System Identification Project Part 1: Fitting an unknown function

Team: Project Fit 17

Moroșanu Ștefania Raluca

Cuc Ana Alexia

Capac Teodora

Table of contents

Problem Statement

- Having a nonlinear static function f, using a polynomial approximator, what would the model for this function be?
- What is the most optimal degree for which the approximated output is as close as possible to the true output?

Approximator structure and finding the parameters

Approximator structure:

- Two independent variables as input
- The polynomial expansion up to a configurable maximum degree
- Matrix of regressors(φ)

Finding the parameters:

- The regressor matrix is formed for each degree
- Including all possible combinations for each degree
- Combining the terms to form each polynomial regressor term
- Finding the vector of parameters(θ) using left division

```
theta = PHI \ Y;
```

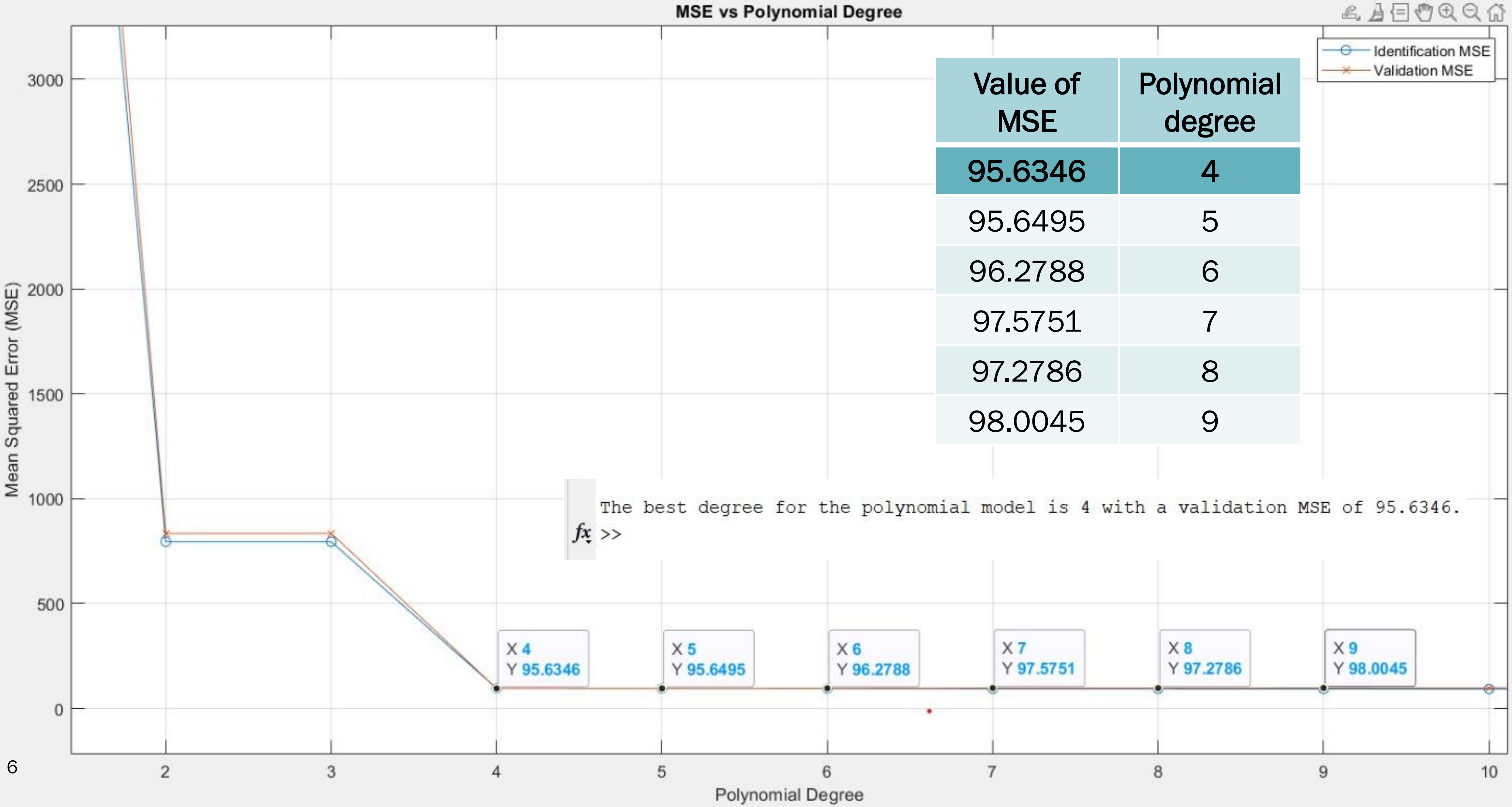
Key features

Constructing the regressor matrix

- Initially, φ is a column of ones.
- For each polynomial degree, all possible combinations of powers are included
- Exponents of X1 and X2 are computed so that they always sum up to the current degree.
- Φ is constructed by concatenation of columns which correspond to each polynomial term

