Project Assignment
System Identification 2020-2021

Part1
Fitting an Unknown Function

Written by:

Pîțan Timeea-Ioana 8/8

Suciu Sorina-Gabriela 8/8

Simion Alexandru Iulian 8/8

Table of Contents

1. Introduction

For the 1st part of the project, a data set of identification and validation data is given as two 'iddata' files. Each 'iddata' file contains the characteristic values for the inputs x_1, x_2 and the output y. It follows that, one must find the approximation of a function that has two inputs and one output.

In order to solve this problem, linear regression is the best option.

The function $\hat{y}$, which will be the polynomial approximation of the function f, has a configurable degree m. For the first two values of m, $\hat{y}(x)$ will be equal to:

$m = 1,\ \hat{y}(x) = [1,\ x_1,\ x_2] \cdot \theta = \theta_1 + \theta_2 x_1 + \theta_3 x_2$

$m = 2,\ \hat{y}(x) = [1,\ x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2] \cdot \theta = \theta_1 + \theta_2 x_1 + \theta_3 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \theta_6 x_1 x_2$

From the given example for $m = 1$ and $m = 2$ it is obvious that the regressors can be brought to the form $x_1^i \cdot x_2^j$, thus the only unknown variables we must find is θ which is an unknown parameter vector.

The purpose of this project is to show how to obtain the best approximation of the f function using linear regression.

It is required to create a MATLAB script that can create the approximator of the function for a configurable degree, m, that stands between 1 and 100. It means that for each degree, the regressors matrix must be computed and for determining the best case (the value of m that gives the best approximation) the Mean Squared Error for both the validation and the identification data should be assessed.

For $m = 2$, the regressors for the polynomial are $1, x_1, x_2,\ x_1^2,\ x_2^2,\ x_1^2 x_2^2$.

For solving the problem, knowledge from the 3rd part of the lecture and the previous labs are used.

2. MATLAB Procedure

MATLAB is used for creating and compiling three scripts that will be explained below.

2.1 MATLAB script: "project.m"

First, the data sets are imported in MATLAB (the identification and the validation data). For a better understanding of the problem, some plots are computed using the "mesh" function that was chosen because it is easier to analyze the data sets. The plots can be seen in both Fig. 1 and Fig. 2.

