Team #10

Capătă Andrei-Bogdan
Rizoiu Tudor
Căstrase Sergiu-Gabriel

Fitting an unknown function

System Identification Project - Part 1

Table of Contents

Introduction / Problem Statement

Modelling the behavior of an unidentified nonlinear static function using a polynomial approximator

How can we do it?

Using **Polynomial Regression**

What is Polynomial Regression ?

Polynomial regression = method used in system identification to model the relationship between input and output variables in a dynamic system using Polynomial approximators.

Approximator Structure

- **building ‘PHI id’** - regressors matrix, for the identification dataset;
- **finding ‘theta’** - the coefficients of the unknown function, by using matrix left division method between ‘PHI id’ and the output of the identification data set ‘Y id’;
- **building ‘PHI val’** regressors matrix, for the validation dataset
- **obtaining ‘g hat’** - the approximated function by multiplying ‘PHI val’ and ‘theta’.

Key Features

Algorithm for generating ‘PHI’

Works for any dimensions of the input data and any degree of the polynomial

Generating the powers of X1 and X2 directly into the PHI algorithm using two simple FOR loops

Tuning Results - MSE

MSE measures how well a prediction or estimation method performs by calculating the average of the squared differences between predicted values and actual observations.

Lower MSE values indicate better performance

The plot in *Fig. 1* was made so that we could clearly see how the MSE evolves in comparison with the degree of the regressor.

Graphic representation of MSE

Using this graph we determined that the **optimal degree is 16**, obtaining a **MSE of 0.0067**

