Fitting an unknown function
System Identification 2023-2024

Szakacs Armand-Antonio
Petcuț Adrian-Axente
Ilea Cosmin-Ionuț

Group Id: 25

Table of contents :

Problem statement

Given a set of data where outputs are generated by an unknown, nonlinear but static function f and corrupted by noise, we must use a polynomial approximator g of configurable degree m to approximate the function f.

- It is given two data sets: one to identify the model and another to validate it.
- Each data set contains the following fields:
  1. The input X composed by two vectors: $\mathrm{X}\{1\}$ and $\mathrm{X}\{2\}$ containing n points each.
  2. The output Y as a matrix of size $n \times n$, where $Y(i, j)$ is equal to the value of f at the point ($\mathrm{X}\{1\}(i), \mathrm{X}\{2\}$(j)).

Polynomial Approximator

We were given a polynomial of approximator with a configurable degree m of the following form:

- $m = 1, \hat{g}(x) = [1, x_1, x_2] \cdot \theta$
- $m = 2, \hat{g}(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2\ ] \cdot \theta$
- $m = 3, \hat{g}(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3, x_1 x_2, x_1^2 x_2, x_1 x_2^2] \cdot \theta$

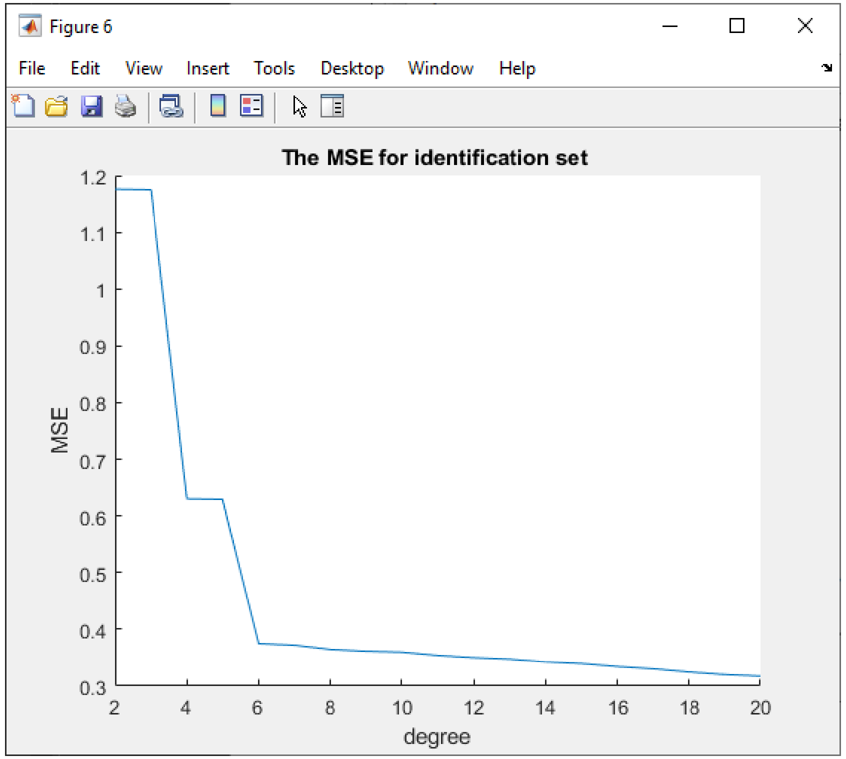
To generate the polynomial we developed the following algorithm:

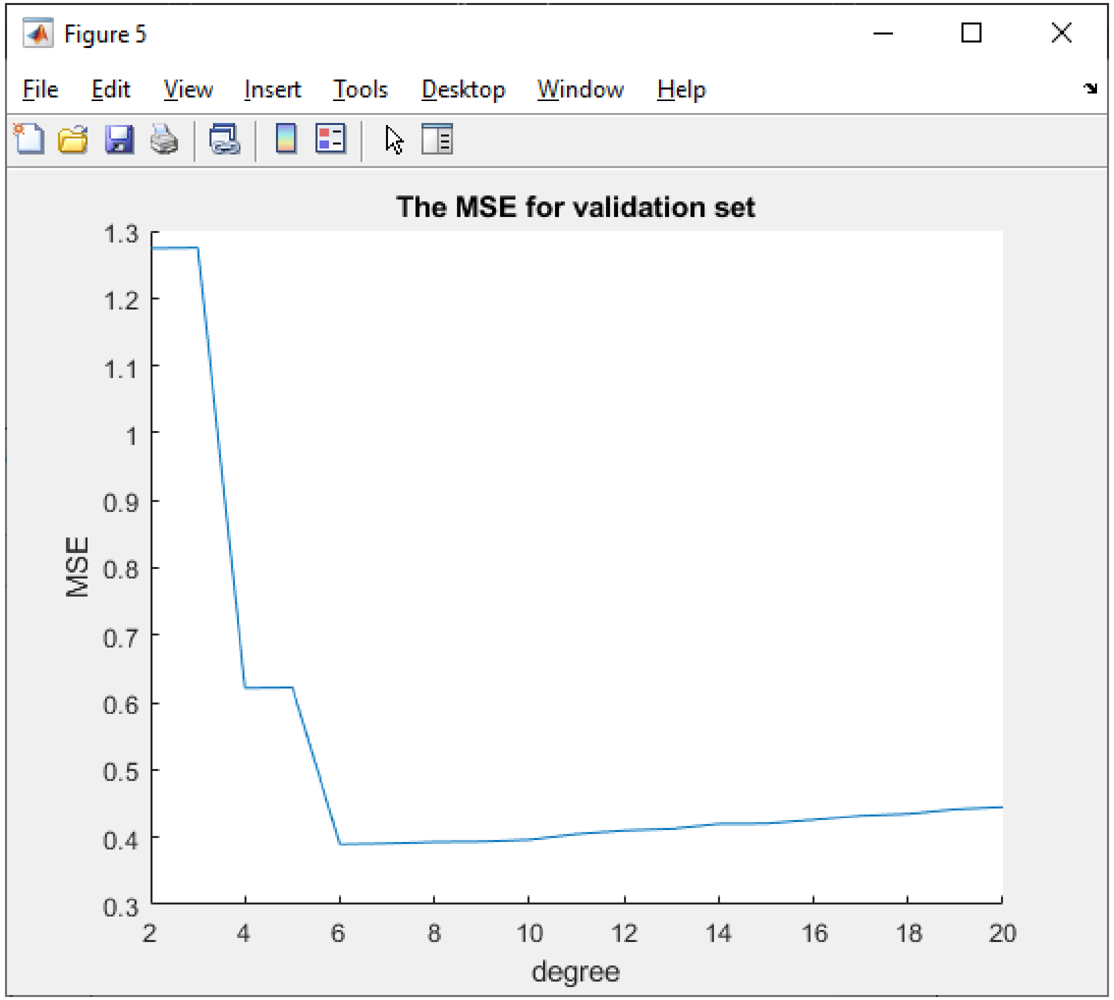
- The first element is always 1.
- Iterate over the powers of x_1 and x_2 using nested loops.
- Add $x_1^i, x_2^j, x_1^j, x_2^i$ to the vector x with the condition that $i + j \leq m$.
- Return a vector x as the output containing each term in the polynomial.

