Technical University of Cluj-Napoca

Faculty of Automation and Computer Science

System Identification Project 2021 - 2022
Part 1

Students: **Crișan Diana**

Bădău Elena Nicoleta

Gavrilă Diana-Andreea

Group: 30332

Group indices: **03 / 20**

Contents

Introduction and Problem Description

The regression is a statistical method used to predict a real value. To achieve this, the relationship between one or more variables (called independent variable) and the response (called the dependent variable) is modelled, starting from measured data.

The goal of the project is to model an unknown function, using linear regression. The initial data is separated in two sets. To avoid overfitting, the approximator of the function will be determined using the first set, called identification set, but it will be verified on the second one, called validation set. In fact, several approximators will be determined and the best one will be chosen.

Procedure

The data set

The given data set is separated into an identification set and a validation set.

Each one of these sets, contains the unknown function, $y(x_1, x_2)$, which given as a set of input and output variables: two input variables, x_1 and x_2, and one output variable, y.

In what follows, the procedure of finding the approximated function will be explained. At this point, it is important to mention that this function will be calculated using the identification set and it will be validated on the validation set.

The polynomial approximator

For the approximated function, the notation $\hat{y}$ is used, having the following structure:

$$\hat{y}(x_1, x_2) = \sum_{a+b \leq m} \theta_{a,b} \cdot x_1^a \cdot x_2^b$$

This is a polynomial in two variables of degree m, with n terms.

Next, we will show that the number of terms is $n = \frac{m^2 + 3m + 2}{2}$.

$$\hat{y}(x_1, x_2) = \sum_{a+b \leq m} \theta_{a,b} \cdot x_1^a \cdot x_2^b$$

$$\begin{aligned}
&\text{for } a = 0 && \Rightarrow b \in \{0, 1, 2, \ldots, m\} && \Rightarrow m + 1 \text{ terms} \\
&\text{for } a = 1 && \Rightarrow b \in \{0, 1, 2, \ldots, m-1\} && \Rightarrow m \text{ terms} \\
&\text{for } a = 2 && \Rightarrow b \in \{0, 1, 2, \ldots, m-2\} && \Rightarrow m - 1 \text{ terms} \\
&\ldots \\
&\text{for } a = m-1 && \Rightarrow b \in \{0, 1\} && \Rightarrow 2 \text{ terms} \\
&\text{for } a = m && \Rightarrow b \in \{0\} && \Rightarrow 1 \text{ term}
\end{aligned}$$

$$\Rightarrow n = 1 + 2 + \ldots + (m+1) = \frac{(1 + (m+1))) \cdot (m+1)}{2} = \frac{(m+2)(m+1)}{2} = \frac{m^2 + 3m + 2}{2}$$

Therefore, we use the polynomial linear regression, which is a type of linear regression. It is called linear because $\widehat{y}$ is a linear combination of the coefficients $\theta_{a,b}$.

For example, the polynomial approximator of degree $m = 2$, is the following one, where $\varphi_i(x_i)$ are the regressors and θ_i are the parameters:

$$\begin{aligned}\widehat{y}(x_1, x_2) &= \theta_1 x_1^2 + \theta_2 x_2^2 + \theta_3 x_1 + \theta_4 x_2 + \theta_5 x_1 x_2 + \theta_6 \\ &= \theta_1 \varphi_1(x_1) + \theta_2 \varphi_2(x_2) + \theta_3 \varphi_3(x_1) + \theta_4 \varphi_4(x_2) + \theta_5 \varphi_5(x_1, x_2) + \theta_6\end{aligned}$$

As a general form:

$$\begin{aligned}\widehat{y}(x_1, x_2) &= \theta_1 \varphi_1(x_1) + \theta_2 \varphi_2(x_2) + \ldots + \theta_{n-1} \varphi_{n-1}(x_1, x_2) + \theta_n \\ &= \begin{bmatrix} \varphi_1(x_1) & \varphi_2(x_2) & \ldots & \varphi_{n-1}(x_1, x_2) & 1 \end{bmatrix} \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \\ \ldots \\ \theta_{n-1} \\ \theta_n \end{bmatrix}\end{aligned}$$

Obtaining the parameters

To begin with, we know that for the identification set, each value $\hat{y}_k(x_1, x_2)$ should match $y_k(x_1, x_2)$, meaning that the approximated outputs should match the given ones.