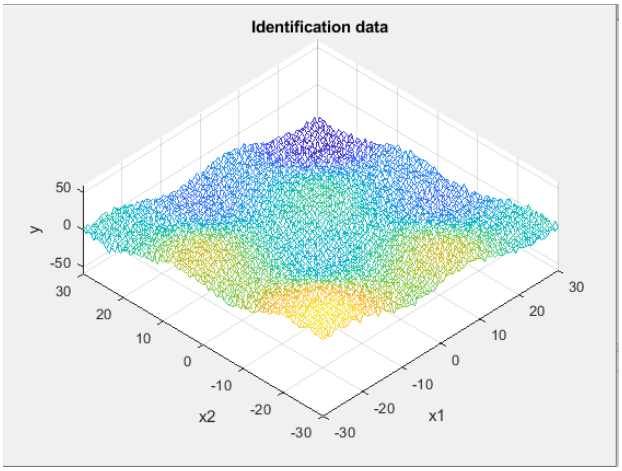

Fig. 1 Identification Data

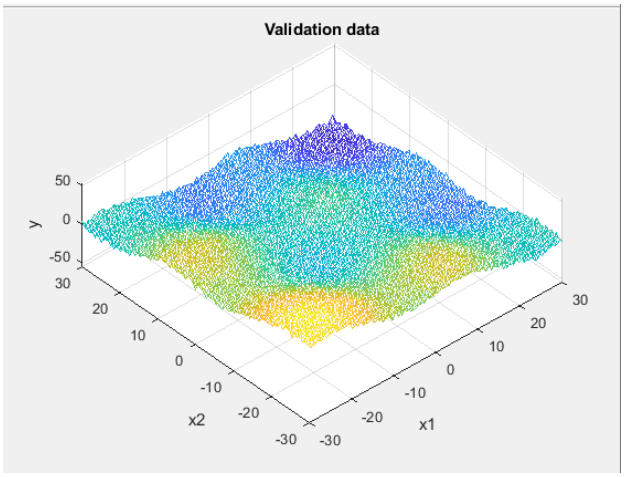

Fig. 2 Validation Data

The first for loop creates two vectors called MSE_{id} and MSE_{val} that hold the values of the Mean Squared Error of the identification and validation data for each degree of m. The function approx_f is used and it will be explained later in the report.

```
for order = 1:max_order
      display = 0;
      [MSEid(order), MSEval(order)] = approx_f(order, idx1, idx2, idy, valx1, valx2,
      valy, display);
end
```

The max_order parameter is a configurable parameter and represents the maximum degree of the polynomial (the m mentioned above).

The next step is a plot of the MSE values for both types of data in order to determine the m that will give the best approximation. By analyzing the graph in Fig. 3, it is easy to notice that the minimum value of MSE corresponds to a value of $m = 7$ and it will give the best approximation. For values of m above 7, the values of the MSE increase and the function approximation will be more and more inaccurate.

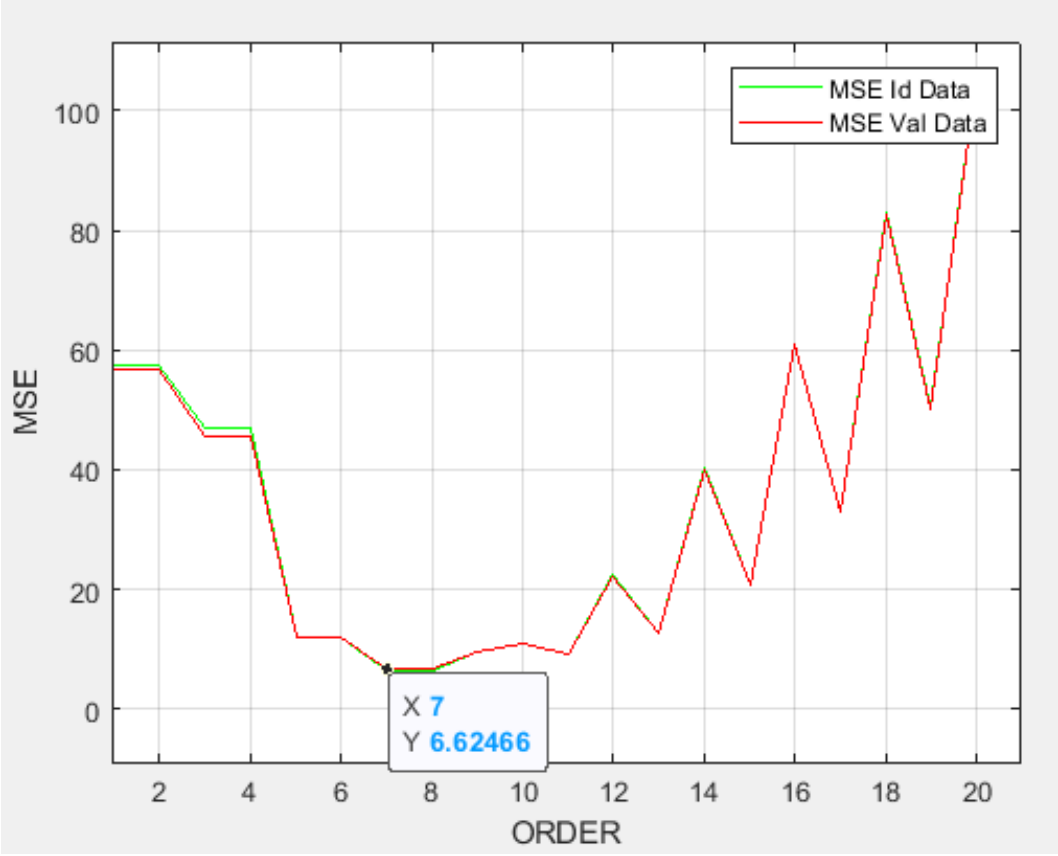

Fig. 3 MSE values for identification and validation data

Further, the minimum value of the MSE of the validation and the index where this value is found is selected and the approximated function is plotted in comparison to the real values, as seen in Fig. 4.