Key features

- Easy to read code
- Graphs for every type of identification and validation
- The algorithm works for any dataset given without additional user input
- We transform the matrix into a collum vector for easier computation and understanding of the code
- We automated the process of choosing the best m based on the lowest Mean Square Error

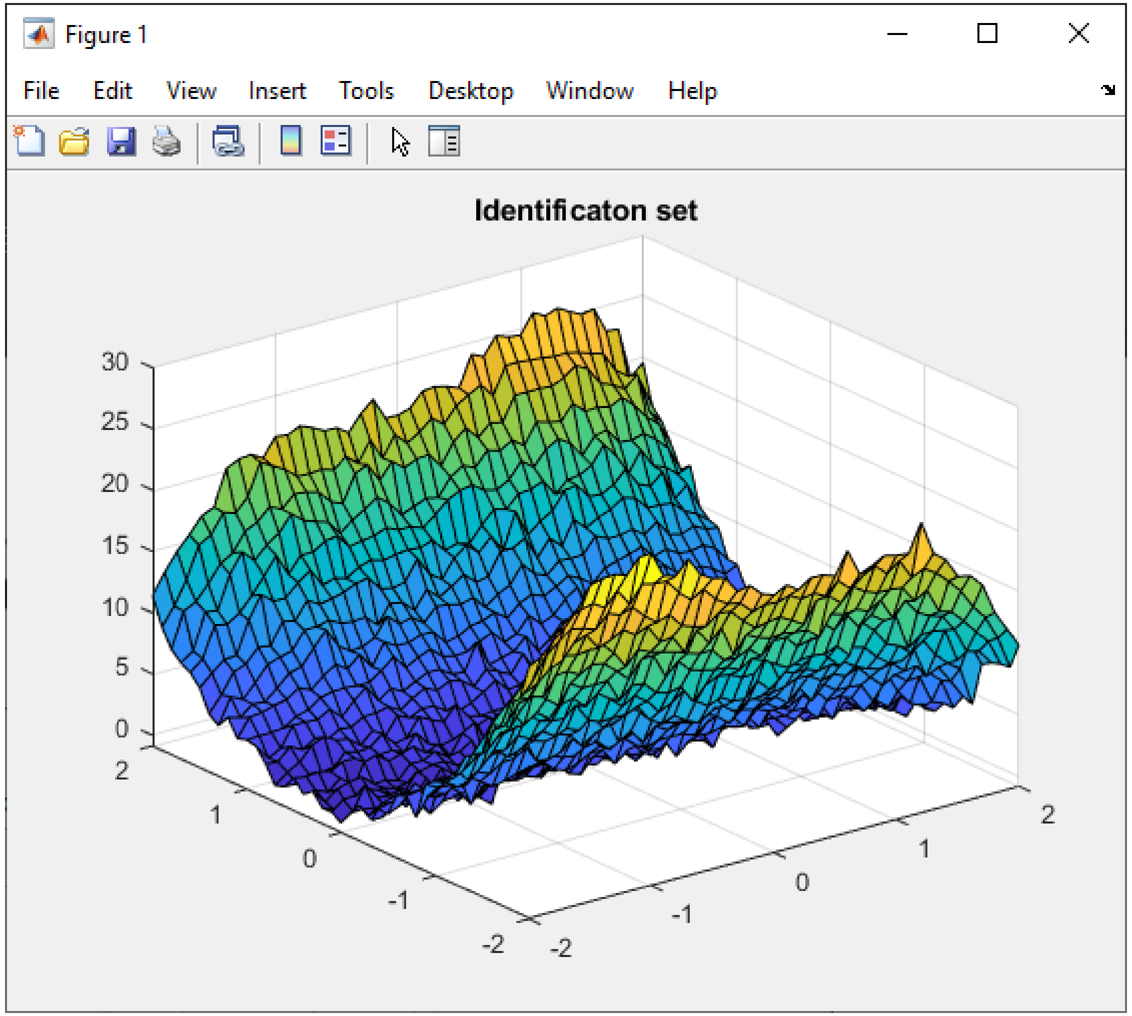
Tuning Results

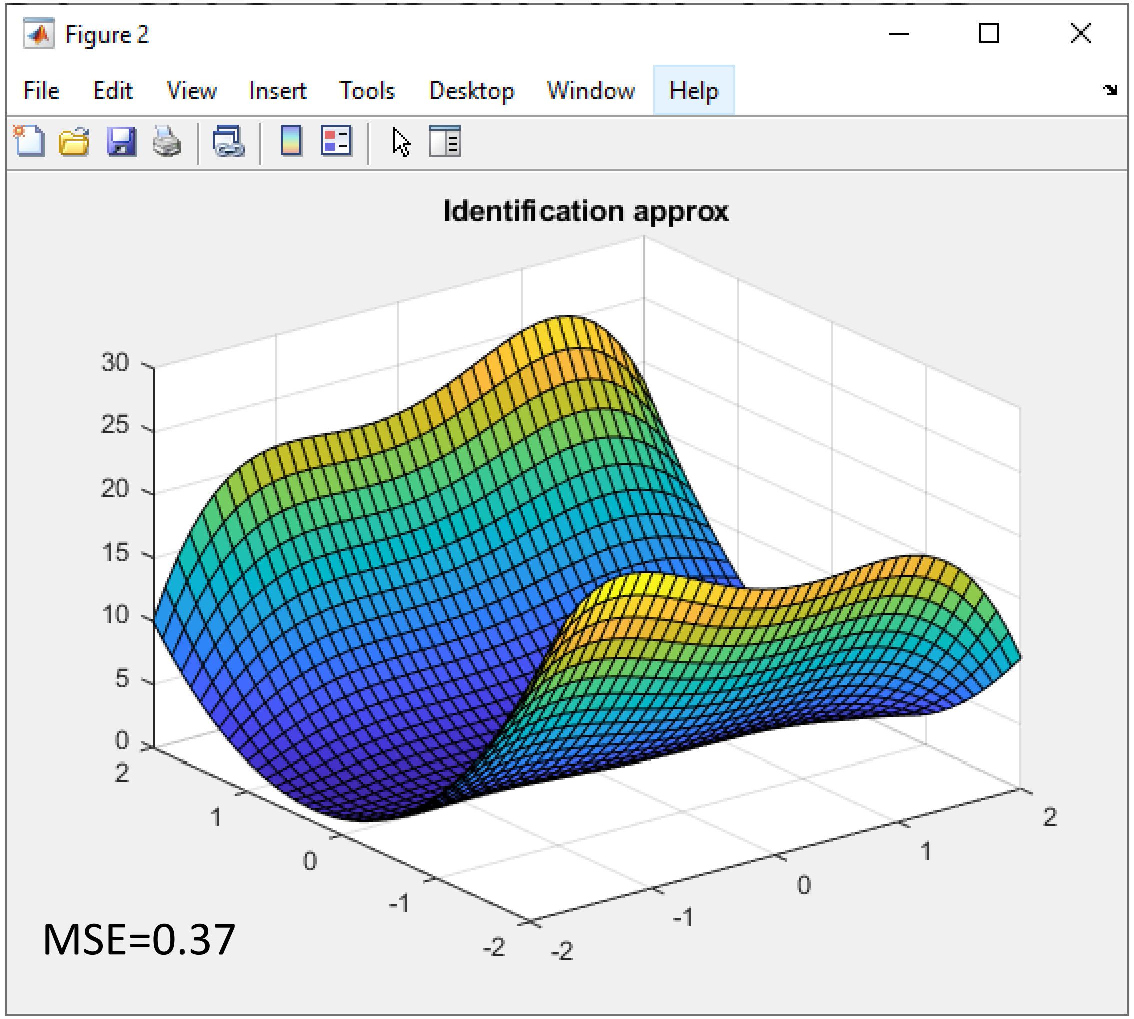




To avoid overfitting we choose 6 as the optimal value for m.

