Modelling the behaviour of an unknown static function using aproximated polynomials

By Rus Alexandru, Pal Robert and Suciu Andrei
Index 13

Problem Statement

Develop a polynomial model ***g*** to approximate an unknown, nonlinear function ***f*** with two input variables and noisy outputs.

Objectives:

- **Polynomial Model**
- **Model Fitting**
- **Model Evaluation**
- **Analysis**

Approximator Structure

- Polynomial order m is incrementally tested to find best fit.
- Polynomial terms are generated, creating a ***regression matrix***.

$$R = [1, x_1, x_2, x_1^2, x_1 x_2, x_2^2, \ldots]$$

- Parameters θ, are estimated by ***minimizing training error***.

$$\theta = (R^\top R)^{-1} R^\top y \qquad \hat{y} = R\theta$$

Approximator Structure

- ***MSE*** is calculated for ***training and validation*** data sets.

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

- The model selects the order with the ***least MSE*** as the optimal one.
- The parameters for the optimal model are saved for ***final predictions***.

Key features:

- A notable innovation in our solution is the dynamic ***polynomial term generator function***. This function efficiently creates all possible polynomial terms up to a specified degree ***m*** for variables ***x1*** and ***x2***.

Visualization of Polynomial Terms for Degree m = 3

Each term represents a combination of x1 and x2 raised to powers that sum to the total degree.

