TECHNICAL UNIVERSITY OF CLUJ-NAPOCA
FACULTY OF AUTOMATION AND COMPUTER SCIENCE

PART I

FITTING AN UNKNOWN FUNCTION

at

SYSTEM IDENTIFICATION

Sem. **I**, Year **2020-2021**

Students: Bodea Victor Lucian
Colda Andreea Ariana
Guran Delia

Group: 30332
Indicies: 01/01

Contents

Chapter 1. Introduction

The report presents a short description of the way in which was found a solution for fixing the problem. The simulations for the results are obtained in Matlab. The requirement is to find a polynomial approximator with a configurable degree.

The data sets are known from a Matlab data file, containing two inputs and one output of an unknown function. The problem is solved for the identification data, following that the results will be verified for the validation data.

Through linear regression method we can predict the evolution, the outcome of a variable, taking into account the values of another variable.

Chapter 2. The Description of the Algorithm

2.1. Input and Output Data

The input data is represented by X1 and X2 of size 1x41 for the identification and 1x71 for the validation.

The output data is represented by Y of size 41x41 of the identification and 71x71 for validation. It is also affected by noise and it is assumed to be zero-mean.

We create a system with a direct relation between the inputs and the output, so that the output in (i,j) will be equal to the input in (x1(i),x2(j)).

2.2. Identification

To find the function approximator the main formula is:

$$Y = \Phi * \theta$$

Where:

- Y is the output
- Φ is the regressor
- Θ is a vector of parameters

First, we generate the regressors in the variable Φ, which contains all the combinations of x1 and x2 at different powers. Then, the parameter Θ is computed from $\theta = \Phi\backslash Y$. This is called linear regression. Initially, Y is nxn and what we need is that Y is the size of $1xn^2$, where n in our case is 41, respectively 71.

Few examples of the approximator with different degrees of m, that leads us to find the general form are:

m=1, yhat(x) = [1, x1, x2]* θ = $\theta_1 + \theta_2 x1 + \theta_3 x2$
m=2, yhat(x) = [1, x1, x2, $x1^2$, $x2^2$, x1x2]* θ = $\theta_1 + \theta_2 x1 + \theta_3 x2 + \theta_4 x1^2 + \theta_5 x2^2 + \theta_6$x1x2

The input – output data will be processed and displayed in a 3D form (Fig. 2.1.) with the *mesh* command from Matlab.