$$\Phi = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

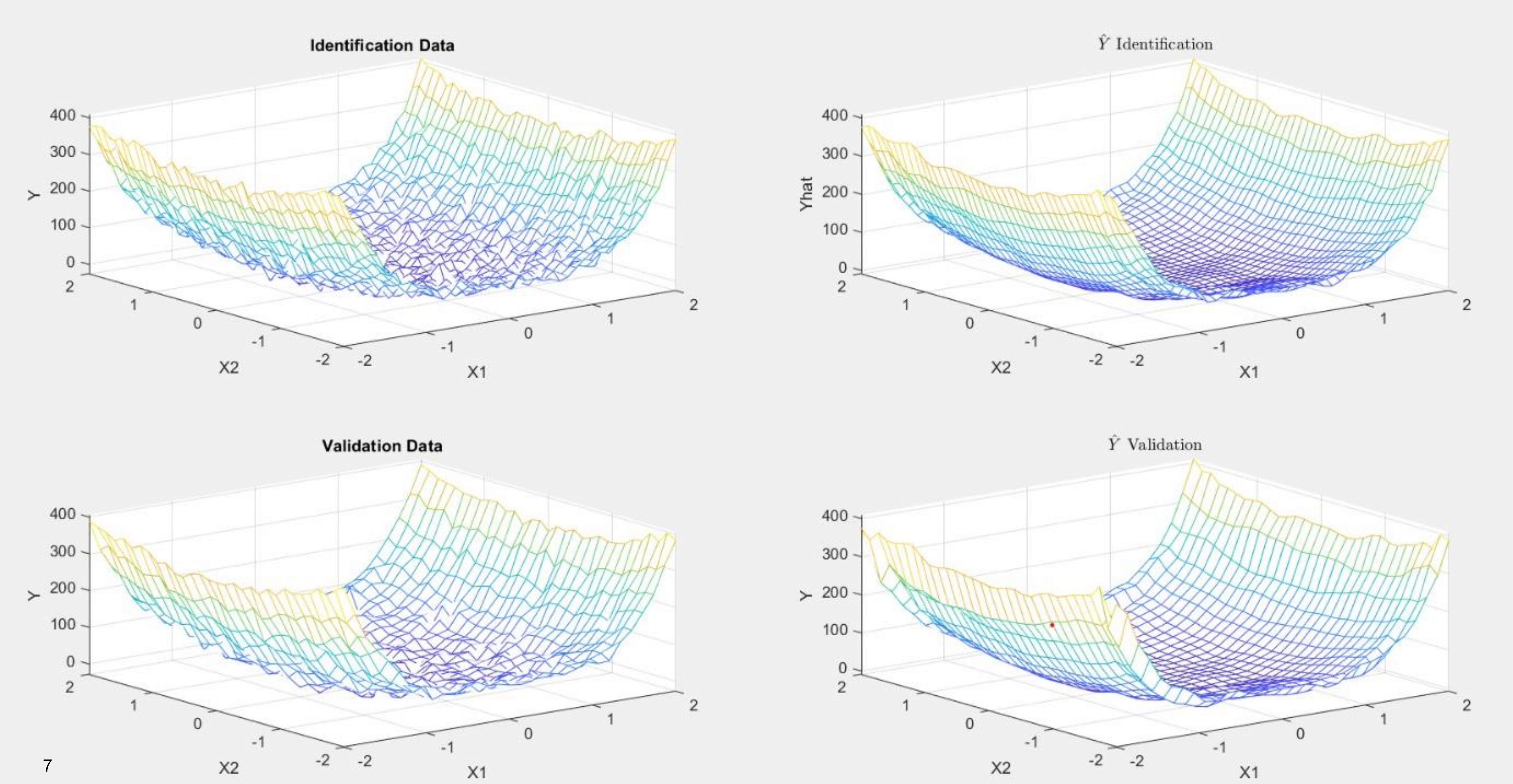
Tuning results



Value of MSE	Polynomial degree
95.6346	4
95.6495	5
96.2788	6
97.5751	7
97.2786	8
98.0045	9

```
The best degree for the polynomial model is 4 with a validation MSE of 95.6346.
fx >>
```

Representative plots



Conclusion

- **Goal:**
  - to build a polynomial approximator by using a matrix of regressors
  - to find the optimal degree for which the MSE has the lowest value
  - to find the polynomial model which has the best performance
- **How we achieved it:**
  - formed the regressor matrix and theta accordingly up to a maximum degree
  - compared the values of MSEs for each degree and took the minimum one
  - evaluated the representative plots until finding a maximum degree which fits performs accurately

