Verkfræðilegar Bestunaraðferðir

Day 11 - Group 2

T-423-ENOP

Arnar Gylfi Haraldsson
arnarh23@ru.is

Hafþór Árni Hermannsson
hafthorh20@ru.is

Ragnheiður Gná Gústafsdóttir
ragnheidurg@ru.is

May 15, 2024

Exercise 1

Description

Consider a data pairs provided in a file exercise_1_data.mat, and shown in a plot: Construct a linear regression model of this data using basis functions of the form: 1, sin(x), cos(x), sin(2x), cos(2x), ..., sin(nx), cos(nx). Plot both the input data and the regression function for different values of n. Plot approximation error versus n. What is the smallest n ensuring sufficient accuracy of the regression model in this case?

Functionality

The code has the following steps and attributes:

- It starts by loading the contents of exercise_1_data.mat into a vector called y, initializes the x vector with one column of length Y as ones and a integer vector nv that spans from 1 to n .
- For each value of n a column of sin(nx) and a column of cos(nx) is added onto the X matrix.
- λ is calculated with equation (1).
- Y_{pred} is calculated with equation (2).
- The data points and predicted curve is plotted together.
- The approximation error vs n is calculated and then plotted.

$$\lambda = (X^T X)^{-1} X^T y \tag{1}$$

$$Y_{pred} = X * \lambda \tag{2}$$

We chose to plot within the for loop for each n to be able to see the progress the regression made with each n.

Solution

The regression was done for n = 2, 5, 8 and 20. The results of these regression models are displayed in figures 1 to 4. The error is displayed in figure 5, on that it can be observed that between n=37 to n=40 the error drops significantly. When using the cursor on the matlab figure the biggest difference can be seen between n=37 and n=38, where is drops from 0.0382 to $2.4988e-11$. Therefore the smallest n ensuring sufficient accuracy of the regression model in this case is n = 38.