Firstly, we want to explain the way in which the index k is used. As shown in Figure 1, we consider the index i for the samples in the independent variable x_1, and the index j for the samples in x_2. Since the number of samples is N_{ident} for both, $k \in \left[1, N_{\text{ident}}^2\right]$.

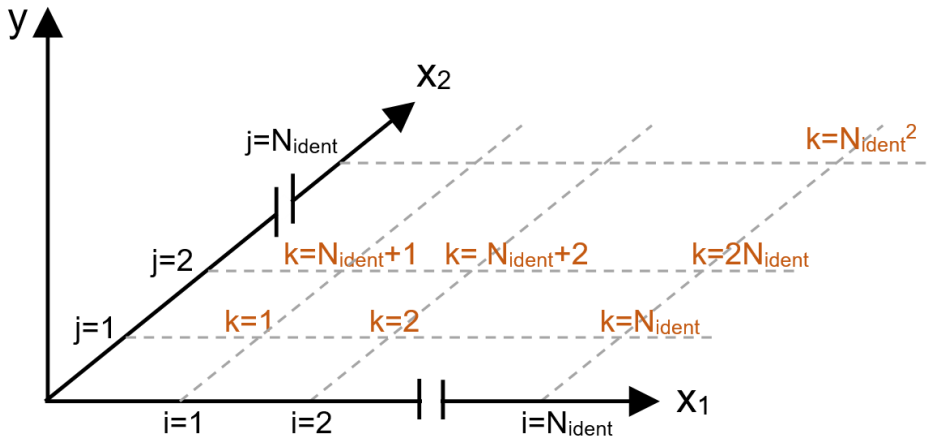


Figure 1. Indices used for the independent and the dependent variables

Therefore, the values of the index k are shown in Table 1.

Table 1. Indices used for the independent and the dependent variables

Index for x_2	Index for x_1	Index for y
$j = 1$	$i = 1$	$k = 1$
	$i = 2$	$k = 2$
	$\dots$	
	$i = N_{\text{ident}}$	$k = N_{\text{ident}}$
$j = 2$	$i = 1$	$k = N_{\text{ident}} + 1$
	$i = 2$	$k = N_{\text{ident}} + 2$
	$\dots$	
	$i = N_{\text{ident}}$	$k = 2 \cdot N_{\text{ident}}$
	$\dots$	
$j = N_{\text{ident}}$	$i = N_{\text{ident}}$	$k = N_{\text{ident}}^2$

Consequently, by writing the equations for each sample point, we obtain:

$$\begin{aligned} y_{k=1}(x_1, x_2) &= \widehat{y}_{k=1}(x_1, x_2) \\ &= \theta_1 \varphi_1(x_{1(k=1)}) + \theta_2 \varphi_2(x_{2(k=1)}) + \ \ldots + \theta_{n-1} \varphi_{n-1}(x_{1(k=1)}, x_{2(k=1)}) + \theta_n \end{aligned}$$

$$\begin{aligned} y_{k=2}(x_1, x_2) &= \widehat{y}_{k=2}(x_1, x_2) \\ &= \theta_1 \varphi_1(x_{1(k=2)}) + \theta_2 \varphi_2(x_{2(k=2)}) + \ \ldots + \theta_{n-1} \varphi_{n-1}(x_{1(k=2)}, x_{2(k=2)}) + \theta_n \end{aligned}$$

$\ldots$

$$\begin{aligned} y_{k=N^2_{\text{ident}}}(x_1, x_2) &= \widehat{y}_{k=N^2_{\text{ident}}}(x_1, x_2) \\ &= \theta_1 \varphi_1\left(x_{1\left(k=N^2_{\text{ident}}\right)}\right) + \theta_2 \varphi_2\left(x_{2\left(k=N^2_{\text{ident}}\right)}\right) + \ \ldots + \theta_{n-1} \varphi_{n-1}\left(x_{1\left(k=N^2_{\text{ident}}\right)}, x_{2\left(k=N^2_{\text{ident}}\right)}\right) + \theta_n \end{aligned}$$