$$Degree\ 0:\ x_1^0$$

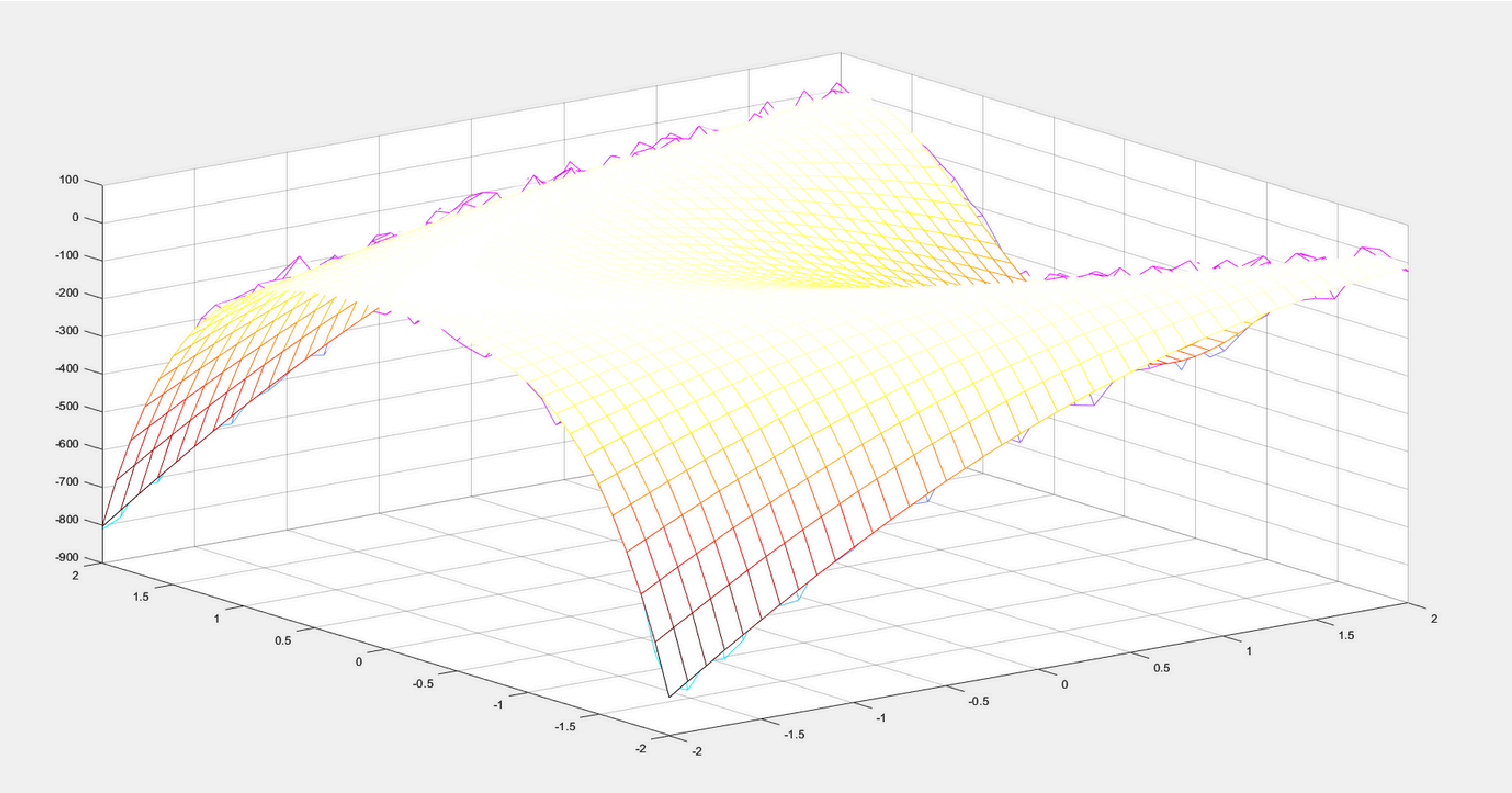
$$Degree\ 1: x_2^1 + x_1^1$$

$$Degree\ 2: x_2^2 + x_1^1 