Project I System identification

Group:30331

Students:
Durus Mihai-Valer
Nicoara Toma-Stefan
Dioane Radu-Alexandru

Indexes: 14/15

Function approximator using polynomial regression

1.1 Basic concepts and project topic presentation

Machine learning is one of the fastest-growing branches of artificial intelligence, a technology which is based on constantly improving algorithms automatically when accumulating and processing more and more data. One of the fundamental methodologies on which machine learning is based is system identification which is responsible for building a mathematical model based on the input and output sets.

The topic of our project was to build a mathematical model which determines how two input variables denoted with x1, x2 influence the output y. An example in everyday life would be: What is the relationship between how many vegetable kilograms (x1) and how many red meat kilograms(x2) eats an adult male on year and his life expectancy measured in years (y).

In our project, we use 2 main datasets: identification data and validation data. The identification data has been used to calculate the model while the validation data is used to test the obtained model.

The mathematical model obtained by studying the relationship between inputs, outputs and the past behavior of the system, will then be used to predict the system outputs with a certain degree of error which of course is desired to be minimal. There are many techniques for obtaining mathematical models of the systems. In our project, we used polynomial regression for model determination and mean squared error (MSE) for error calculation.

More specifically, we built a polynomial approximator. A polynomial approximator of variable degree which tries to find the polynomial function which best describes the system behavior. The degree of the polynomial function will vary from 1 to m (input variable), and the program will return the degree with the model which best fits the system's output. This process can be divided into 2 parts:

1.The model construction using polynomial regression on identification data, at the end we will obtain a model described by a polynomial function.

2. Mean square error calculation, in which we calculate the error between the system output and the output of every model with a degree from 1 to m. We calculate the minimum error and display the model corresponding to it. This represents the best approximation and will be the output of our programmed solution.

1.2 Introduction of mathematical concepts

Mean squared error is an error calculation method which is an average of the squared differences between the output of the system at a given time, and the output obtain with the mathematical model at that time. A simplified example of MSE in action can be observed in appendix at **Figure 1.1**. Also there are further details regarding MSE and its implementation in our project in the coming sections.

A problem which occurs often is overfitting. After calculating models and mean squared errors for different models (different m) on identification data and the minimum error is very small (close to 0) it might be a problem. In real systems the signals usually are corrupted by noise, and the fact that a very small error was obtained might be because the noise was also modeled by the algorithm. This is very problematic because it might ruin the whole model. That is exactly the reason why we have

validation data. To be sure that the model corresponds, we test it on different data sets than the ones used for obtaining it. If overfitting occurs (the noise of the identification data was also modeled) we will obtain a large mean square error on validation data. It means that the model we calculated only fits the identification data, so we need to choose another model so that the MSE on validation data set to be minimum. An example of overfitting model can be found at **Figure 1.2**.

Polynomial regression is a form of regression analysis in which the relationship between the independent variable and the depending one is given by a polynomial function of degree n. It is the mathematical algorithm based on which we compute models on identification data. Further details regarding this topic and its implementation in our project can be found in the coming sections.

2.1 Mathematical Approach

First, we started approaching the problem with the mathematical modelization of the polynomial function of the two input variables from the identification data. This function has a specific combination of powers depending on the maximum degree of the predictor variables which will construct our regressor. In the regressor, there is always a one which represents the zero power of the inputs. The highest power allowed in the regressor variables is that of the degree (leading term) and for the power combinations of both inputs, the sum of the powers should not exceed the value of the degree in order to get the desired form. The order of the terms in the function does not matter regarding our problem, because with every change in the displacement of the function will just change its parameters set. The parameters of the function will be called thetas during the modelization process.

$$f(x_1,x_2) = \Theta_1 + \Theta_2 * x_1 + \Theta_3 * x_2 + \dots + \Theta_{2m-1} * {x_1}^m + \Theta_{2m-1} * {x_2}^m + \dots + \Theta_{2m} * x_1 * x_2 + \Theta_{2m+1} * {x_1}^2 * x_2 + \cdots$$

f – modelled function

x_i – regressors

Θ_i - parameters

As we mentioned in the introductory part, our goal is to compute those theta parameters in order to best fit the given data sets. From a mathematical point of view, we started computing the modelled function for different values of the identification data and generated these results in our output column vector (corresponding to our yflat variable used in the identification process). The thetas will be stored in a column vector and the predictor variables in a row one in order to generate a single output variable at a time. Next step consists of the separation of the independent variables from our parameters to be able to separately compute the regressor matrix (corresponding to our xflat variable used in the identification process).