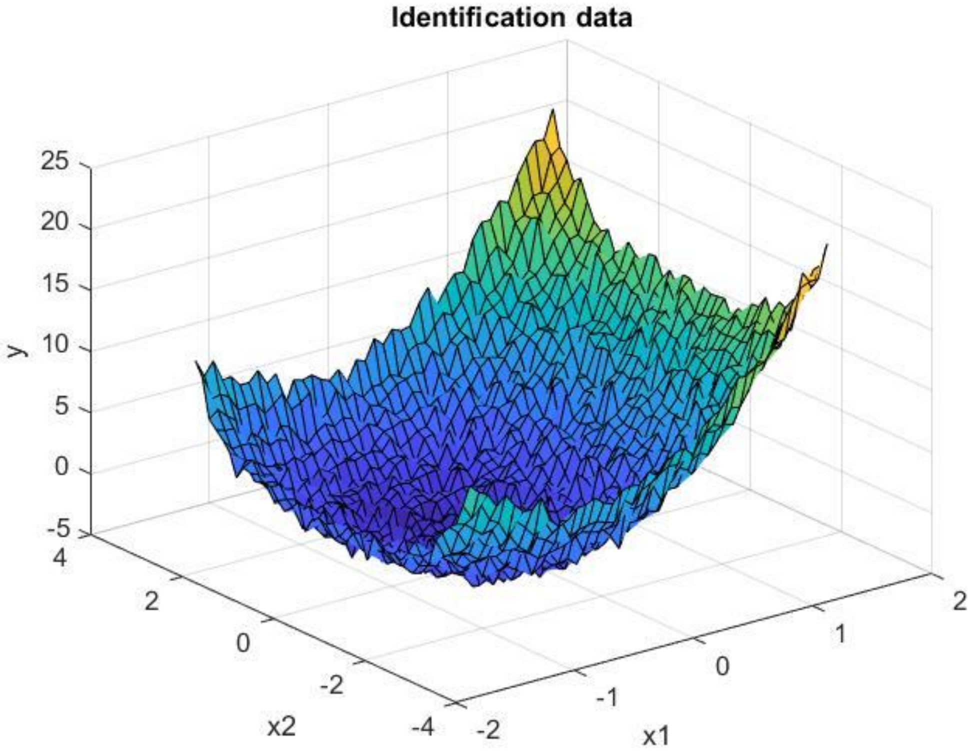

Fig. 2.1. Identification Data

We compute the matrix of regressor denoted by Φ, but we noted is as x. We add all the power combinations of x1 and x2 from 0 to m degree. Thus the general shape of a line is constructed. We notice that the x has 412 lines and sum from 1 to m+1 columns. We will work with a copy of the m value because we will need to decrease this value. The first value in the matrix is the multiplication of the first element from X1 and X2 vectors at power 0. Then, for the next m elements of the first line, we increase the power of x2 by 1 until it reaches the m value. At this point, we have written all the elements with x1 at power 0 and x2 at each power between 0 and m. Then, we decrease by 1 the value of the copy of m because that will be the maximum power x2 will reach now. And we start with the power of x2 again from 0. After that, we increase the power of x1 by 1. And again, we start multiplying x1 and x2, while the power of x2 increases by 1 until reaching the value of the copy of m decreased. The process repeats until m-copy decreases to 0, and that will be the last power of x2 and the last element in the line, while the power of x1 has reached the initial value of the m, and m-copy also. The same thing is done for the next lines but the index of X2 vector is increasing for every line, in order to have all the possible combinations of elements from X1 and X2 vectors. When the index of X2 becomes over 41, it comes back to 1 and we increase by 1 the index of X1. We repeat the algorithm until both X1 and X2 vectors reach the end.

```
index_x1 = 1;
index_x2 = 1;
x = ones((length(p.X{1})^2),sum(1:(m+1)));
for i = 1:length(p.X{1})^2
    if index_x2==42
       index_x1 = index_x1+1;
       index_x2 = 1;
    end
    power_x1 = 0;
    power_x2 = -1;
    copy_m = m;
    for j = 1:(sum(1:(m+1)))
        power_x2 = power_x2+1;
        x(i,j) = ((x1(index_x1))^(power_x1))*((x2(index_x2))^(power_x2));
        if power_x2==copy_m
           power_x2 = -1;
           copy_m = copy_m-1;
           power_x1 = power_x1+1;
        end
    end
    index_x2 = index_x2+1;
end
```

Fig. 2.2. Computing x

We have to modify the structure of Y (*y_array*) because in linear regression formula we need it as a column vector, *y_column*. After we computed x, the command to solve the linear system is matrix left division (\).

$$theta = x \backslash y\_column$$

An estimated y, *yhat_id*, is generated with the aid of previous calculations, which is used for mean-squared error formula. The vector *yhat_id* is brought to a matrix form, to be able to display the true values compared to the approximator outputs (Fig. 2.3.) depending on the input - output identification data set.

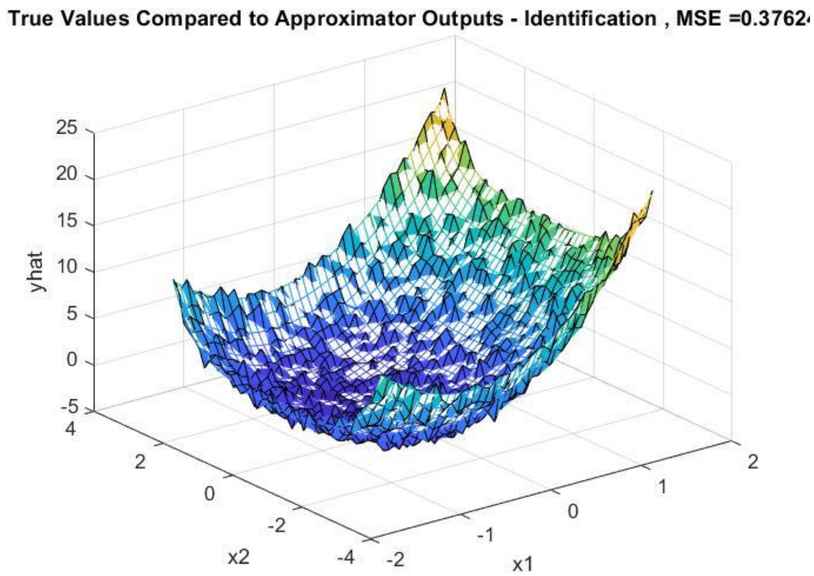

Fig. 2.3. Polynomial Approximator of Degree m=6

2.3. Validation

Another set of data is used for the validation (Fig. 2.4.).