x_2^1 + x_1^2$$

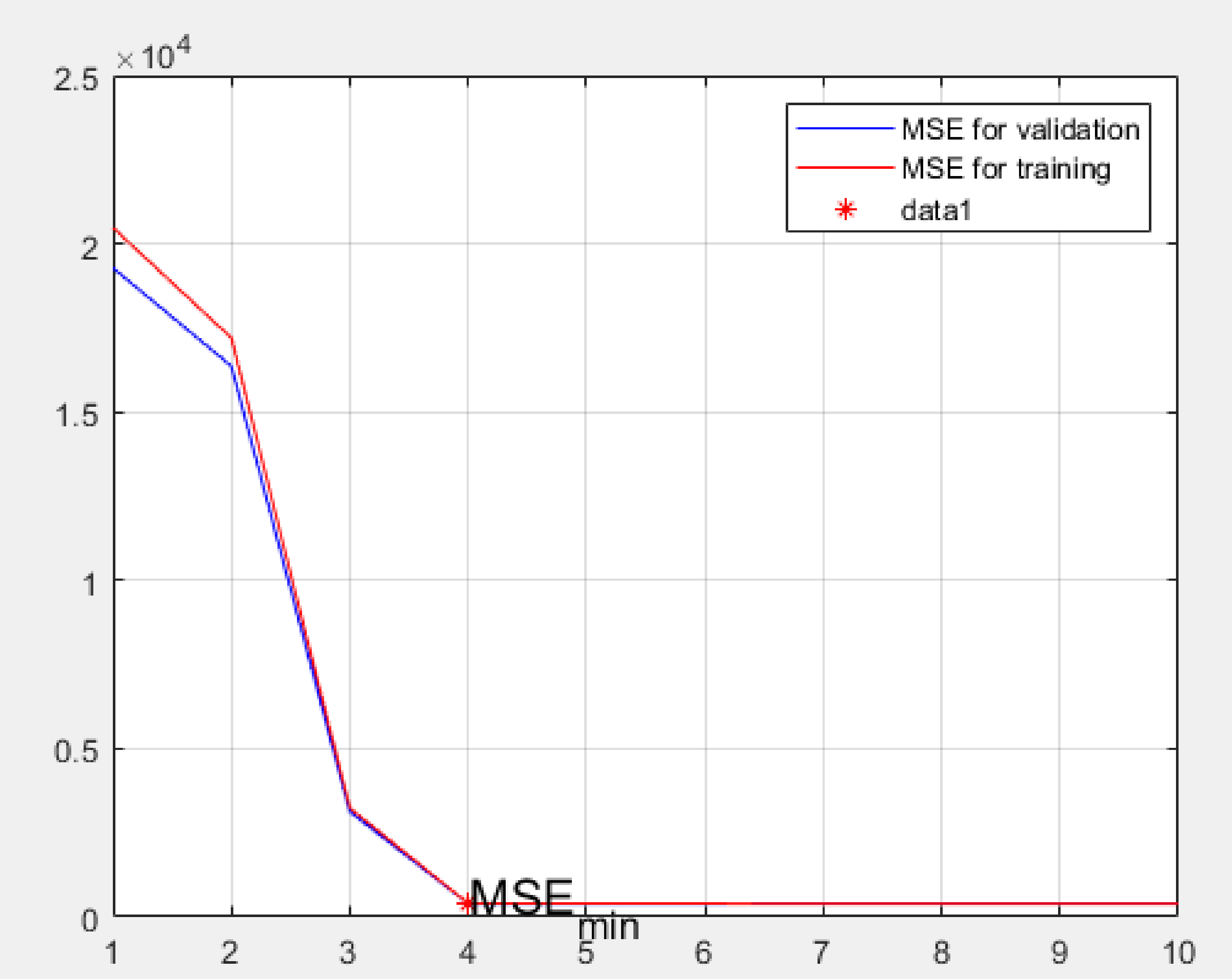
$$Degree\ 3: x_2^3 + x_1^1 x_2^2 + x_1^2 x_2^1 + x_1^3$$