Therefore:

$$\begin{bmatrix} y_{k=1}(x_1, x_2) \\ y_{k=2}(x_1, x_2) \\ \ldots \\ y_{k=N^2_{\text{ident}}}(x_1, x_2) \end{bmatrix} = \begin{bmatrix} \varphi_1(x_{1(k=1)}) & \varphi_2(x_{2(k=1)}) & \ldots & \varphi_{n-1}(x_{1(k=1)}, x_{2(k=1)}) & 1 \\ \varphi_1(x_{1(k=2)}) & \varphi_2(x_{2(k=2)}) & \ldots & \varphi_{n-1}(x_{1(k=2)}, x_{2(k=2)}) & 1 \\ \ldots & \ldots & \ldots & \ldots & 1 \\ \varphi_1\left(x_{1\left(k=N^2_{\text{ident}}\right)}\right) & \varphi_2\left(x_{2\left(k=N^2_{\text{ident}}\right)}\right) & \ldots & \varphi_{n-1}\left(x_{1\left(k=N^2_{\text{ident}}\right)}, x_{2\left(k=N^2_{\text{ident}}\right)}\right) & 1 \end{bmatrix} \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \\ \ldots \\ \theta_{n-1} \\ \theta_n \end{bmatrix}$$

By denoting the vector of output values by Y, the matrix of regressors by Φ and the vector containing the weights by θ, we obtain:

$$Y = \Phi \cdot \theta$$

By using the MATLAB backslash operator, the vector of parameters will be obtained:

$$\theta = \Phi \setminus Y$$

At this point, the approximated function is determined. However, the above-mentioned procedure is carried out for different degrees of the polynomial approximator, to find the best solution. This part will be fully explained in the following sections.

Implementation

For this project, we gave particular importance to writing clean code. Therefore, several functions were made. Their purpose is to avoid repetition, to improve code readability and reusability, and to reduce the chances of error.

On top of that, as a personal contribution, a comparison with the function `fitlm(tbl)` has been made. For this, only the functions 13 and 14 are used. The results are discussed in the following section.

1 - Function that retrieves the identification and validation data

Receives the name of a file.

Returns the identification and validation data.

```
function [x1_ident, x2_ident, y_ident, ...
          x1_valid, x2_valid, y_valid] = get_data(file_name)

    experiment_data = load(file_name);
    ident_data = experiment_data.id;
    valid_data = experiment_data.val;

    X_ident = ident_data(1).X;
    x1_ident = X_ident{1};
    x2_ident = X_ident{2};
    y_ident = ident_data(1).Y;

    X_valid = valid_data(1).X;
    x1_valid = X_valid{1};
    x2_valid = X_valid{2};
    y_valid = valid_data(1).Y;
end
```

2 - Function that computes the MSE

Receives two matrices y and $\hat{y}$.

Returns the mean squared error, MSE, by using the following formula:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2$$

```
function mse = get_mse(y, y_hat)
    mse = double(sum(sum((y - y_hat).^2))/numel(y));
end
```

3 - Function that finds all the pairs (a, b) for which $a + b \leq m$

This function will be used to find the polynomial of degree m in two variables, since each a and b represents the powers of one term of the polynomial.

Receives a number m.

Returns the vectors a and b, where $a(i)$ and $b(i)$ represent one pair of powers.

```
function [a, b] = get_power_pairs(m)
    n = (m^2 + 3*m + 2) / 2;

    a = zeros(n, 1);
    b = zeros(n, 1);

    index = 1;
    for i = 0 : m
        for j = 0 : m-i
            a(index) = i;
            b(index) = j;
            index = index + 1;
        end
    end
end
```

4 - Function that finds the polynomial of degree m in two variables x_1 and x_2, with n terms

Receives the degree of the polynomial m and the two variables x_1 and x_2.

Returns a vector containing the terms of the polynomial.

The number of number of terms is: $n = \frac{m^2 + 3m + 2}{2}$.