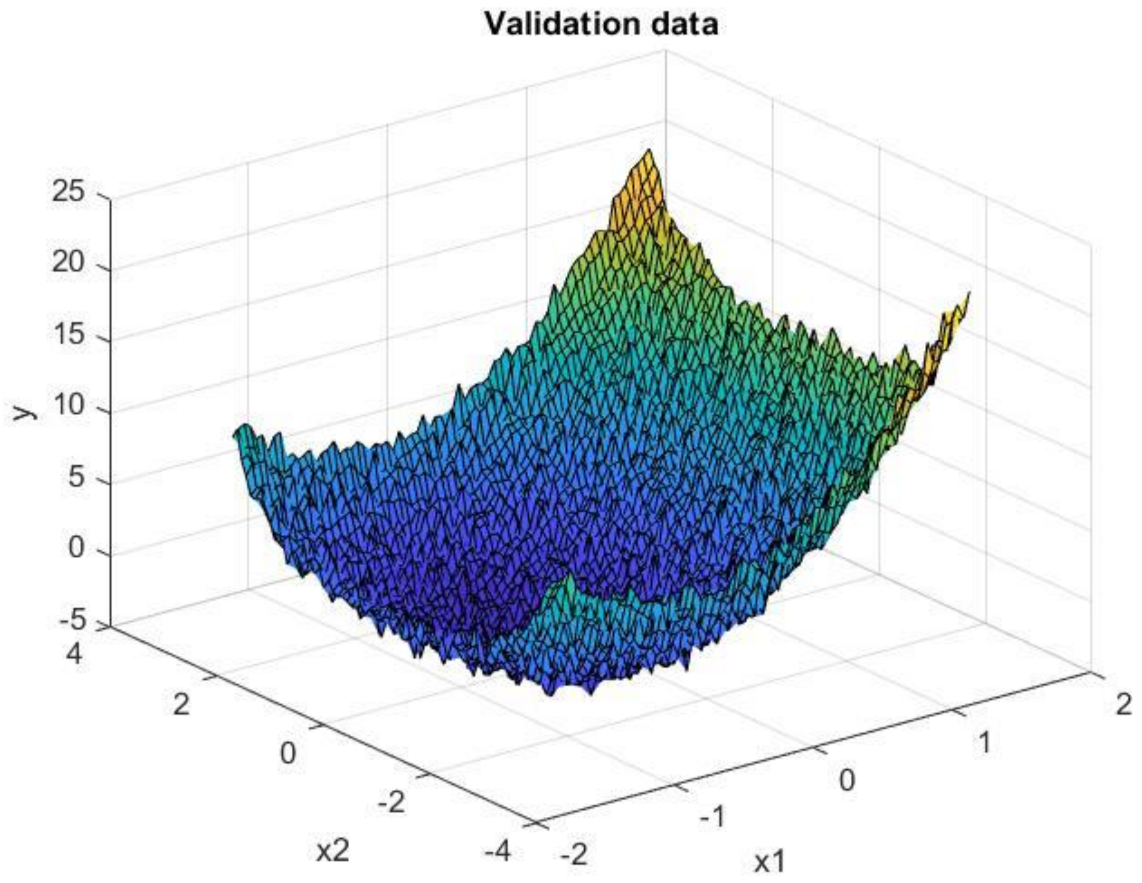

Fig. 2.4. Validation Data

It is analogous to the identification part, so the steps are resumed: build the matrix *xval*, of size 71^2xsum(1:(m+1)) and modify the structure of *yval_array* to *yval_column*, calculate *y_hat* and transform it into a matrix, *y_hat_matrix* and generate the mean-squared error.

A difference between the identification and validation data set is that for the validation we don't need to compute another theta and we use the one from the identification in order to verify the correctness of the implemented method.

