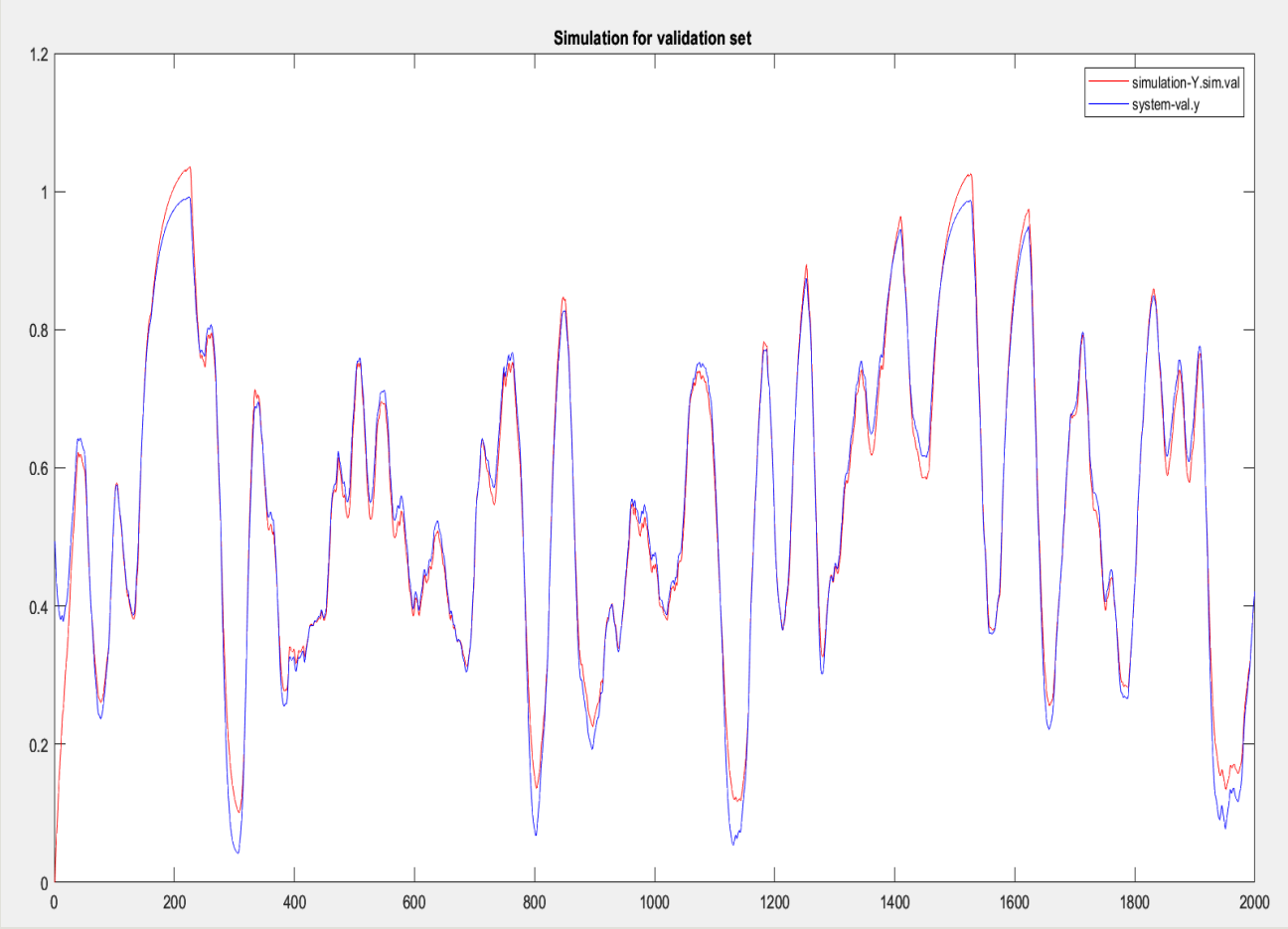




Tuning and results

For simulation:





Tuning and results

Errors for prediction: 1) on identification set

2) on validation set

polynomial degree *m* – on lines, model orders *na* and *nb* on columns

Variables - V_MSE_approx_id

V_MSE_approx_id

3x3 double

	1	2	3	4
1	1.5179e-04	1.5003e-04	1.4803e-04	
2	1.3545e-04	6.6291e-05	2.9464e-05	
3	1.1548e-04	4.3591e-06	2.8013e-06	
4				

Variables - V_MSE_approx_val

V_MSE_approx_val

3x3 double

	1	2	3	4
1	1.4686e-04	1.4515e-04	1.4338e-04	
2	1.3070e-04	0.0018	0.0019	
3	1.1454e-04	0.5116	2.6324e+03	
4				

Tuning and results

Errors for simulation: 1) on identification set

2) on validation set

polynomial degree *m* – on lines, model orders *na* and *nb* on columns

Variables - V_MSE_sim_id

V_MSE_sim_id

3x3 double

	1	2	3	4
1	0.0018	0.0020	0.0022	
2	0.0012	NaN	NaN	
3	0.0017	NaN	NaN	
4				

Variables - V_MSE_sim_val

V_MSE_sim_val

3x3 double

	1	2	3	4
1	0.0019	0.0020	0.0021	
2	0.0012	NaN	NaN	
3	0.0022	NaN	NaN	
4				

Conclusions

To sum up, we combined our knowledge from linear regression together with ARX identification to determine a polynomial model for our dynamic system. We used this model in order to determine the one step ahead prediction output and also the simulation output for both our identification and validation sets.

We were able to find a good approximation for our model considering the model orders na and $nb = 1$ and the polynomial degree $m = 2$ by calculating the errors for all computed outputs. These parameters were the best for the simulated output and they also worked well for the approximated output.

Thank you for your attention!

Any questions?