This function will be used for computing the matrix Φ and the approximated function $\hat{y}(x_1, x_2)$.

```
function poly = get_poly_two_var(m, x1, x2)

    [a, b] = get_power_pairs(m);

    poly = zeros(1, size(a, 1));
    for i = 1 : size(a, 1)
        poly(i) = x1^a(i) * x2^b(i);
    end
end
```

This function is the same as the above, but it does not use preallocating. In this way, it can also be used with symbolic variables.

```
function poly = get_poly_two_var_syms(m, x1, x2)

    [a, b] = get_power_pairs(m);

    for i = 1 : size(a, 1)
        poly(i) = x1^a(i) * x2^b(i);
    end
end
```

5 - Function that computes the matrix Φ

Receives the identification set, consisting of the two vectors that represent the independent variables and the degree of the polynomial, m.

Returns matrix of regressors, Φ.

```
function phi = get_phi(x1_ident, x2_ident, m)
    % number of parameters
    n = (m^2 + 3*m + 2)/2;

    % number of samples in the identification set
    N = size(x1_ident, 2);

    % index for the first and the second independent variable
    i = 1; j = 1;

    phi = zeros(N^2, n);
    for k = 1 : N^2
```

```
        % complete one row of the matrix phi
        phi(k, 1 : n) = get_poly_two_var(m, x1_ident(i), x2_ident(j));

        % update the i index
        i = i + 1;

        % when k reached N, 2N, ... , N^2
        if rem(k, N) == 0
            i = 1;
            j = k/N + 1;
        end
    end
end
```

6 - Function that computes the vector Y

Receives the dependent variable from the identification set.

Returns the vector obtained by concatenating all the columns from the original matrix.

```
function Y = get_Y(y_ident)
    Y = reshape(y_ident, 1, [])';
end
```

7 - Function that computes the vector θ

Receives the identification set, consisting of the two vectors that represent the indepenedent variables, and one vector that represents the dependent variable. It also receives the degree of the polynomial, m.

Returns the vector of parameters, θ.

```
function theta = get_theta(x1_ident, x2_ident, y_ident, m)
    phi = get_phi(x1_ident, x2_ident, m);
    Y = get_Y(y_ident);
    theta = phi\Y;
end
```

8 - Function that computes the matrix $\hat{y}(x_1, x_2)$

Receives the parameters θ, the independent variables (either from the identification or the validation set), x_1 and x_2, and the degree of the approximator polynomial, m.

Returns the approximated function as a matrix.

```
function y_hat = get_y_hat_from_theta(theta, x1, x2, m)
    % number of samples in the set
    N = size(x1, 2);

    % transpose the vector theta
    theta = theta';

    y_hat = zeros(N);
    for i = 1:N
        for j = 1:N
            % the approximated polynomial, with parameter values, as a vector
            y_hat_vector = (theta .* get_poly_two_var(m, x1(i), x2(j)));

            % the approximated polynomial, with parameter values, as a sum
            y_hat(i,j) = double(sum(y_hat_vector));
        end
    end
end
```

9 - Function that finds the MSEs for m in a specified interval

Receives the variables for the identification and validation set, and the interval for the degree of the polynomial, $m \in [m_{\text{start}}, m_{\text{stop}}]$.

Returns two vectors with the mean squared errors, corresponding both the identification and the validation set.

```
 function [mse_ident, mse_valid] = get_mse_ident_valid(x1_ident, x2_ident,
y_ident, ...
                                                       x1_valid, x2_valid,
y_valid, ...
                                                       m_start, m_stop)
    % preallocate to improve speed
    mse_ident = zeros(m_stop,1);
    mse_valid = zeros(m_stop,1);
```

```
    for m = m_start : m_stop
        % calculate the parameters on the identification set
        theta = get_theta(x1_ident, x2_ident, y_ident, m);

        % calculate the approximated functions on identification
        % and validation set
        y_hat_ident = get_y_hat_from_theta(theta, x1_ident, x2_ident, m);
        y_hat_valid = get_y_hat_from_theta(theta, x1_valid, x2_valid, m);

        % calculate the MSEs, for each degree m, for the identification
        % and validation set
        mse_ident(m) = get_mse(y_ident, y_hat_ident);
        mse_valid(m) = get_mse(y_valid, y_hat_valid);
    end
end
```

10 - Function that finds the best m and the best MSE in a specified interval

Receives the mean squared error for the validation set.