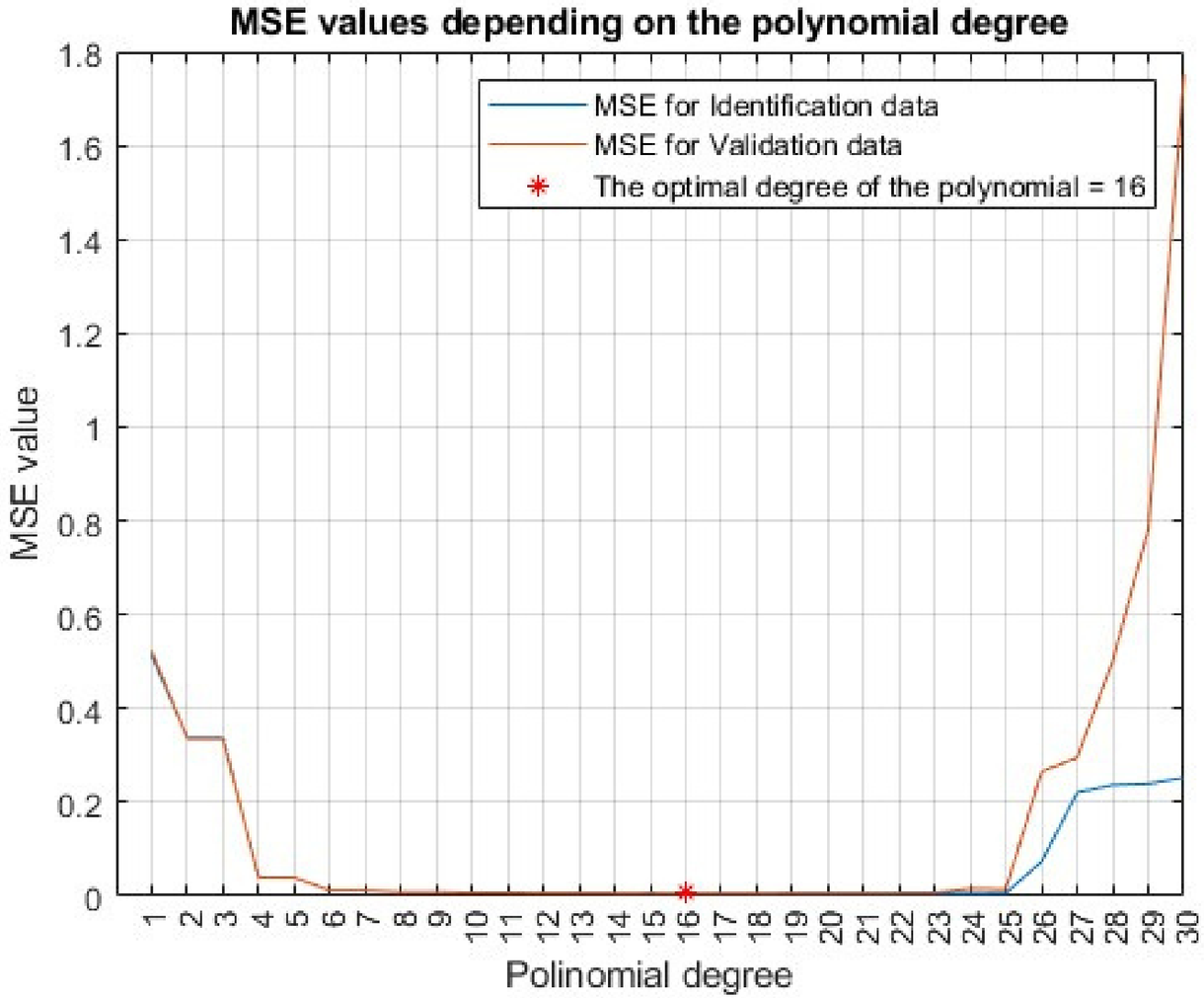

Fig. 1

Representative plots

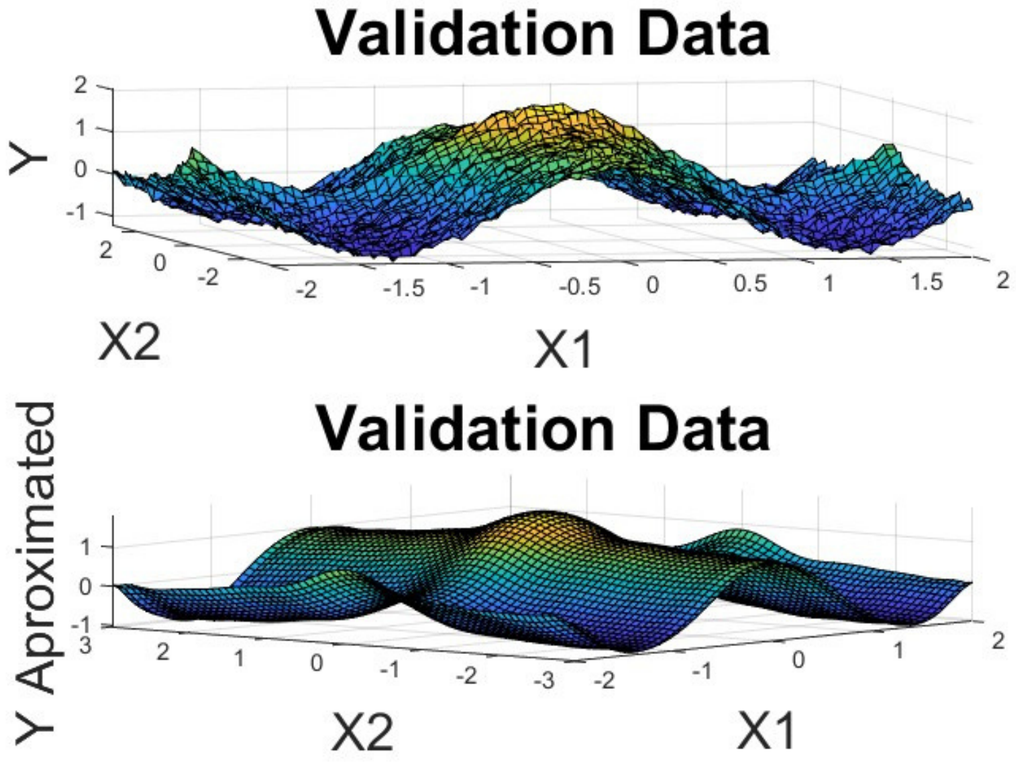

Fig. 2

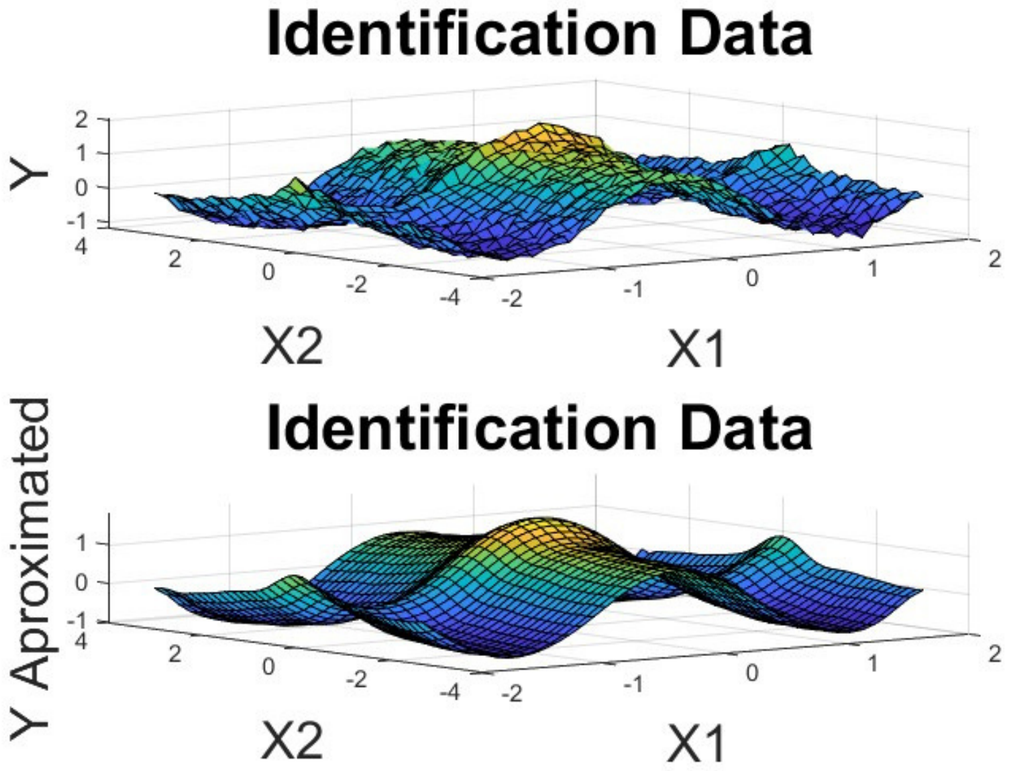

Fig. 3

Conclusion

Starting from a set of data corrupted by noise, we managed to find an approximated function.

This algorithm can be used in cases that require calculating an unknown function based on supplied training data.

Code Snippets

Algorithm for ‘PHI’ for the identification data set

```
% Initialize the PHI matrix with ones for identification
PHI = ones(id_dim(1) * id_dim(2), nrterms);

% Loop through data points to construct the polynomial basis functions for identification
for i = 1:id_dim(1) % construct 41 of j iteration to do the 41x41 rows
    for j = 1:id_dim(2) % construct first 41 rows in one repetitive structure
        k = 2; % because the first element is 1 k is the index that tells to go to the right
        for q = 1:l % Loops to construct powers from 1 to l
            for p = 0:q
                PHI((i - 1) * id_dim + j, k) = x1_id(i)^(q - p) * x2_id(j)^p;
                k = k + 1;
                % i go down 41 rows
                % j go down 1 row
                % k goes to right 1 column
                % q-p power of x1
                % p power of x2
            end
        end
    end
end

% Calculate the coefficients theta for identification dividing PHI matrix
% by left division by Y reshaped into a column vector
theta = PHI \ y_id(:);
```

Algorithm for ‘PHI’ for the validation data set