Plots for the optimal value (m=6)





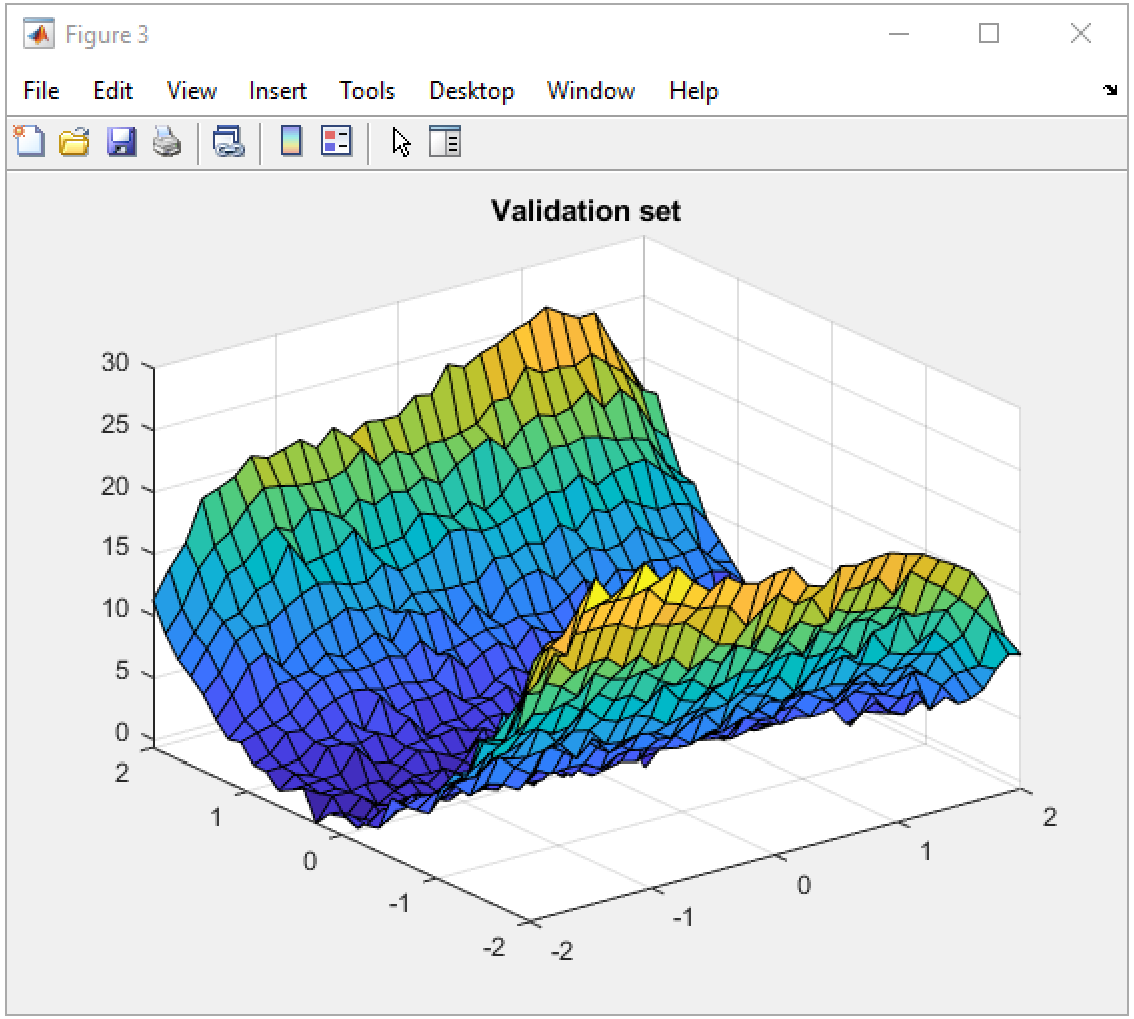


Figure 3
File Edit View Insert Tools Desktop Window Help
Validation set
30
25
20
15
10
5
0
2
1
0
-1
-2
-2
-1
0
1
2