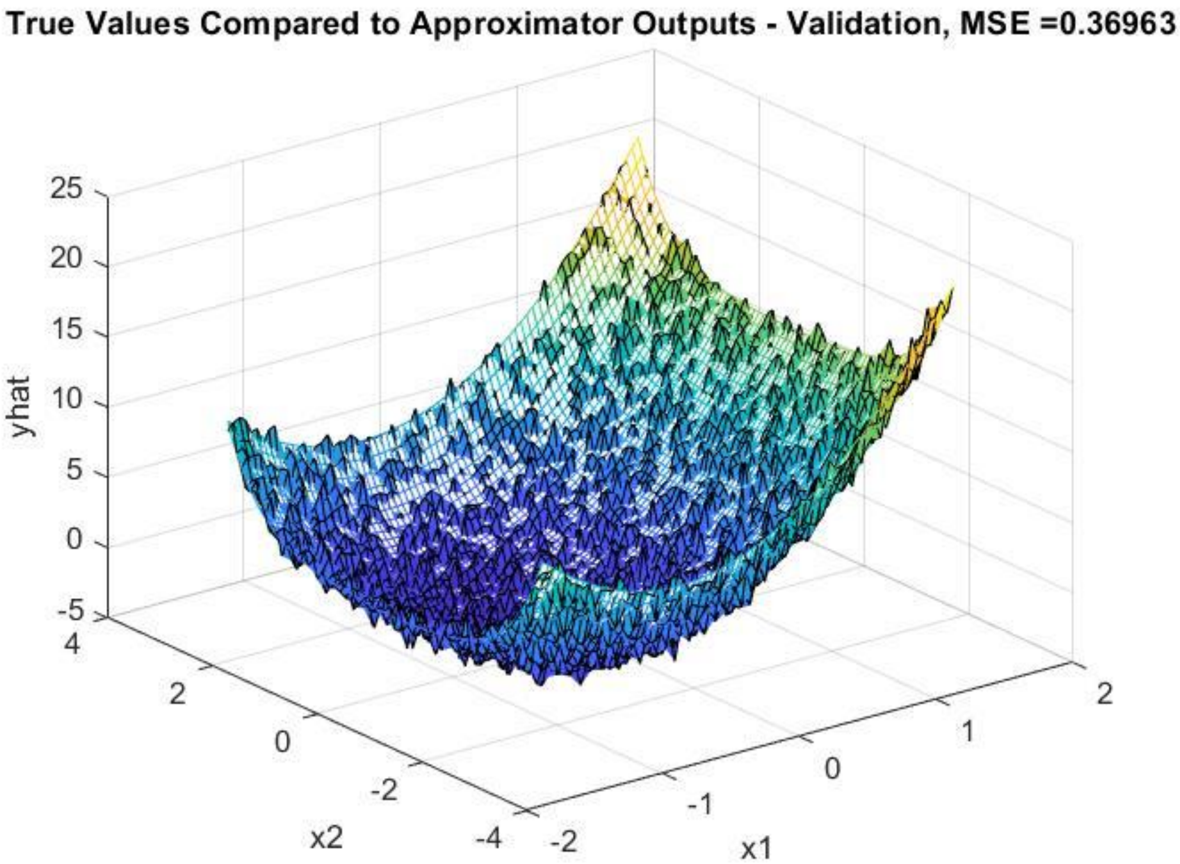

Fig. 2.5. Polynomial Approximator of Degree m=6

Chapter 3. Results and Conclusions

3.1. Mean-Squared Error

Through the mean-squared error we clarify what degree has the best approximation. Running the program for values of m from 1 to 10, the best approximation that we have is at degree m=6, MSE = 0.3696. (Fig. 3.1)

```
mse_minim = msevector(1);
M_mse_minim = 1;
for i = 2:m
    if msevector(i)<mse_minim
        mse_minim = msevector(i);
        M_mse_minim = i;
    end
end
```

Fig. 3.1. Minimum MSE for Validation

In Fig. 3.2. and Fig. 3.3. is represented all the mean-squared error values of degree 1 to 10, for both identification and validation data sets. It is pointed out the optimal value of m.