Key features:

- Checking the MSE at each polynomial degree, selecting the degree with ***the lowest error,*** saving the corresponding optimal $\boldsymbol{\theta}$ and ***polynomial degree*** for further calculations.
- So by the end we can calculate ***the best fitting solution***.

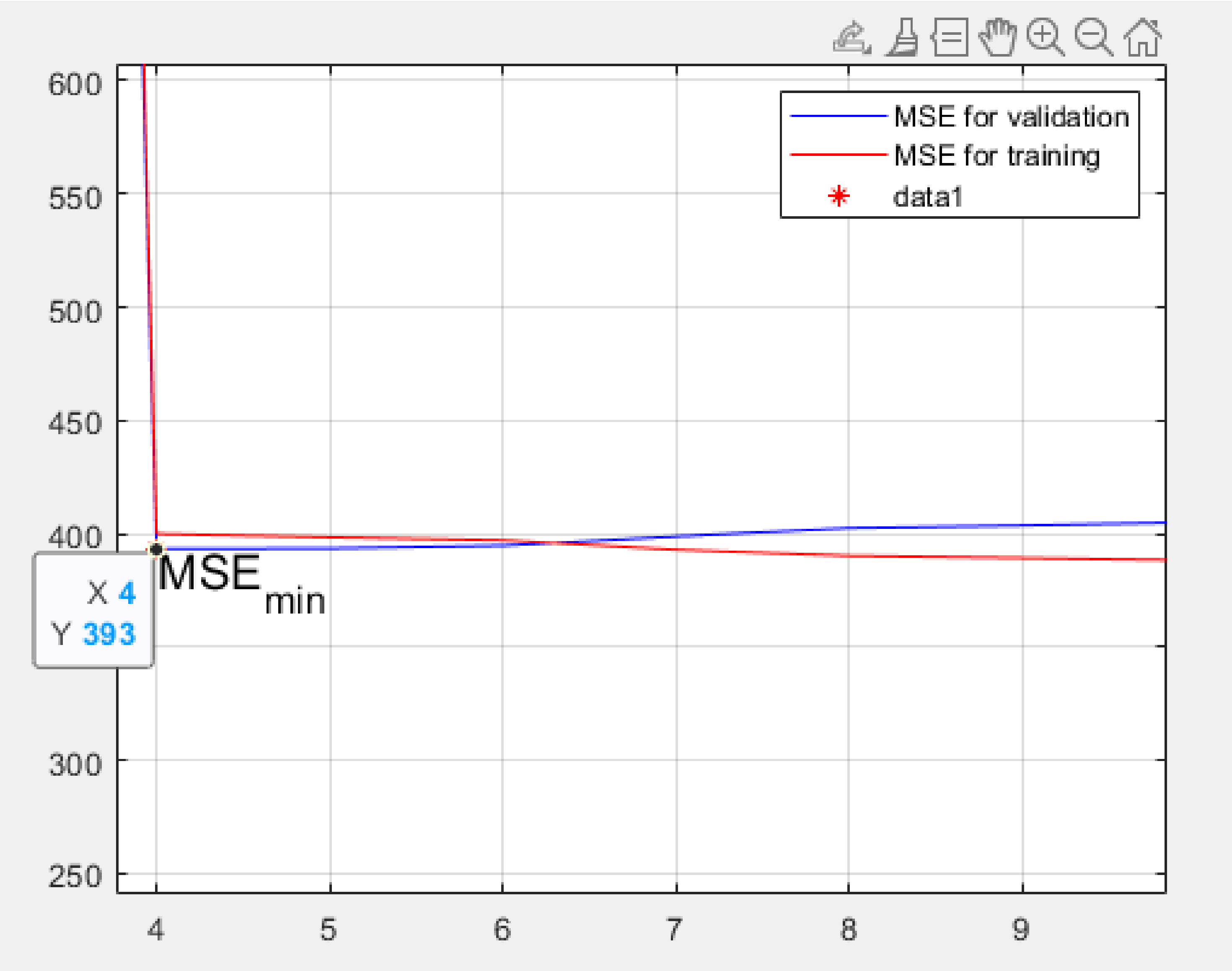


Tuning Results - MSE



MSE over time for both validation and training data

Lowest ***MSE*** from which the optimal grade of the polynomial is chosen



Overall Conclusion

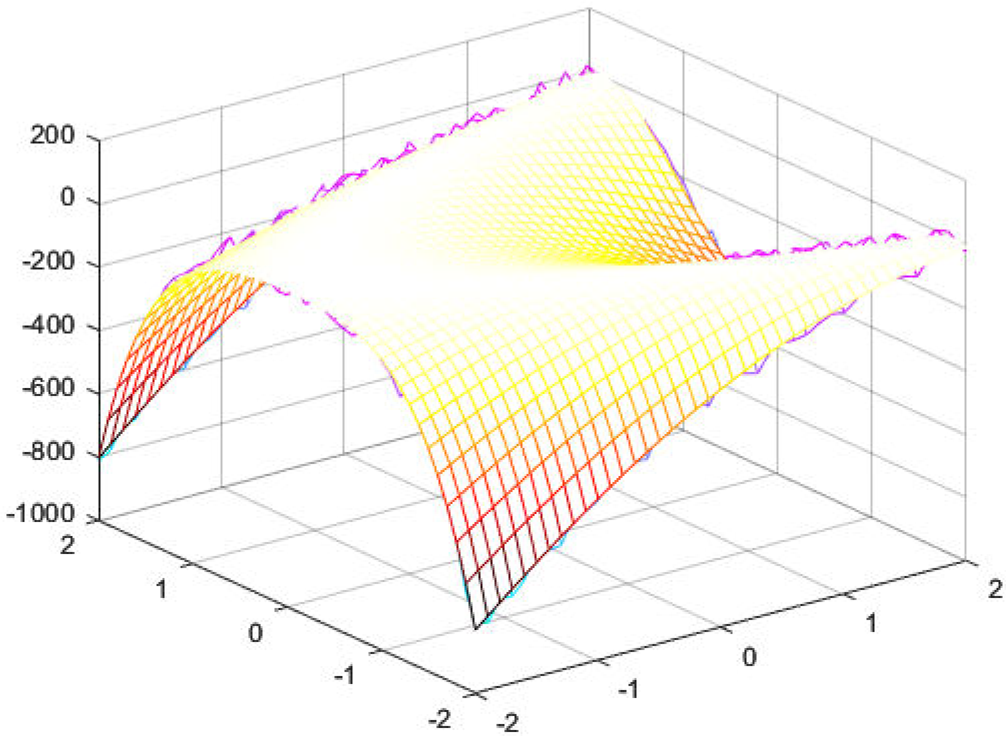
- Optimal polynomial degree minimizes MSE, balancing fit and complexity.
- Dynamic term generator enables efficient, adaptable modeling.
- Model selection validated with low MSE and strong generalization.