$$y(1,1) = [\ 1\ x_1(1)\ \ x_2(1)\ \ x_1^2(1)\ \ x_2^2(1) \dots\ ] * [\Theta_1\ \Theta_2 \dots\ ]$$

$$y(1,2) = [\ 1\ x_1(1)\ \ x_2(2)\ \ x_1^2(1)\ \ x_2^2(2) \dots\ ] * [\Theta_1\ \Theta_2 \dots\ ]$$

$$\dots$$

$$y(1,N) = [\ 1\ x_1(1)\ \ x_2(N)\ \ x_1^2(1)\ \ x_2^2(N) \dots\ ] * [\Theta_1\ \Theta_2 \dots\ ]$$

$$\dots$$

$$y(N,1) = [\ 1\ x_1(N)\ \ x_2(1)\ \ x_1^2(N)\ \ x_2^2(1) \dots\ ] * [\Theta_1\ \Theta_2 \dots\ ]$$

$$\dots$$

$$y(N,N) = [\ 1\ x_1(N)\ \ x_2(N)\ \ x_1^2(N)\ \ x_2^2(N) \dots\ ] * [\Theta_1\ \Theta_2 \dots\ ]$$

In consequence of the above mathematical approach, each output variable will be computed as a product between the thetas and the corresponding regressor row vector. To obtain the thetas from the previously mentioned relation, we have to left-multiply by the transpose of the regressor matrix followed by another left-multiplication by the inverse of the product between transpose of the matrix of the predictor variables and its original non-transposed form.

$\Theta = (\phi^T * \phi)^{-1} * \phi^T * Y$, where $\Theta = [\Theta_1\ \Theta_2\ ...\ ]$, ϕ – regressor matrix, Y – resulted ouput column vector

Regarding the modelization procedure chosen, we also need to create somehow a validation process. Our choice of going is with the mean squared error method which consists of the squared difference between the resulted outputs for the obtained parameters and the given output data set divided by the length of those sets. The minimum mean squared error obtained will be used to generate the best model fit with the least amount of overfitting (as the error and number of parameters increases so does the overfit).

2.2 Key features of our solution

In the solution we developed using Matlab, we followed the previously described method adapting it to the tools which this programme has to offer.

Our implementation starts with the extraction of the identification and validation data sets for a subsequent procession of them. After the data managing part, we programmed the constructor for the approximator function taking in regard a variable degree to find the best fit and minimum error, this variation of the order of the polynomial approximator, implemented using a simple for loop from m=1 up to m=20 (we chose only 20 because of the overfitting problem described here and in the introuductory part).

In the construction of the regressor matrix, we firstly initialized it to a column matrix with ones on every position. We observed using the mathematical model that the first input variable iterates only when the second one goes through all the input values, translated to our code in two for loops designed to iterate in this exact way using i,j as indexes for the loops, k and k1 as indexes for the predictor variables and id.dims for the dimension of the identification data.

Each row of the regressor is computed in two parts: the columns from 2 to 2*m+1 (m - order of the polynomial approximator) for the powers of x1 and x2 without the product power combinations between them and from 2*m+2 till the end of the regressor columns (the number of columns depends on the degree) for the rest of the terms (respecting the impose mathematical form). The first part is done using a for loop, an additional variable p for generating the powers of the independent variables and an if-else structure which places the first input variable on the even columns and the second one in the odd ones, p variable is incremented only in the even columns to get the same power for both predictor variables. The second part is realised using two for loops which iterate from 1 to m-1 this time because the powers increment differently and we have to take into account two variables used for the new powers: p1 and p2 taking into account that the sum of them must not exceed m. This procedure is done for both data sets: identification and validation generating the two later used column vectors: Xflat and Xflatval.

```
count=1;
for index=1:20
Xflat(1:1681,1)=ones(1681,1);% making 1st coloumn of the regressor equal to 1

p=1;%used to generate the powers of x1 and x2 (up to index-1, this is not used for the powers of x1*x2)
k=1;%used to iterrate through x1
kl=1;%used to iterrate through x2

for j=0:id.dims-1
for i=id.dims*j+1:id.dims*j+id.dims
    %the two for loops are used to iterrate throught the vectors x1 and x2 in designd way, 41 by
    %41 for x1, and 1 by 1 for x2 for every iterration of x1
 for jcol=2:2*index+1 % this loop generates the powers of x1 and x2 (from 1 to index-1) for the regressor and places them in the regressor vector Xfla
  if(mod(jcol,2)==0)
  Xflat(i,jcol)=x1(k).^p;
  p=p+1;
  else
  Xflat(i,jcol)=x2(kl).^(p-1);
  end
 end
  p=1;
  jcol=2*index+2; % we continue to generate the rest of the terms, as combinatos of x1*x2 at different powers (p1+p2<=index)
  for p1=1:index-1% p1,p2 are used for the powers of x1*x2 (p1+p2<=index)
      for p2=1:index-1
         if(p1+p2<=index)
         Xflat(i,jcol)=x1(k).^p1.*x2(kl).^p2;
         jcol=jcol+1;
         end
      end
  end
  kl=kl+1;
end
k=k+1;
kl=1;
end

% we repeat the above process for the validation data set
```