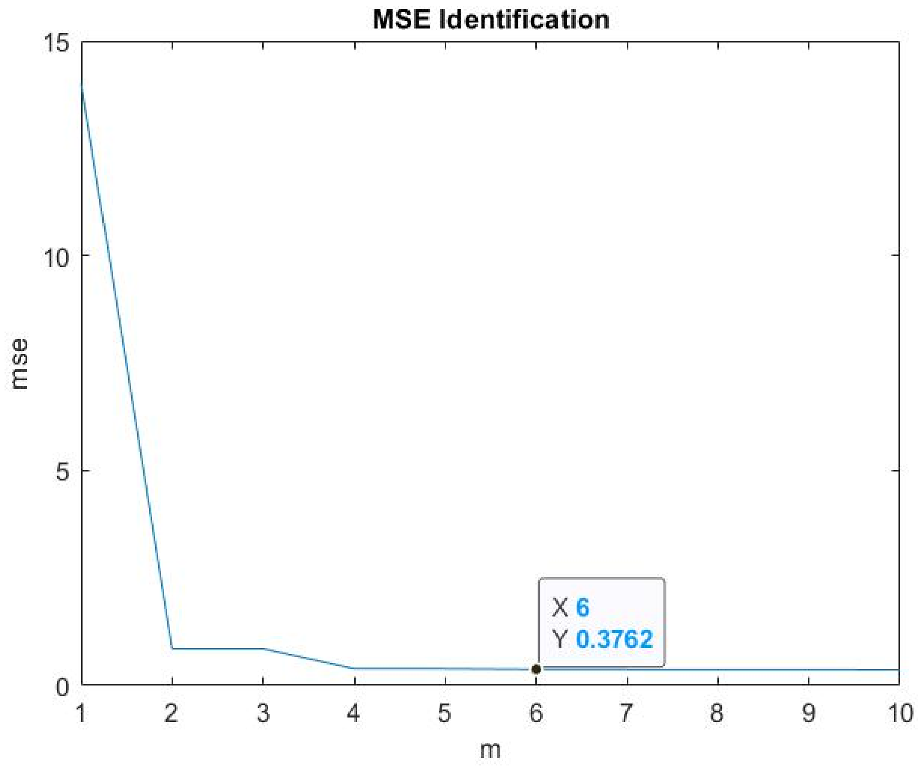

Fig. 3.2. MSE Values for Identification

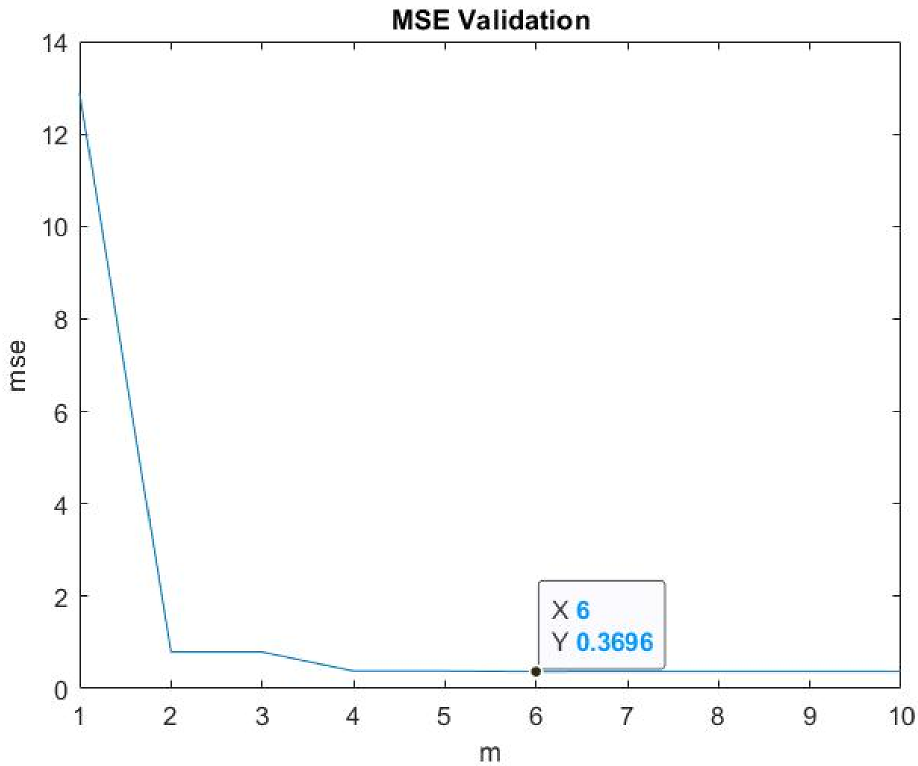

Fig. 3.2. MSE Values for Validation

ANNEXES

A.1. Matlab Code

```matlab
%% PART I: FITTING AN UNKNOWN FUNCTION
%%
clear
f = load('proj_fit_01.mat');
p = f.id;
q = f.val;

% read and plot the identification data
x1 = p.X{1};
x2 = p.X{2};
y = p.Y;
surf(x1,x2,y), title('Identification data');
xlabel('x1')
ylabel('x2')
zlabel('y')

% read and plot the validation data
x1val = q.X{1};
x2val = q.X{2};
yval = q.Y;
figure, surf(q.X{1},q.X{2},q.Y), title('Validation data');
xlabel('x1')
ylabel('x2')
zlabel('y')
```

```matlab

msevector = [];

for m = 1:10
    % for identification
    % computing the matrix phi noted with x
    index_x1 = 1;
    index_x2 = 1;
    x = ones((length(p.X{1})^2),sum(1:(m+1)));
    for i = 1:length(p.X{1})^2
        if index_x2==42
           index_x1 = index_x1+1;
           index_x2 = 1;
        end
        power_x1 = 0;
        power_x2 = -1;
        copy_m = m;
        for j = 1:(sum(1:(m+1)))
            power_x2 = power_x2+1;
            x(i,j) = ((x1(index_x1))^(power_x1))*((x2(index_x2))^(power_x2));
            if power_x2==copy_m
               power_x2 = -1;
               copy_m = copy_m-1;
               power_x1 = power_x1+1;
            end
        end
```

```
        index_x2 = index_x2+1;
    end

    % modify y form a matrix to a column
    index = 1;
    for i = 1:41
        for j = 1:41
            y_array(index) = y(i,j);
            index = index+1;
        end
    end
    y_column = transpose(y_array);

    % using linear regression to obtain theta
    theta = x\y_column;

    % compute MSE for identification data
    yhat_id = x*theta;
    mse_id = 1/1681*sum((y_column-yhat_id).^2);
    mse_id_v(m) = mse_id;

    index_m = 1;
    yhat_id_matrix = ones(41,41);
    for i = 1:41
        for j = 1:41
```

```
            yhat_id_matrix(i,j) = yhat_id(index_m);
            index_m = index_m+1;
        end
    end

    % plot the true values compared to the approximator output
    figure, surf(x1,x2,y)
    hold on
    mesh(x1,x2,yhat_id_matrix), title(['True Values Compared to Approximator Outputs - Identification ,
    xlabel('x1')