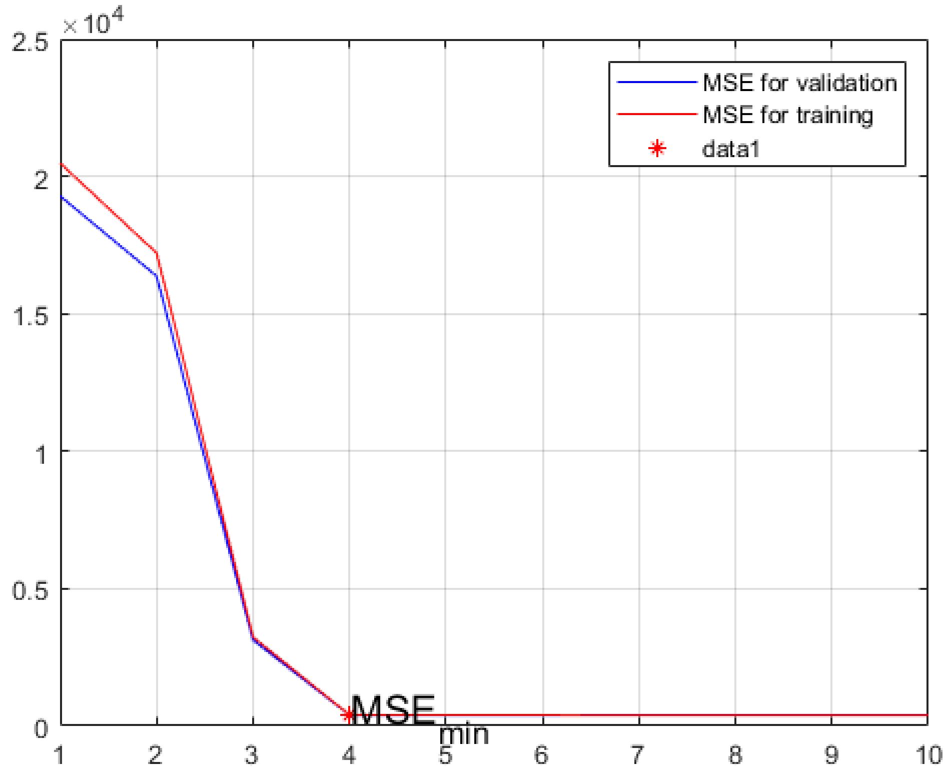
for the m order polynomial

```
load 'proj_fit_13.mat'
[X1_grid, X2_grid] = meshgrid(id.X{1, 1}, id.X{2, 1});
x1_flat = X1_grid(:);
x2_flat = X2_grid(:);
y_flat = id.Y(:);

[X1_val_grid, X2_val_grid] = meshgrid(val.X{1, 1}, val.X{2, 1});
x1_val_flat = X1_val_grid(:);
x2_val_flat = X2_val_grid(:);
y_val_flat = val.Y(:);

n = 10;
MSE_train = zeros(1, n);
MSE_value = zeros(1, n);

min_mse = inf;
best_m = 0;
theta_best = [];

for m = 1:n
    R_train = polynomialTerms(x1_flat, x2_flat, m);
    R_val = polynomialTerms(x1_val_flat, x2_val_flat, m);

    theta = R_train \ y_flat;
    yhat_train = R_train * theta;
    yhat_val = R_val * theta;

    errors_train = y_flat - yhat_train;
    MSE_train(m) = mean(errors_train.^2);

    errors_val = y_val_flat - yhat_val;
    MSE_value(m) = mean(errors_val.^2);

    if MSE_value(m) < min_mse
        theta_best = theta;
        min_mse = MSE_value(m);
        best_m = m;
    end
end

R_plot = polynomialTerms(x1_flat, x2_flat, best_m);
yhat_plot = R_plot * theta_best;

ymatrix = reshape(yhat_plot, size(id.Y));

figure;
ax1 = axes;
mesh(ax1, id.X{1, 1}, id.X{2, 1}, id.Y);
colormap(ax1, 'cool');
hold on;
```

```
ax2 = axes;
mesh(ax2, id.X{1, 1}, id.X{2, 1}, ymatrix);
colormap(ax2, 'hot');

set(ax2, 'Color', 'none');
set(ax2, 'XAxisLocation', 'top', 'YAxisLocation', 'right');
linkprop([ax1, ax2], {'XLim', 'YLim', 'ZLim', 'View'});
view(3);

figure
plot(1:n, MSE_value, 'blue'); grid
hold on
plot(1:n, MSE_train, 'red');
legend('MSE for validation','MSE for training');
plot(4,393,'rs','Marker','*');
text(4,380,'MSE_{min}','Color','k','FontSize',15);

min_mse = min(MSE_value);
disp(min_mse)

function R = polynomialTerms(x1, x2, m)
    terms = [];
    for total_degree = 0:m
        for i = 0:total_degree
            j = total_degree - i;
            term = (x1.^i) .* (x2.^j);
            terms = [terms, term];
        end
    end
    R = terms;
end
```

```
393.3541
```


200
0
-200
-400
-600
-800
-1000
2
1
0
-1
-2
-2
-1
0
1
2


$\times 10^4$
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7
8
9
10
MSE for validation
MSE for training
data1
MSE
min