In the following steps, using two simple for loops structure we transformed the outputs from the identification and validation data to column vectors form representing the Yflat and yvalT variables. As we presented in the mathematical model, the theta column vector can be computed using the regressor matrix and output data set as a column (corresponding to Yflat). The equation described earlier can be translated into MatLab with the "left divide" operator (theta=Xflat\Yflat), which is much easier than computing it analytically. Using a similar approach we compute the new output column vectors as a product between the obtained thetas and the previously computed validation/identification column vectors (Yhatval=theta*Xflatval and Yhatid=theta*Xflat).

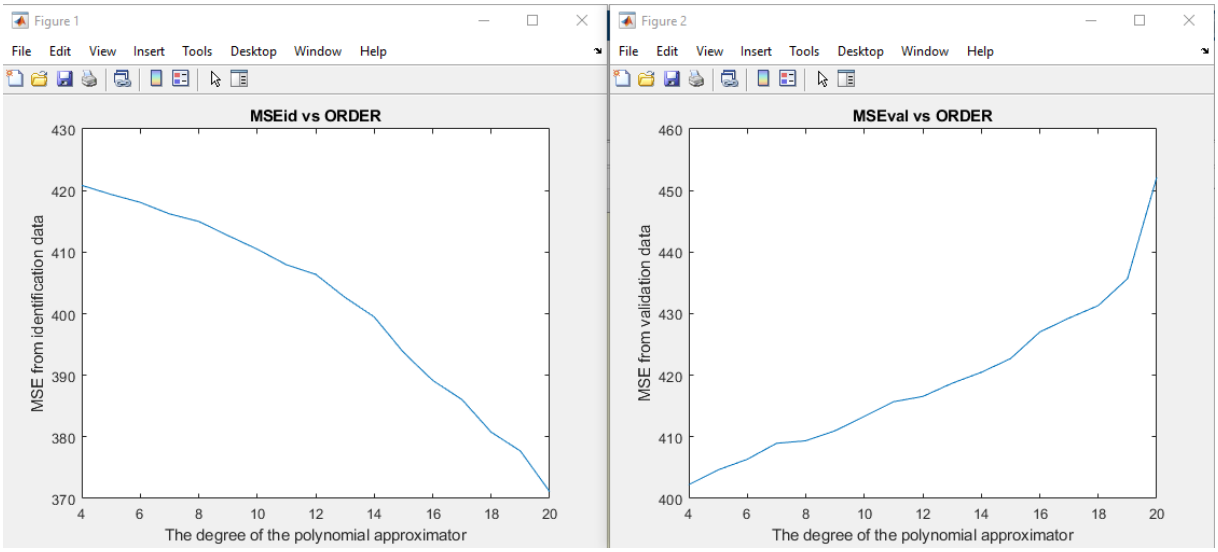
The mean squared error is calculated by a for loop structure which iterates through the output column vectors (Yflat and yvalT) as the squared difference between the resulted outputs for the obtained parameters (Yflatid and Yflatval) and the given output data set (identification/validation as column vectors) divided by the length of those sets. The resulted values are stored in two separate vectors (mseid and mseval) for each data set (identification/validation). After all the calculations the minimum of the mseval vector is computed and used to find the exact order of it, which will confer the best fit.

```
poz=1;
for i=1:id.dims% transformation of the output indentification data to a coloumn vector
    for j=1:id.dims
    Yflat(poz,1)=y(i,j);
    poz=poz+1;
    end
end
poz=1;
for i=1:length(yval)% transformation of the output validation data to a coloumn vector
    for j=1:length(yval)
        yvalT(poz,1)=yval(i,j);
        poz=poz+1;
    end
end

theta=Xflat\Yflat;% computation of the paramaters of the regressor from the identification data
Yhatval=Xflatval*theta;% computation of the new output from the validation data and the regressor parameters obtained
Yhatid=Xflat*theta;% computation of the new ouput from the identification data and the regressor parameters obtained

MSEid=0;% calculation of the MSE from the approximator and the validation data
for i=1:length(Yflat)
   MSEid=MSEid+1/length(Yflat)*((Yhatid(i,1)-Yflat(i,1)).^2);
end
mseid(count)=MSEid;% addition of every MSE to a mse vector for the purpose of determining the best order fit

MSE=0;% calculation of the MSE from the approximator and the validation data
for i=1:length(yvalT)
   MSE=MSE+1/length(yvalT)*((Yhatval(i,1)-yvalT(i,1)).^2);
end

mseval(count)=MSE;% addition of every MSE to a mse vector for the purpose of determining the best order fit
count=count+1

end

[ minim i]=min(mseval)% computation of the lowest MSE value and its coresponding order
```