```
% Initialize the PHI matrix with ones
PHI_val = ones(val_dim(1) * val_dim(2), nrterms);

% Loop through data points to construct the polynomial basis functions for validation
for i = 1:val_dim(1) % construct 71 of j iteration to do the 71x71 rows
    for j = 1:val_dim(2) % construct first 71 rows in one repetitive structure
        k = 2; % because the first element is 1 k is the index that tells to go to the right
        for q = 1:m % Loops to construct powers from 1 to m
            for p = 0:q
                PHI_val((i - 1) * val_dim(1) + j, k) = x1_val(i)^(q - p) * x2_val(j)^p;
                k = k + 1;
                % i go down 71 rows
                % j go down 1 row
                % k goes to right 1 column
                % q-p power of x1
                % p power of x2
            end
        end
    end
end
```

Generating the approximated function “g hat” based on the validation dataset and theta from the identification dataset

```
% Calculate the aproximated values for validation in a column vector
% We use theta, identified in the identification dataset
ghat_val = PHI_val * theta;

% Reshape ghat into a matrix, because due to the properties of matrix
% multiplication ghat results in a column vector of dim(1)*dim(2) X 1
ghat_val = reshape(ghat_val, val_dim(1), val_dim(2));

% Calculate MSE
true_mse = mean((y_val(:) - ghat_val(:)).^2);
```

Thank you for your attention!

Please address any questions regarding our presentation.