Our team

Capac Teodora

-forming the matrix of regressors

Cuc Ana Alexia

-calculating the parameters of the approximator

Moroșanu Ștefania

-finding the optimal degree for the approximation according to the MSE

*data acquisition, plots, research and the presentation were a joined effort

Table of Contents


```
clear
clc

load('proj_fit_17.mat');
```

Declaring the identification and validation data

```
X1 = id.X{1};
X2 = id.X{2};
X1_val = val.X{1};
X2_val = val.X{2};
Y = id.Y(:);          %outputs reshaped into column vectors
Y_val = val.Y(:);
```

Define maximum polynomial degree

```
m_max = 25;

MSE_id = zeros(1, m_max);
MSE_val = zeros(1, m_max);
```

Finding the best polynomial model for our data

```
   %MSE for both the identification and validation datasets across different
polynomial degrees (m)
   %By tracking these MSE values, we identify which polynomial degree
results in the lowest
   % validation error (MSE_val), helping to select the most accurate model

for m = 1:m_max
    phi = form_phi(X1, X2, m); %builds polynomial order of degree m through
function form_phi

    theta = phi\Y;          %calculate model parameters

    yhat_id = phi * theta;

    MSE_id(m) = (1 / length(Y)) * sum((Y - yhat_id) .^ 2);
```

```
    phi_val = form_phi(X1_val, X2_val, m); %phi_val is a matrix where each
column represents
                                            %a different polynomial term up
to degree m for the validation data.

    yhat_val = phi_val * theta;
    MSE_val(m) = (1 / length(Y_val)) * sum((Y_val - yhat_val) .^ 2);
end
```

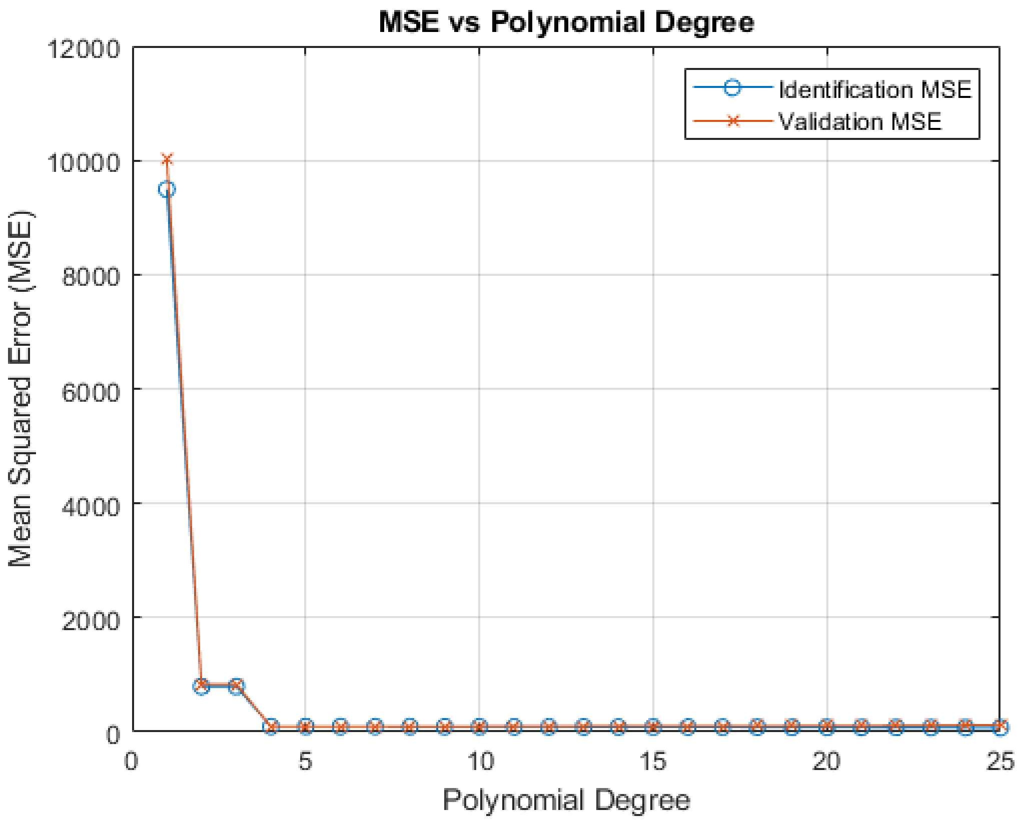
Find Best Polynomial Degree Based on Validation MSE

```
[best_MSE_val, best_degree] = min(MSE_val);
fprintf('The best degree for the polynomial model is %d with a validation
MSE of %.4f.\n', best_degree, best_MSE_val);
```

The best degree for the polynomial model is 4 with a validation MSE of 95.6346.