3 Tuning results and representative plots

Using the degree and the mseval and mseid vectors we plotted two graphs which show the evolution of the mean squared error concerning the degree m. The minimum MSE value can be observed in the right graph (MSEval vs ORDER). We can also check for the correctitude of the solution by examining the evolution of these two graphs, the identification one should decrease a bit with respect to the degree and the validation one should slightly increase on the incrementation of the order.

.



It can be seen in the above figures, exactly from MSEval vs Order, that the minimum MSE on the validation data is obtained for m=4. Another check for the MSE calculus is the fact that square root of the MSE should be lower than ten percent of the output range of the data.

```
>> max(max(yval))-min(min(yval))

ans =

  912.1568

>> 0.1*912.1568

ans =

   91.2157
```

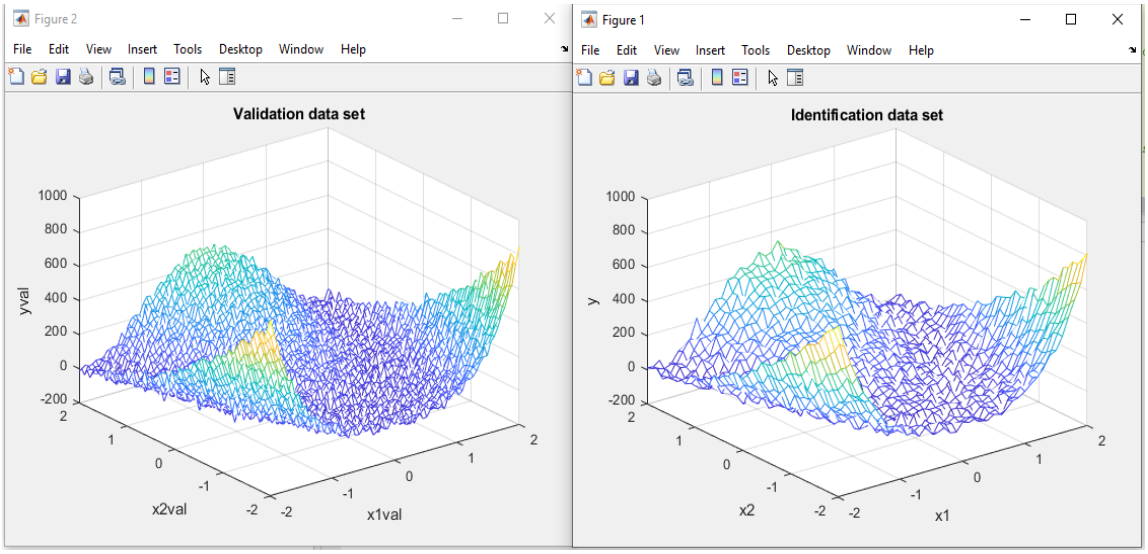
```
minim =

   402.2350

i =

     4

>> sqrt(minim)

ans =

   20.0558
```