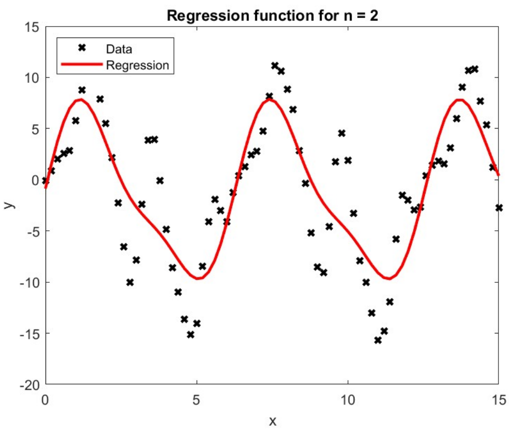

Figure 1: n = 2

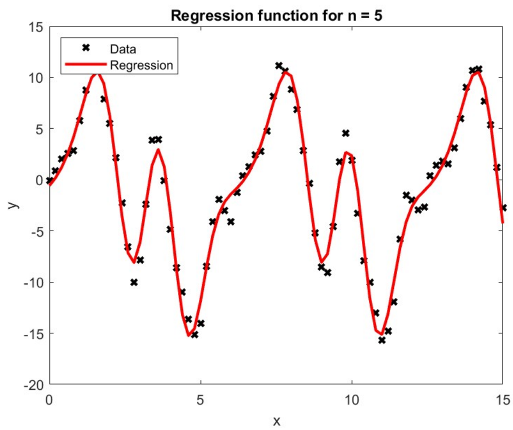

Figure 2: n = 5

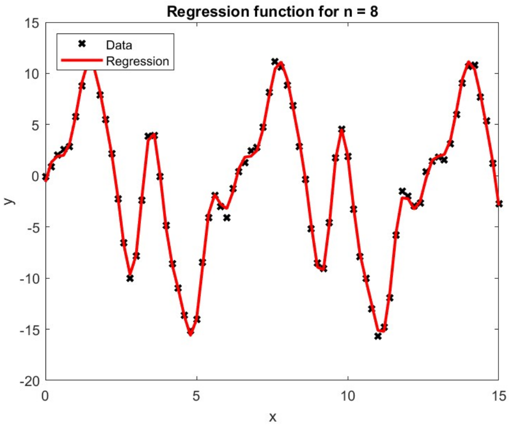

Figure 3: n = 8

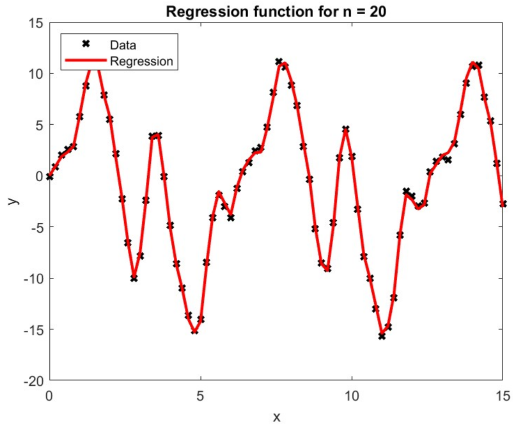

Figure 4: n = 20

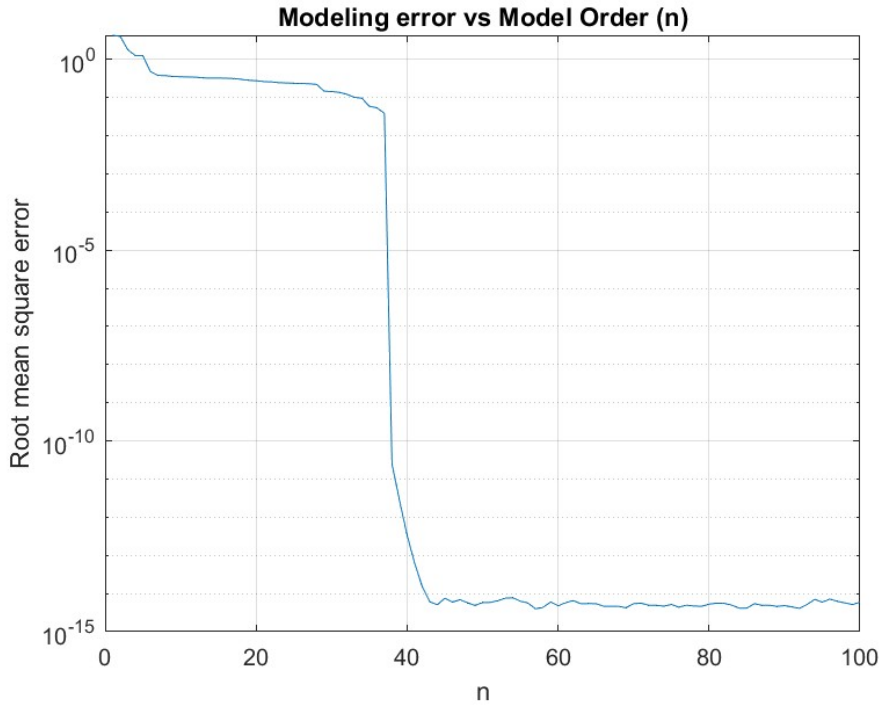

Figure 5: The error of the regression model on a logarithmic scale for better visualization.

Exercise 2

Description

The following dataset is provided:

The objective is to:

1. Plot the data and think of possible nonlinear regression model that has no more than five parameters.
2. Implement the model and obtain its parameters using lsqnonlin routine from Matlab Optimization Toolbox.
3. Plot both the input data and the regression function.