Returns the smallest mean squared error and the corresponding degree m.

```
function [m_best, mse_best] = get_best_m_mse(mse_valid)
    m = find(mse_valid);
    m_start = m(1); m_stop = m(end);

    % the degree of the polynomial for which mse is minimum
    m_best = find(mse_valid == min(mse_valid(m_start:m_stop)));
    % minimum mse
    mse_best = min(mse_valid(m_start:m_stop));
end
```

11 - Function that plots the MSEs and the minimum M

Receives two vectors with the mean squared errors, corresponding both the identification and the validation set.

Plots the two vectors, along with the smallest mean squared error and the corresponding degree m.

```
function plot_mse_ident_valid(mse_ident, mse_valid)
    [m_best, mse_best] = get_best_m_mse(mse_valid);

    plot(find(mse_ident), mse_ident(mse_ident~=0)); hold on;
    plot(find(mse_valid), mse_valid(mse_ident~=0));

    plot(m_best, mse_best, '*r');
    txt = ['m=',num2str(m_best), '    MSE=', num2str(mse_best)];
    text(m_best+0.5, mse_best+0.04, txt); hold off;

    xlabel('m'); ylabel('MSE');
    legend('identification set', 'validation set');
    title('The variation of MSE depending on the polynomial degree');
end
```

12 - Function that plots the approximated function

Receives the degree m, along with the identification set on which the matrix θ.

If these are the only arguments, the comparison will be also made for the identification set.

Otherwise, if the validation set is also received as arguments, the comparison will be made for it.

```
function plot_y_hat(m, comparison_for_ident, ...
                    x1_ident, x2_ident, y_ident, ...
                    x1_valid, x2_valid, y_valid)

    % compute theta for identification set
    theta = get_theta(x1_ident, x2_ident, y_ident, m);

    % check if comparison is made for identification or for validation
    if comparison_for_ident
        set_name = 'Identification set';
        x1 = x1_ident;
        x2 = x2_ident;
        y = y_ident;
    else
        set_name = 'Validation set';
        x1 = x1_valid;
        x2 = x2_valid;
        y = y_valid;
    end
```

```
    y_hat = get_y_hat_from_theta(theta, x1, x2, m);

    figure,
    subplot(211)
    mesh(x1, x2, y);
    xlabel('x1'); ylabel('x2'); zlabel('y');

    subplot(212);
    mesh(x1, x2, y_hat);
    xlabel('x1'); ylabel('x2'); zlabel('y hat');

    mse = get_mse(y, y_hat);
    sgtitle([set_name, ' for m = ', num2str(m), ' and mse = ', num2str(mse)]);
end
```

13 - Function that arranges the two independent variables as columns in a table

Receives the two independent variables, either from the identification or the validation set.

Returns two vectors which represent the variables rearranged as columns in a table, since the `fitlm()` function expects the variables, including the dependent one, to be column vectors.

```
function [x1_new, x2_new] = rearrange(x1, x2)
    x1_new = [];
    for i = 1:size(x1, 2)
        x1_new = [x1_new repmat(x1(i), 1, size(x1, 2))];
    end
    x2_new = repmat(x2, 1, size(x2, 2));
end
```

14 - Function that approximates the function using `fitlm()`

Receives the degree of the polynomial, m and the identification and validation data.

Returns the predicted values for the output variable.

```
function Y_predicted = get_Y_predicted(m, x1_ident, x2_ident, y_ident, ...
                                       x1_valid, x2_valid)
    % get the variables in the form required by fitlm()
    % as column vectors
```

```
    [x1_train, x2_train] = rearrange(x1_ident, x2_ident);
    [x1_test, x2_test] = rearrange(x1_valid, x2_valid);
    Y_train = get_Y(y_ident);

    % table with the variables
    tbl = table(x1_train', x2_train', Y_train, 'VariableNames', {'x1', 'x2',
'y'});

    % the polynomial of degree m
    syms x1 x2
    poly = char(sum(get_poly_two_var_syms(m, x1, x2)));
    formula = strcat('y ~ ', poly);

    model = fitlm(tbl, formula);
    Y_predicted = model.predict([x1_test' x2_test']);

    % rearrange the predicted values as a matrix
    Y_predicted = reshape(Y_predicted, [size(x1_valid,2), size(x1_valid,2)]);
end
```

Results and Representative Plots

The goal is to find the best degree of the polynomial approximator, which is the degree for which the mean squared error between the approximated function and the true one is minimum, on the validation set.