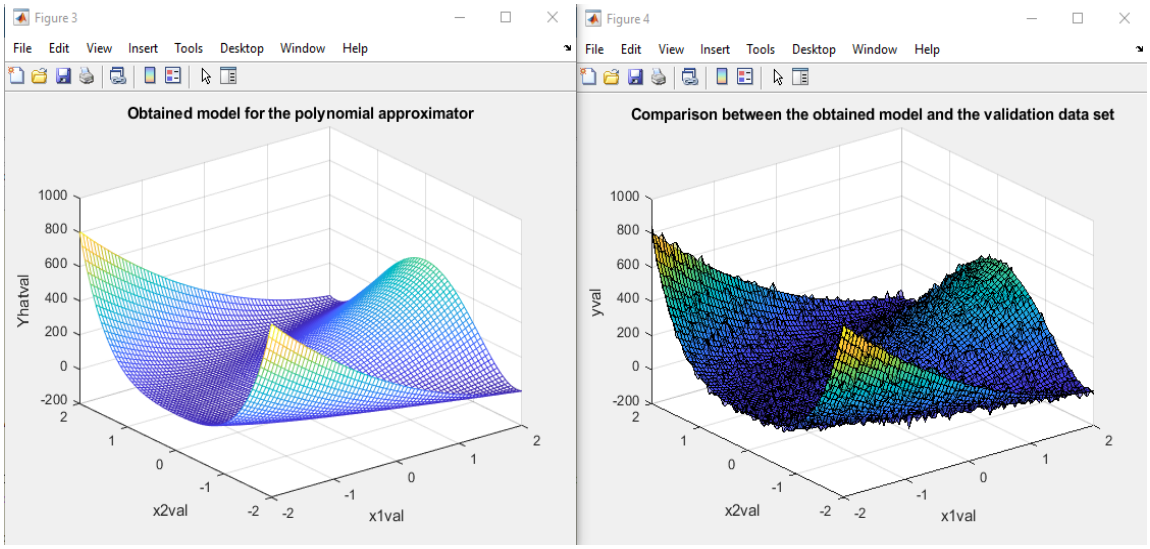
From the above pictures it can be clearly seen that the designed polynomial regressor respects the last statement we made (20.0558 is lower than 91.2157 which represents the output range).

After we found the best fitting order we repeat the same exact algorithm without the degree variation part, to recompute the all the necessary components and plot the necessary graphs which sustain our modelling and degree choice.

In the final part of our programme, we have four graphs created using matlab built-in functions: surf, mesh and meshgrid. The first one represents the identification data: x1 and x2 plotted against the

given output y, the second one is the same but on the validation data set (x1val and x2val against yval). The third one is represented by the obtained model on the validation data: x1val and x2val plotted against the output generated using the designed regressor and obtained theta (Yhatval). The last plot is the final comparison between our approximator and the given validation data set. Using this last graphical representation we can visually check the effectiveness of our developed polynomial regressor.





It can be seen that the validation data has a lot more spikes and deformation than the obtained model, which has a regulated form looking very alike the "banana function" presented in the 3rd course. This is not a very big problem because our choice of regression has its own limitations regarding how much it can reproduce the data it is applied on.

4 Conclusions

As a conclusion, the chosen mathematical approach offered us an easy and logical structure to follow from the given function of the two input variables up to the more complicated computations of the regressor matrix and the thetas used to obtain the model. We designed a fairly accurate polynomial regressor which minimizes the error and the noise with respect to the validation data. The programme is written taking into consideration the number of instructions and efficiency based on our understanding and knowledge of the MatLab programming language.

The practical implenmtation of this solution is building a model based on input and output data sets which is the core of system indenficaiton. This problem offered us the opportuinity to better understand and use the concepts of regression, function approximation, overfitting and error minimization.