MSE vs. Polynomial Degree

```
figure;
plot(1:m_max, MSE_id, '-o', 'DisplayName', 'Identification MSE');
hold on;
plot(1:m_max, MSE_val, '-x', 'DisplayName', 'Validation MSE');
xlabel('Polynomial Degree');
ylabel('Mean Squared Error (MSE)');
legend show;
title('MSE vs Polynomial Degree');
grid on;
```



Mesh Plots of Identification and Validation Data

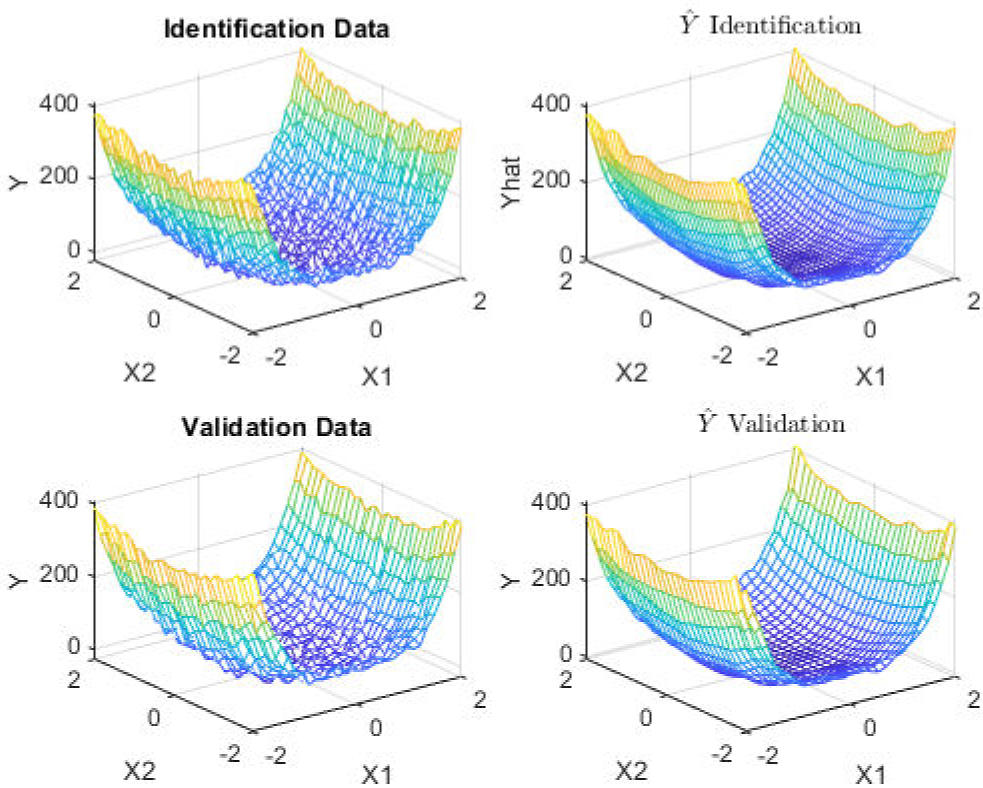
```
figure;
subplot(2, 2, 1);
mesh(X1, X2, id.Y);
title('Identification Data');
xlabel('X1');
ylabel('X2');
zlabel('Y');

subplot(2, 2, 2);
mesh(X1, X2, reshape(yhat_id, length(X2), length(X1)));     %Creates a 3D
mesh plot of the identification data s predicted values,
                                                             %reshaped to
match the dimensions of X1 and X2.
title('$\hat{Y}$ Identification', 'Interpreter', 'latex');
xlabel('X1');
ylabel('X2');
zlabel('Yhat');

subplot(2, 2, 3);
mesh(X1_val, X2_val, val.Y);
title('Validation Data');
xlabel('X1');
ylabel('X2');
```

```
zlabel('Y');

subplot(2, 2, 4);
mesh(X1_val, X2_val, reshape(yhat_val, length(X2_val), length(X1_val)));
title('$\hat{Y}$ Validation', 'Interpreter', 'latex');
xlabel('X1');
ylabel('X2');
zlabel('Y');
```



Construction of the Regressor Matrix

```
function phi = form_phi(X1, X2, degree)    %form_phi constructs the feature
matrix phi for a polynomial model of a specified degree.
    [X1, X2] = meshgrid(X1, X2);              %returns 2-D grid coordinates
based on the coordinates contained in vectors x and y
    phi = ones(prod(size(X1)), 1);           %initializes phi with a column of
ones to account for the constant term in the polynomial.

    for i = 1:degree
        for j = 0:i
            k = i - j;
            phi = [phi, (X1(:) .^ k) .* (X2(:) .^ j)];  % concatenates each
polynomial term to phi.
        end
```

```
    end
end
```

Published with MATLAB® R2023b