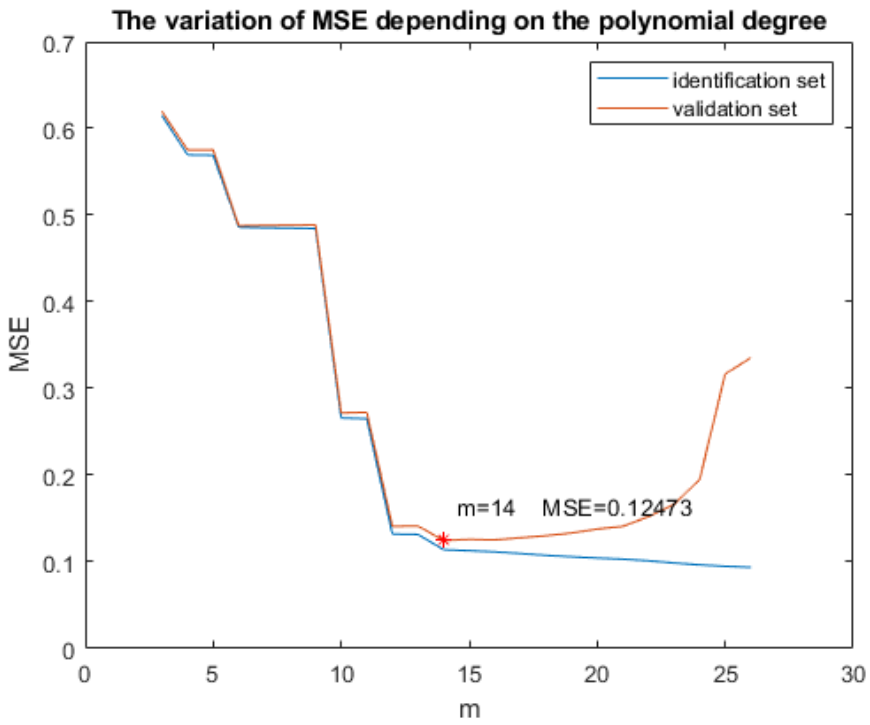
Therefore, we choose an interval for the degree to be analysed: $m \in \{m_{\text{start}}, m_{\text{start}} + 1, \ldots, m_{\text{stop}}\}$. For each degree, the following steps are made:

1. The vector of parameters θ is computed, on the identification set.
2. The approximated functions are computed for both the identification and the validation set.
3. The mean squared errors are computed for both the identification and the validation set.

The vectors of mean squared errors are plotted using the following code:

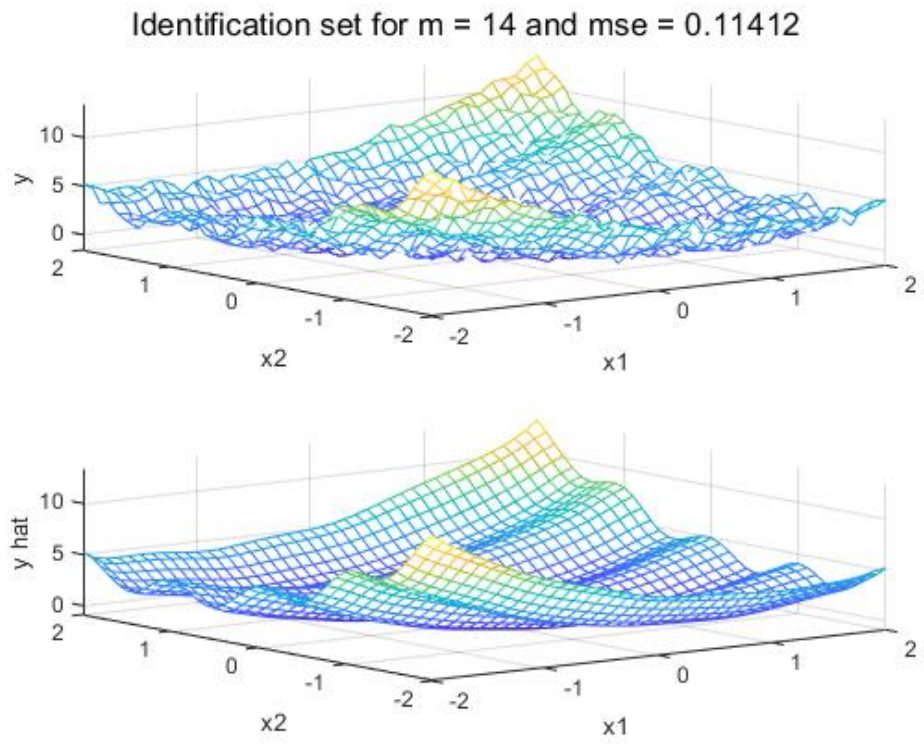
```
clear
close all
[x1_ident, x2_ident, y_ident, ...
 x1_valid, x2_valid, y_valid] = get_data("proj_fit_03.mat");

m_start = 3; m_stop = 26;
[mse_ident, mse_valid] = get_mse_ident_valid(x1_ident, x2_ident, y_ident, ...
                                             x1_valid, x2_valid,
y_valid, ...
                                             m_start, m_stop);
plot_mse_ident_valid(mse_ident, mse_valid);
```

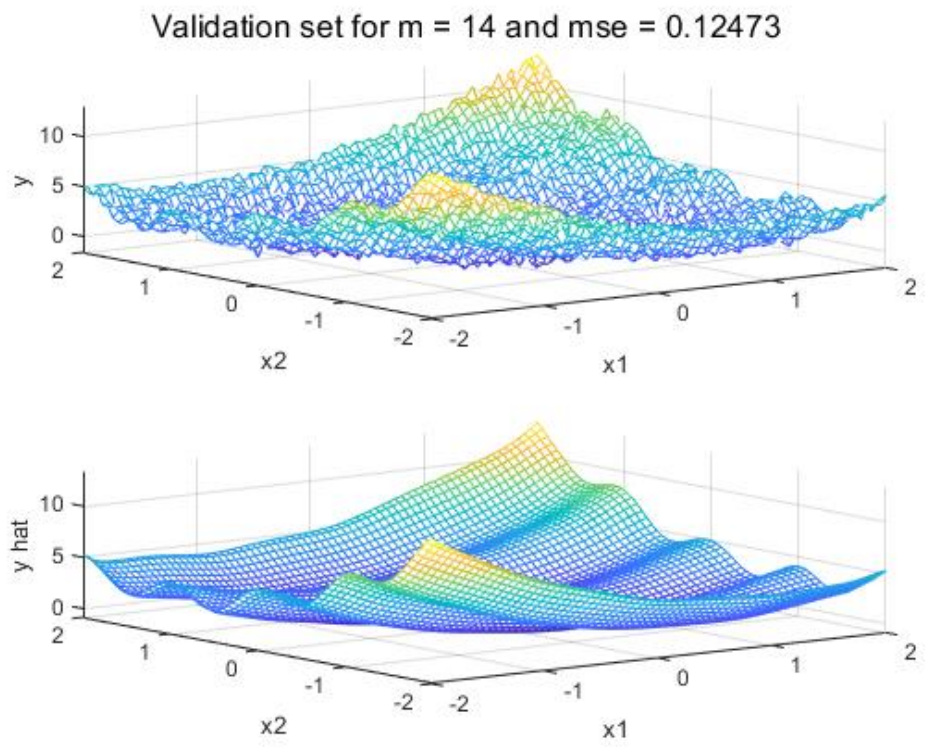


As one can observe, the mean squared errors for the identification set continue to decrease, which happened because the model starts modelling the noise, conducting to overfitting. Therefore, the minimum mean squared error is found on the validation set ($\text{mse} = 0.12473$), along with the corresponding degree ($m = 14$). For this particular degree of the polynomial, the comparison between the true and the approximated function is made, for both the identification and the validation set.

```
 [m_best, ~] = get_best_m_mse(mse_valid);
 plot_y_hat(m_best, true, x1_ident, x2_ident, y_ident, x1_valid, x2_valid,
y_valid)
```




```
 plot_y_hat(m_best, false, x1_ident, x2_ident, y_ident, x1_valid, x2_valid,
y_valid)
```