5 Appendix

Figure 1.1

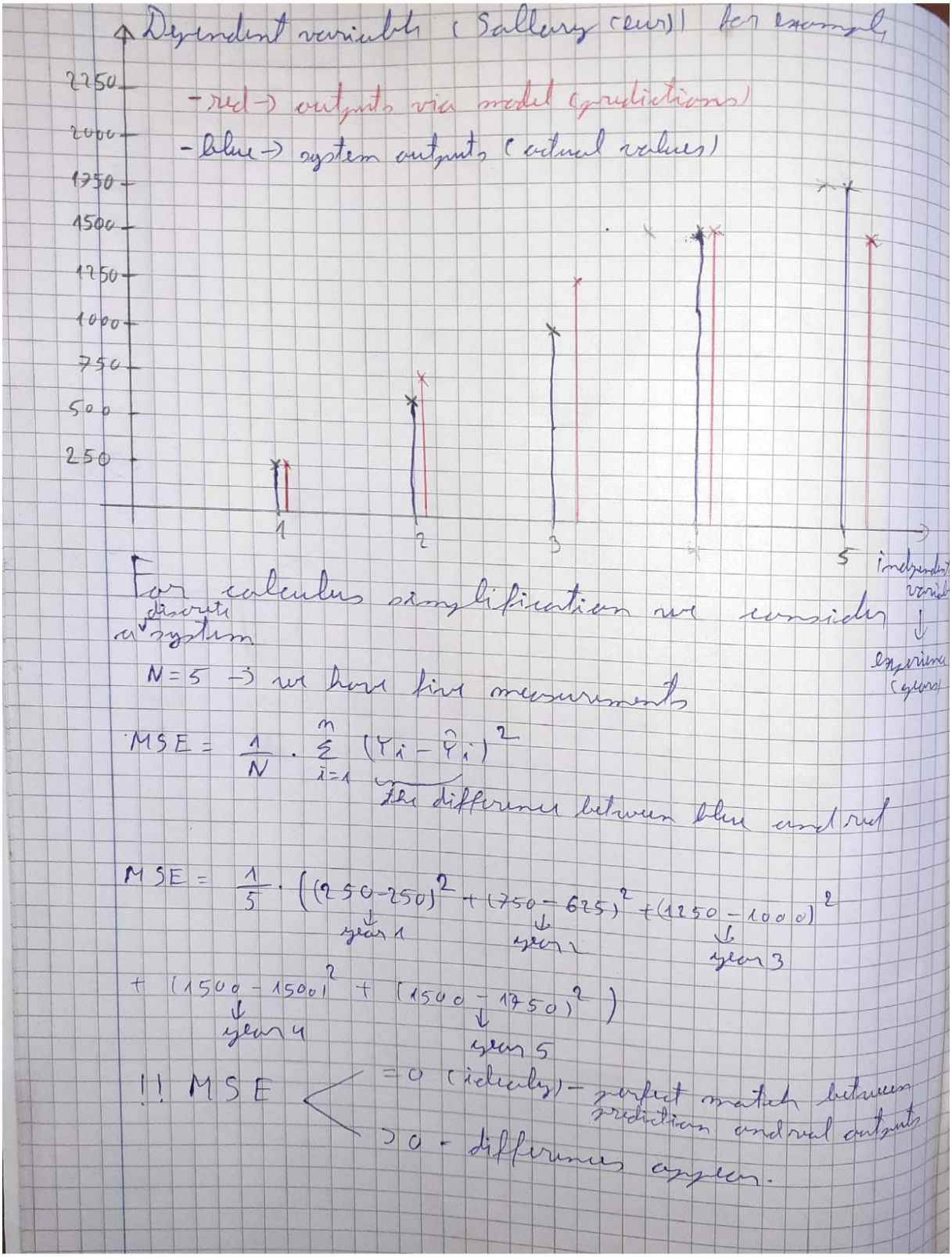
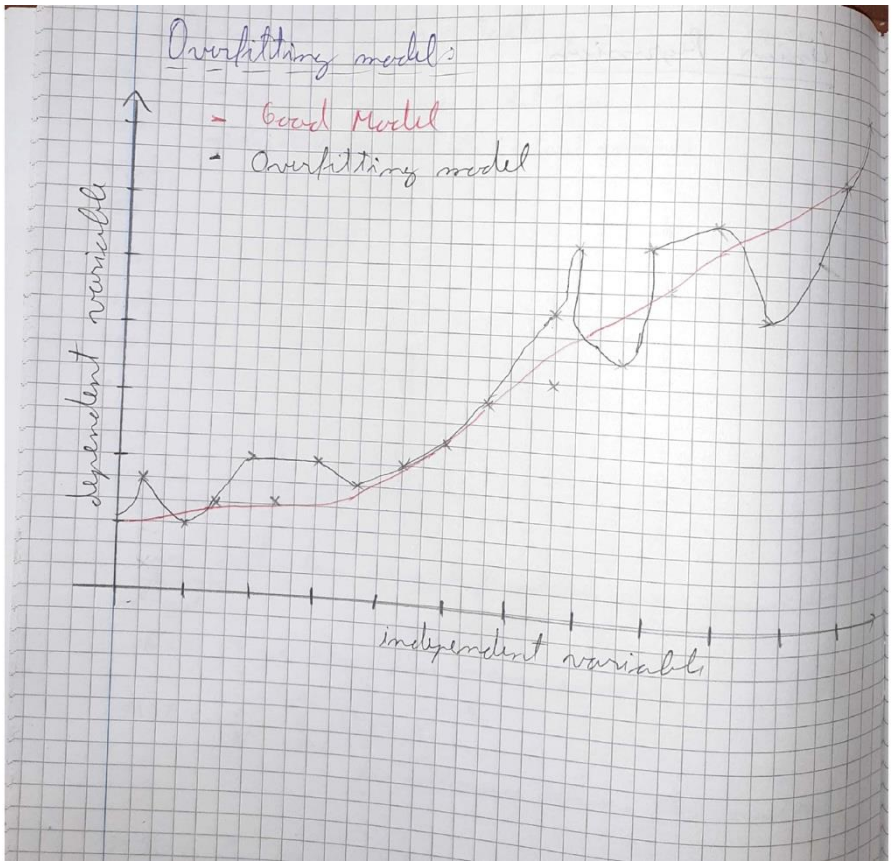