Appendix

```
%%
clear all
load('iddata-06.mat');
u_id=id.InputData;
y_id=id.OutputData;
t=0:2:3998;
% plot(t,u_id);figure;plot(t,y_id)
% figure
% plot(t,val.y);figure;plot(t,val.u );

%%
 for na = 1:1
     nb = na;
  for m = 1:2
l = 1;
c = 1;
clear combinations
combinations(l,:) = zeros(1,na+nb);
v = zeros(1,na+nb);
len = height(combinations);
```

```
while c<=(na+nb)
   for nr = 1:m
      v = zeros(1,na+nb);
      v(c) = nr;
      for k = 1:len
         Vprot = combinations(k,:)+v;
         sum = 0;
         for number = 1:(na+nb)
            sum = sum+Vprot(number);
         end
         flag = 0;
         for contr = 1:height(combinations)
            if combinations(contr,:) == Vprot
               flag = 1;
            end
         end
         if (sum <= m) && (flag == 0)
            l = l+1;
            combinations(l,:) = Vprot;
         end
         Vprot = 0;
      end
      len = height(combinations);
   end
 c = c+1;
end
```

```
data_matrix_id = zeros(length(id.u),na+nb);
% data matrix on id
for k = 1:length(id.u)
   for j = 1:na
         if k-j<=0
            data_matrix_id(k,j) = 0;
         else
            data_matrix_id(k,j) = id.y(k-j);
         end
   end

   for j = 1:nb
         if k-j <=0
             data_matrix_id(k,na+j) = 0;
         else
             data_matrix_id(k,na+j) = id.u(k-j);
         end
   end
end
% data matrix pe val
data_matrix_val = zeros(length(val.u), na+nb);
for k = 1:length(val.u)
   for j = 1:na
         if k-j<=0
            data_matrix_val(k,j) = 0;
         else
            data_matrix_val(k,j) = val.y(k-j);
         end
   end

   for j = 1:nb
         if k-j <=0
             data_matrix_val(k,na+j) = 0;
         else
             data_matrix_val(k,na+j)= val.u(k-j);
         end
   end
end
```

```
lines_c = length(combinations(:,end));
% PHI, Y_hat and theta on id
PHI_id = ones(length(id.y),lines_c);

for i = 1:length(id.y) % nr of lines - one line corresponds to one polynomial equation
  for j = 1:lines_c   % for the number of columns of PHI (which is = with the number of lines of combinations) - one column corresponds to one product of the
polinomial equation
    for k = 1:na+nb  % for taking the corresponding data from the data_matrix and the corresponding powers from the combinations matrix -each k
corresponds to one term of the j'th product
        PHI_id(i,j) = PHI_id(i,j)*data_matrix_id(i,k)^combinations(j,k);
    end
  end
end

THETA  = PHI_id\id.y;

Y_hat_id = PHI_id*THETA; % y prediction for identification set

% PHI, Y_HAT ON VAL
PHI_val = ones(length(val.y),lines_c);

for i = 1:length(val.y) % nr of lines
  for j = 1:lines_c   % for the number of columns of PHI which is the number of lines of combinations
    for k = 1:na+nb % for taking the corresponding data from the data_matrix and the corresponding powers from the combinations matrix
        PHI_val(i,j) = PHI_val(i,j)*data_matrix_val(i,k)^combinations(j,k);
    end
  end
end


Y_hat_val = PHI_val*THETA; % y prediction for validation set
```

```
% Ysim id
Y_sim_id = zeros(length(id.y),1); % y simulation for identification

for k = 1:length(id.y)
for p = 1:lines_c
   prod = 1;
    for i = 1:na
        if(k-i<=0)
           prod = 0;
        else
           prod = prod * (Y_sim_id(k-i)^combinations(p,i));
        end
    end

    for i = 1:nb
         if(k-i<=0)
           prod = 0;
         else
          prod = prod * (id.u(k-i)^combinations(p,na+i));
         end
    end
 Y_sim_id(k) = Y_sim_id(k) + prod * THETA(p);
end
end
```

```
% Ysim val
Y_sim_val = zeros(length(val.y),1); % y simulation for identification

for k = 1:length(val.y)
for p = 1:lines_c
  prod = 1;
    for i = 1:na
        if(k-i<=0)
           prod = 0;
        else
           prod = prod * (Y_sim_val(k-i)^combinations(p,i));
        end
    end

    for i = 1:nb
         if(k-i<=0)
           prod = 0;
         else
          prod = prod * (val.u(k-i)^combinations(p,na+i));
         end
    end
 Y_sim_val(k) = Y_sim_val(k)+ prod * THETA(p);
end
end
```

```
% errors
% For identification
s1 = 0;
for i = 1:length(id.y)
    s1 = s1+(id.y(i)-Y_hat_id(i)).^2;
end
s1 = 1/length(id.y)*s1;
V_MSE_approx_id(m,na)=s1;

 s2 = 0;
for i = 1:length(id.y)
    s2 = s2+(id.y(i)-Y_sim_id(i)).^2;
end
s2 = 1/length(id.y)*s2;
V_MSE_sim_id(m,na)=s2;

% For validation
s3 = 0;
for i = 1:length(val.y)
    s3 = s3+(val.y(i)-Y_hat_val(i)).^2;
end
s3 = 1/length(val.y)*s3;
V_MSE_approx_val(m,na)=s3;


s4 = 0;
for i = 1:length(val.y)
    s4 = s4+(val.y(i)-Y_sim_val(i)).^2;
end
s4 = 1/length(val.y)*s4;
V_MSE_sim_val(m,na)=s4;


  end
 end
```

```
%% plot the prediction and real data for identification
figure
plot(Y_hat_id,'r');
hold on
plot(id.y,'b');
legend('approximation-Y.hat.id','system-id.y')
title ('Approximation for identification set ')
```

```
%% plot the prediction and real data for validation
figure
plot(Y_hat_val,'r');
hold on
plot(val.y,'b');
legend('approximation-Y.hat.val','system-val.y');
title ('Approximation for validation set ');

%% plot the simulation and real data for identification
figure
plot(Y_sim_id,'r');hold;
plot(id.y,'b');
legend('simulation-Y.sim.id','system-id.y');
title ('Simulation for identification set');
%% plot the simulation and real data for validation
figure
plot(Y_sim_val,'r');hold;
plot(val.y,'b');
legend('simulation-Y.sim.val','system-val.y')
title ('Simulation for validation set ')
```