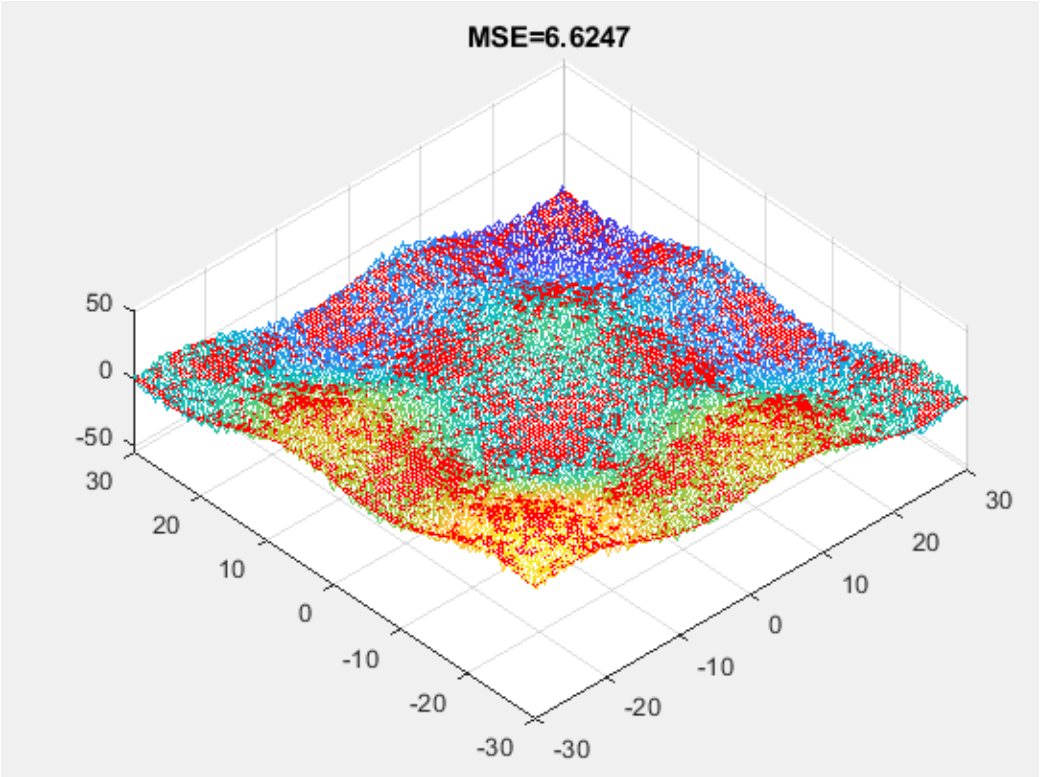

Fig. 4 Best function approximation for m=7

2.2 MATLAB function: "asssmeblepolinom.m"

```
function [out] = assemblepolinom(x1, x2, m)
```

Input parameters:

-x_1 → values of x1 (identification or validation)

-x_2 → values of x2 (identification or validation)

-m → the degree of the polynomial approximation

Output parameters:

-out → a matrix line where each cell represents the product of x1 and x2 to different powers

The function uses the formula, $x_1^i \cdot x_2^j$, presented in the introduction for computing a line of the regressors matrix.