Figure 1.2



Matlab code

```
%% general solution
clear all
% Loading of the data
load('proj_fit_14','-mat');

% The extraction of the identification data
x1=id.X{1};
x2=id.X{2};
y=id.Y;

% The extraction of the the validation data
x1val=val.X{1};
x2val=val.X{2};
yval=val.Y;

% Start of the variable degree function approximator
% count is used to iterrate through mse vector
count=1;
```

```
for index=1:30

Xflat(1:1681,1)=ones(1681,1);% making 1st coloumn of the regressor equal to
1

p=1;%used to generate the powers of x1 and x2 (up to index-1, this is not
used for the powers of x1*x2)
k=1;%used to iterrate through x1
k1=1;%used to iterrate through x2

for j=0:id.dims-1
for i=id.dims*j+1:id.dims*j+id.dims
    %the two for loops are used to iterrate throught the vectors x1 and x2
in designd way, 41 by
    %41 for x1, and 1 by 1 for x2 for every iterration of x1
 for jcol=2:2*index+1 % this loop generates the powers of x1 and x2 (from 1
to index-1) for the regressor and places them in the regressor vector Xflat
  if(mod(jcol,2)==0)
  Xflat(i,jcol)=x1(k).^p;
  p=p+1;
  else
  Xflat(i,jcol)=x2(k1).^(p-1);
  end
 end
  p=1;
  jcol=2*index+2; % we continue to generate the rest of the terms, as
combinatos of x1*x2 at different powers (p1+p2<=index)
  for p1=1:index-1% p1,p2 are used for the powers of x1*x2 (p1+p2<=index)
      for p2=1:index-1
         if(p1+p2<=index)
         Xflat(i,jcol)=x1(k).^p1.*x2(k1).^p2;
         jcol=jcol+1;
         end
      end
  end
  k1=k1+1;
end
k=k+1;
k1=1;
end

% we repeat the above process for the validation data set

Xflatval(1:5041,1)=ones(5041,1);

p=1;
k=1;
k1=1;

for j=0:val.dims-1
for i=val.dims*j+1:val.dims*j+val.dims
  for jcol=2:2*index+1
  if(mod(jcol,2)==0)
  Xflatval(i,jcol)=x1val(k).^p;
  p=p+1;
  else
  Xflatval(i,jcol)=x2val(k1).^(p-1);
  end
  end
  p=1;
```

```
  jcol=2*index+2;
  for p1=1:index-1
      for p2=1:index-1
        if(p1+p2<=index)
         Xflatval(i,jcol)=x1val(k).^p1.*x2val(k1).^p2;
         jcol=jcol+1;
        end
      end
  end
  k1=k1+1;
end
k=k+1;
k1=1;
end

poz=1;
for i=1:id.dims% transformation of the output indentification data to a
coloumn vector
    for j=1:id.dims
    Yflat(poz,1)=y(i,j);
    poz=poz+1;
    end
end

poz=1;
for i=1:length(yval)% transformation of the output validation data to a
coloumn vector
    for j=1:length(yval)
        yvalT(poz,1)=yval(i,j);
        poz=poz+1;
    end
end

theta=Xflat\Yflat;% computation of the paramaters of the regressor from the
identification data
Yhatval=Xflatval*theta;% computation of the new output from the validation
data and the regressor parameters obtained
Yhatid=Xflat*theta;% computation of the new ouput from the identification
data and the regressor parameters obtained

MSEid=0;% calculation of the MSE from the approximator and the validation
data
for i=1:length(Yflat)
   MSEid=MSEid+1/length(Yflat)*((Yhatid(i,1)-Yflat(i,1)).^2);
end
mseid(count)=MSEid;% addition of every MSE to a mse vector for the purpose
of determining the best order fit

MSE=0;% calculation of the MSE from the approximator and the validation