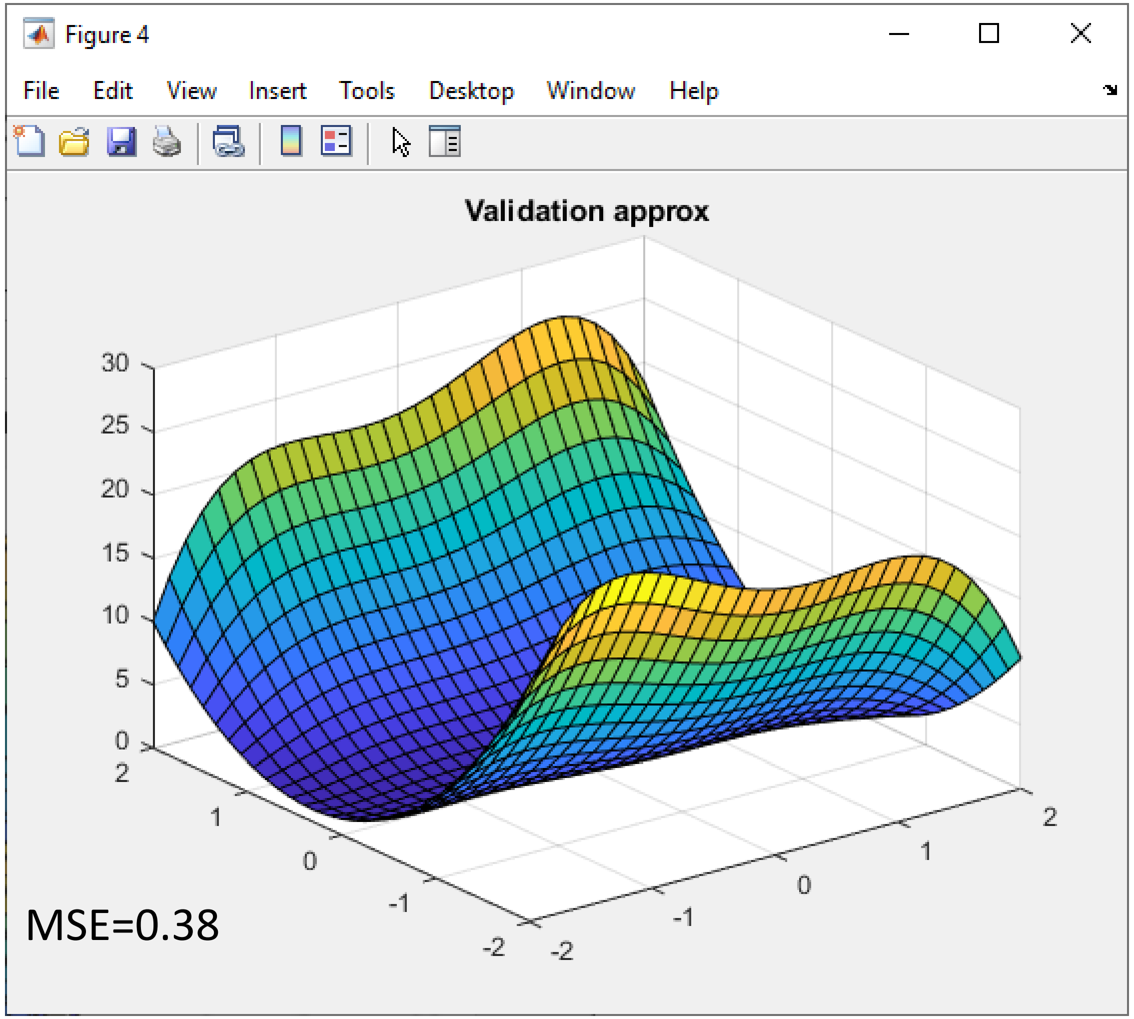


Figure 4
File Edit View Insert Tools Desktop Window Help
Validation approx
30
25
20
15
10
5
0
2
1
0
-1
-2
-2
-1
0
1
2
MSE=0.38