    ylabel('x2')
    zlabel('yhat')

    % for validation
    % computing the matrix phi_val noted with xval
    index_x1val = 1;
    index_x2val = 1;
    xval = ones((length(q.X{1})^2),sum(1:(m+1)));
    for i = 1:length(q.X{1})^2
        if index_x2val==72
            index_x1val = index_x1val+1;
            index_x2val = 1;
        end
        power_x1val = 0;
        power_x2val = -1;
```

```
            copy_mval = m;
            for j = 1:(sum(1:(m+1)))
                power_x2val = power_x2val+1;
                xval(i,j)=((x1val(index_x1val))^(power_x1val))*((x2val(index_x2val))^(power_x2val));
                if power_x2val==copy_mval
                    power_x2val = -1;
                    copy_mval = copy_mval-1;
                    power_x1val = power_x1val+1;
                end
            end
            index_x2val = index_x2val+1;
        end

        % modify yval form a matrix to a column
        index = 1;
        for i = 1:71
            for j = 1:71
                yval_array(index) = yval(i,j);
                index = index+1;
            end
        end
        yval_column = transpose(yval_array);

        y_hat = xval * theta;

        index = 1;
        y_hat_matrix = ones(71,71);
        for i = 1:71
            for j = 1:71
                y_hat_matrix(i,j) = y_hat(index);
                index = index+1;
            end
        end

        % compute MSE for validation
        mse = 1/5041*sum((yval_column-y_hat).^2);
        msevector(m) = mse

        % plot the true values compared to the approximator output
        figure, surf(x1val,x2val,yval)
        hold on
        mesh(x1val,x2val,y_hat_matrix), title(['True Values Compared to Approximator Outputs - Validation,
        xlabel('x1')
        ylabel('x2')
        zlabel('yhat')

    end
```

```
% finding the minimum MSE for validation and the most accurate m
mse_minim = msevector(1);
M_mse_minim = 1;
for i = 2:m
    if msevector(i)<mse_minim
        mse_minim = msevector(i);
        M_mse_minim = i;
    end
end

figure, plot(mse_id_v), title('MSE Identification');
xlabel('m');
ylabel('mse');
figure, plot(msevector), title('MSE Validation');
xlabel('m');
ylabel('mse');

mse_minim
M_mse_minim

```