data
for i=1:length(yvalT)
   MSE=MSE+1/length(yvalT)*((Yhatval(i,1)-yvalT(i,1)).^2);
end

mseval(count)=MSE;% addition of every MSE to a mse vector for the purpose
of determining the best order fit
count=count+1

end
```

```
[ minim i]=min(mseval)% computation of the lowest MSE value and its
coresponding order
m=4:1:20;

figure,plot(m,mseid(4:20))% plot for the comparison of the MSE on the
identification data versus the degree
xlabel('The degree of the polynomial approximator');ylabel('MSE from
identification data');title('MSEid vs ORDER');

figure,plot(m,mseval(4:20))% plot for the comparison of the MSE on the
validation data versus the degree
xlabel('The degree of the polynomial approximator');ylabel('MSE from
validation data');title('MSEval vs ORDER');

%% After we select the degree for the best approximator polynomial function
we repeat the same process as before to compute the model

clear all
load('proj_fit_14','-mat');

x1=id.X{1};
x2=id.X{2};
y=id.Y;
yval=val.Y;
x1val=val.X{1};
x2val=val.X{2};
index=4;

Xflat(1:1681,1)=ones(1681,1);

p=1;
k=1;
k1=1;

for j=0:id.dims-1
for i=id.dims*j+1:id.dims*j+id.dims
  for jcol=2:2*index+1
  if(mod(jcol,2)==0)
  Xflat(i,jcol)=x1(k).^p;
  p=p+1;
  else
  Xflat(i,jcol)=x2(k1).^(p-1);
  end
  end
  p=1;
  jcol=2*index+2;
  for p1=1:index-1
      for p2=1:index-1
         if(p1+p2<=index)
         Xflat(i,jcol)=x1(k).^p1.*x2(k1).^p2;
         jcol=jcol+1;
         end
      end
  end
  k1=k1+1;
end
k=k+1;
k1=1;
end
```

```
Xflatval(1:5041,1)=ones(5041,1);

p=1;
k=1;
k1=1;
for j=0:val.dims-1
for i=val.dims*j+1:val.dims*j+val.dims
  for jcol=2:2*index+1
  if(mod(jcol,2)==0)
  Xflatval(i,jcol)=x1val(k).^p;
  p=p+1;
  else
  Xflatval(i,jcol)=x2val(k1).^(p-1);
  end
  end
  p=1;
  jcol=2*index+2;
  for p1=1:index-1
      for p2=1:index-1
        if(p1+p2<=index)
         Xflatval(i,jcol)=x1val(k).^p1.*x2val(k1).^p2;
         jcol=jcol+1;
        end
      end
  end
  k1=k1+1;
end
k=k+1;
k1=1;
end

k=1;

for i=1:id.dims
    for j=1:id.dims
    Yflat(k,1)=y(i,j);
    k=k+1;
    end
end

theta=Xflat\Yflat;
Yhatval=Xflatval*theta;

Yhatval=reshape(Yhatval,71,71);

figure(3);
mesh(x1val,x2val,Yhatval), xlabel("x1val"); ylabel("x2val");
zlabel("Yhatval")% plot of the obtained model for the approximator
title('Obtained model for the polynomial approximator');

[x1plot,x2plot]=meshgrid(x1,x2);
figure(1);
mesh(x1plot,x2plot,y);hold, xlabel("x1"); ylabel("x2"); zlabel("y");% plot
of the identification data set
title('Identification data set');

[x1plotval,x2plotval]=meshgrid(x1val,x2val);
figure(2);
mesh(x1plotval,x2plotval,yval);hold, xlabel("x1val"); ylabel("x2val");
zlabel("yval")% plot of the validation data set
```

```
title('Validation data set');

figure(4);
surf(x1val,x2val,Yhatval);% used to compare the obtained model with the
validation data
hold on

surf(x1val,x2val,yval','FaceAlpha',0.5), xlabel("x1val"); ylabel("x2val");
zlabel("yval")% used to compare the obtained model with the validation data
title('Comparison between the obtained model and the validation data set');
```