Conclusions

- Linear regression is a very strong and simple to understand method for identifying models
- m can be adjusted to get an optimal model
- We developed a strong solution to the problem

Appendix

```
%MSE vectors
MSE_v = [];
MSE_i = [];

%variables we use to find the degree for minium MSE
MSE_previous = 100;
degree_min = 0;

%from output matrix to collumn vector
newY_id = matToVec(id.Y);
newY_val = matToVec(val.Y);
```

```
for degree=2:20
%COMPUTE THETA
phi = compPhi(id.X, degree);
theta = phi \ newY_id;
%approximation for identification set
approxId = phi*theta;
%approximation for validation set
phi = compPhi(val.X, degree);
approxVal = phi*theta;
%compute MSEs for both sets
MSE_id = compMSE(approxId, newY_id);
MSE_val = compMSE(approxVal, newY_val);
MSE_v = [MSE_v MSE_val];
MSE_i = [MSE_i MSE_id];
%finding the degree with the minimum mse
if MSE_val < MSE_previous
MSE_previous = MSE_val;
degree_min = degree;
end
end
```

```
degree = degree_min;
%COMPUTE THETA
phi = compPhi(id.X, degree);
theta = phi \ newY_id;
%approximation for identification set
approxId = phi*theta;
%approximation for validation set
phi = compPhi(val.X, degree);
approxVal = phi*theta;
%approximation from vector to matrix
approxIdMat = vecToMat(approxId, id.dims(1));
approxValMat = vecToMat(approxVal, val.dims(1));
%plot the approximation vs real values
f3 = figure;
movegui(f3, 'north');
surf(id.X{1}, id.X{2}, id.Y);
title("Identificaton set")
f4 = figure;
movegui(f4, 'south');
surf(id.X{1}, id.X{2}, approxIdMat);
title("Identification approx")
f5 = figure;
movegui(f5, 'northeast');
surf(val.X{1}, val.X{2}, val.Y);
title("Validation set")
f6 = figure;
movegui(f6, 'southeast');
surf(val.X{1}, val.X{2}, approxValMat);
title("Validation approx")
%plot the MSE for both sets;
f1 = figure;
hold on;
movegui(f1, 'northwest');
plot(2:20, MSE_v);
xlabel('degree');
ylabel('MSE');
title("The MSE for validation set");
hold off;
%plot the MSE;
f2 = figure;
hold on;
movegui(f2, 'southwest');
plot(2:20, MSE_i);
xlabel('degree');
ylabel('MSE');
title("The MSE for identification set");
hold off;
```

Functions:

Polynomial approximator generator

```
function [x] = polygen(polydegree, x1, x2)
i = 1;
x = [1];
while i < polydegree
j = 0;
while j <= i && i+j <= polydegree
x = [x x1.^i .* x2.^j];
if i ~= j
x = [x x1.^j .* x2.^i];
end
j = j + 1;
end
i = i + 1;
end
j = 0;
x = [x x1.^i .* x2.^j];
x = [x x1.^j .* x2.^i];
end
```

Creating MSE

```
function [MSE] = compMSE(approx, trueVal)
MSE = 0;
for i=1:length(approx)
MSE = MSE + (approx(i) - trueVal(i))^2;
end
MSE = MSE / length(approx);
end
```

Creating Phi

```
function [phi] =
compPhi(cellArray, degree)
phi = [];
for i = 1:length(cellArray{1})
for j = 1:length(cellArray{2})
phi = [phi; polygen(degree,
cellArray{1}(i), cellArray{2}(j))];
end
end
end
```

Matrix to vector

```
function [vec] =
matToVec(matrix)
vec = [];
for i =
1:length(matrix)
vec = [vec
matrix(i,:)];
end
vec = vec';
end
```

Vector to matrix

```
function [matrix] = vecToMat(vector,
dimension)
matrix = [];
for i = 1:length(vector)
if mod(i, dimension) == 0
matrix = [matrix; vector(i-
dimension+1:i)'];
end
end
```