2.3 MATLAB function: "approx_f.m"

```
function [idOUT, valOUT]=approx_f(order, x1, x2, y, x1val, x2val, yval, okay)
```

Input parameters:

-order -> the degree of m

-x_1 → vales of x1 for identification data

-x_2 → vales of x2 for identification data

-y → vales of y for identification data

-$x_1 val$ → vales of x1 for validation data

-$x_2 val$ → vales of x2 for validation data

-$yval$ → vales of y for validation data

-$okay$ → a control parameter which decides whether a plot is made or not

In the first for loop the regressor matrix, Rid, for the identification data and $yflat$ are computed. Using the output validation data, y, $yflat$ is computed (instead of a matrix, $yflat$ is a column vector).

After each line of the regressor is build and $yflat$ is computed, we can apply the formulas given in the lecture and find $theta$ and the approximated value of y, denoted $yhatid$. Now, having the real value and the approximated one, the MSE is computed. Analogically, for the validation data.

If the value of $okay$ equals 1, then a plot is made for the best approximation of the function (Fig. 4).

3. Conclusion

In conclusion, after all the computations, it follows that the best degree for the function is m=7. The maximum degree evaluated was m=20 and it resulted that for all the values of m above 7 the Mean Squared Error will increase leading to more and more inaccurate results.

4. Matlab Code

4.1 The script "project.m"

```
clc;
clear;
close all;
load ('proj_fit_08')

%Identification data
idx1 = id.X{1};
idx2 = id.X{2};
idy = id.Y;

%Validation data
valx1 = val.X{1};
valx2 = val.X{2};
valy = val.Y;

plot3(idx1, idx2, idy');

%Plotting data
figure;
mesh(idx1, idx2, idy');
title('Identification data');
xlabel('x1');
ylabel('x2');
```

```
zlabel('y');

figure;
mesh(valx1,valx2,valy');
title('Validation data');
xlabel('x1');
ylabel('x2');
zlabel('y');

max_order = 20;

%Building the polynomial
for order = 1:max_order
    display = 0;
    [MSEid(order), MSEval(order)] = approx_f(order, idx1, idx2, idy, valx1, valx2,
   valy, display);
end

t = 1:length(MSEid);
figure
plot(t, MSEid, 'g')
hold on;
 plot(t, MSEval, 'r')
hold off
grid
xlabel('ORDER');
ylabel('MSE');
legend('MSE Id Data', 'MSE Val Data');

%%minimal error and its index
errmin = min(MSEval);
for order = 1 : max_order
   if(MSEval(order) == errmin)
          ideal_order = order;
   end
end

%best approximation
display = 1;
approx_f(ideal_order, idx1, idx2, idy, valx1, valx2, valy, display);
```

4.2 First Function: assemblepolinom.m

```
function [out] = assemblepolinom(x1, x2, m)
index = 0;
for i = 0:m
    for j = 0:m
        if i+j <= m
            index = index + 1;
            out(index) = (x1^i)*(x2^j);
        end
    end
end
```

4.3 Second Function: approx_f.m

```
function [idOUT, valOUT]=approx_f(order, x1, x2, y, x1val, x2val, yval, okay)

m = order;
Rid = [];
yflat = [];

%Identification Data
for i = 1:length(x1)
    for j = 1:length(x2)
        Rid = [ Rid ; assemblepolinom(x1(i),x2(j),m)];
        yflat = [yflat ; y(i,j)];
    end
end

theta = Rid\yflat;
yhatid = Rid*theta;
e = yflat-yhatid;
MSE = 1/length(e)*sum(e.^2);
idOUT = MSE;

%Validation Data
Rval = [];
yflatval = [];
for i = 1:length(x1val)
    for j = 1:length(x2val)
        Rval = [Rval; assemblepolinom(x1val(i), x2val(j), m)];
        yflatval = [yflatval; yval(i,j)];
    end
end

yhatval = Rval*theta;
e2 = yflatval-yhatval;
MSEval = 1/length(e2)*sum(e2.^2);
valOUT = MSEval;

%Plotting the data
if okay == 1
    figure
    mesh(x1val ,x2val, yval');
    hold on
    mesh(xval,x2val,reshape(yhatval,length(x1val),length(x2val)));
    title(['MSE=', num2str(MSEval)])
end
end
```