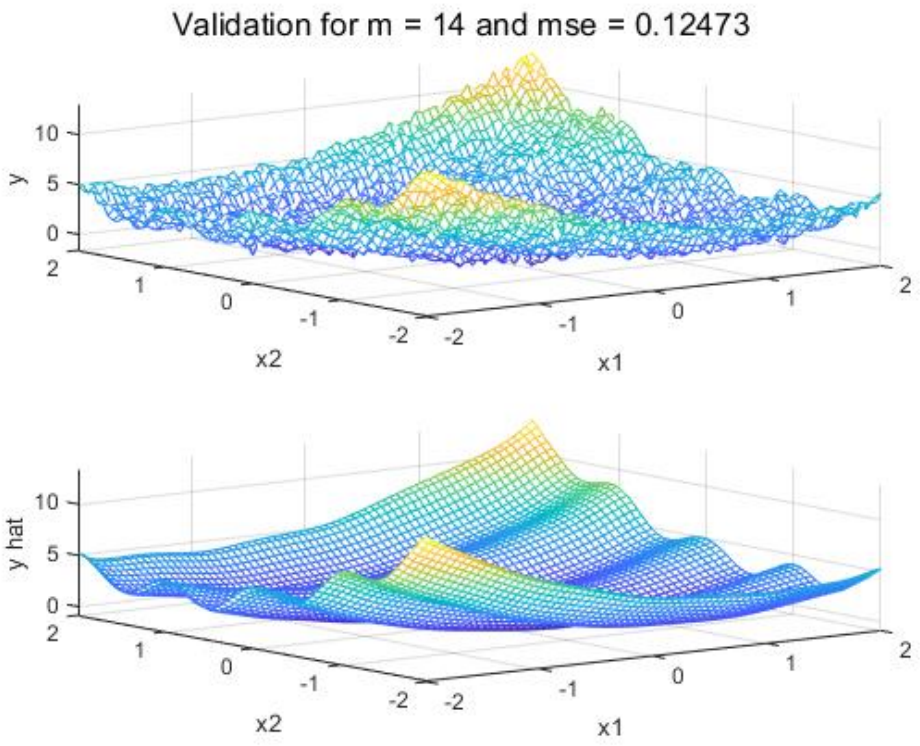
In what follows, an alternative method is used for this exact case of degree $m = 14$. This method uses the `fitlm()` function, which is already implemented in MATLAB, and belongs to the *Statistics and Machine Learning Toolbox*. To be able to use this function, the data is firstly arranged as columns in a table. Then, another argument that this function receives, is the model specification. For this, we create the polynomial approximator of the desired degree, with symbolic variables. The results are shown next.

```
m = 14;
Y_predicted = get_Y_predicted(m, x1_ident, x2_ident, y_ident, ...
                              x1_valid, x2_valid);
mse = get_mse(y_valid, Y_predicted);

figure,
subplot(211)
mesh(x1_valid, x2_valid, y_valid);
xlabel('x1'); ylabel('x2'); zlabel('y');

subplot(212);
mesh(x1_valid, x2_valid, Y_predicted);
xlabel('x1'); ylabel('x2'); zlabel('y hat');

sgtitle(['Validation for m = ', num2str(m), ' and mse = ', num2str(mse)]);
```



Discussion and Conclusion

As a conclusion, we find it important to summarise that upon receiving an unknown function, the parameters of the model are found on the identification set. The best polynomial approximator is found using the validation set, on which the mean squared errors are computed. Consequently, their minimum value indicates the degree of the polynomial approximator in two variables.