Functionality

After plotting up the data the first time it was decided that a combination of $cos(x)$ and $exp(-x)$ would represent the data the best due to the visual shape of it. The objective function for the lsqnonlin routine was therefore:

$$Obj\_func1(param, x) = param(1) * cos(param(2) * x) + param(3) * exp(-param(4) * x) \tag{3}$$

Then to find a better fit three more terms were added, $1, x$ and x^2. Then the objective function for the lsqnonlin routine became:

$$\begin{aligned} Obj\_func2(param, x) = param(1) + param(2) * x + param(3) * x^2 \\ + param(4) * cos(param(5) * x) + param(6) * exp(-param(7) * x) \end{aligned} \tag{4}$$

Result

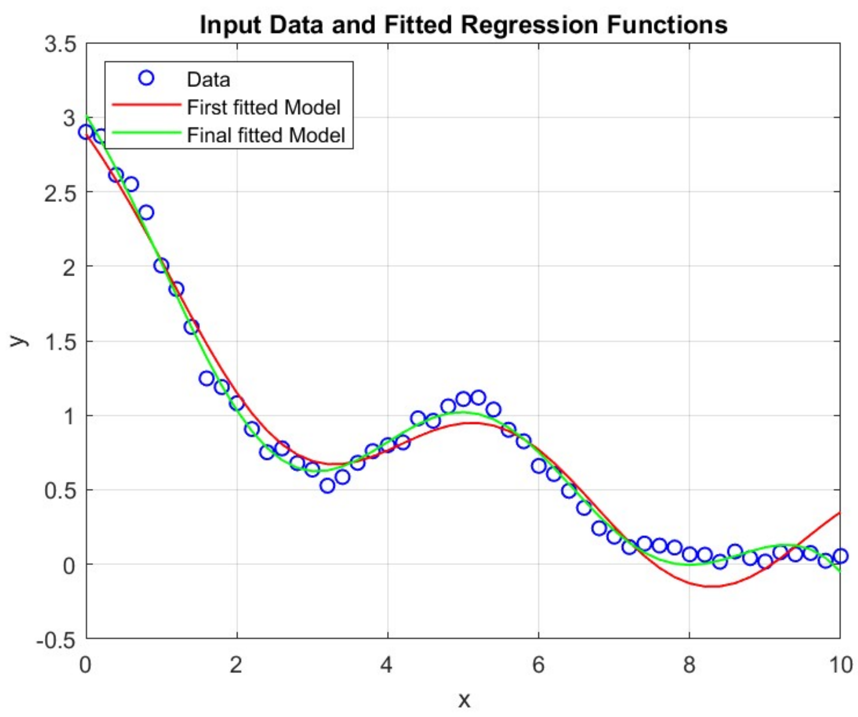

Figure 6: Data points and both regression functions developed.

Exercise 3

Description

Create a radial basis function surrogate model for function exercise_3_function. The domain is [-3,3][-3,3]. Use uniform-grid base set with N = 9, 25, 49, 100, and 169 points. Use Gaussian basis functions with c = 1. Make the surface plot of f and the surface plot of the RBF model; indicate base points as black dots. Calculate the approximation error for each N using a separate set of 100 test points generated using LHS.

Functionality

This script implements a Radial Basis Function (RBF) surrogate model for a given function `exercise_3_function` within the domain $[-3,3]\times[-3,3]$. The main operations performed by the script are as follows:

1. **Parameter Initialization**: Initialize, grid size, N and Gaussian basis function parameter c.
2. **Original Function Evaluation and Plotting**:
   - Generate a grid and evaluate the original function on this grid.
   - Plot the original function as a surface plot.
3. **Test Points Generation Using Latin Hypercube Sampling (LHS)**:
   - Generate 100 test points within the domain using LHS.
   - Evaluate the function at these test points.
4. **RBF Surrogate Model Construction and Evaluation**:
   - Loop through each N value:
     - Generate uniform grid base points.
     - Compute the Gaussian RBF matrix.
     - Evaluate the function at the base points.
     - Solve for the RBF model weights.
     - Predict function values at the base points.
     - Calculate the approximation error using LHS test points.
     - Plot the RBF surrogate model.
5. **Output**:
   - Surface plots of the original function and RBF surrogate models.
   - Approximation errors for each N value displayed in the console.

Result

Table 1: LHS approximation error

N	Approximation Error
9	1.2639
25	0.20266
49	0.02651
100	0.0038961
169	0.00023315

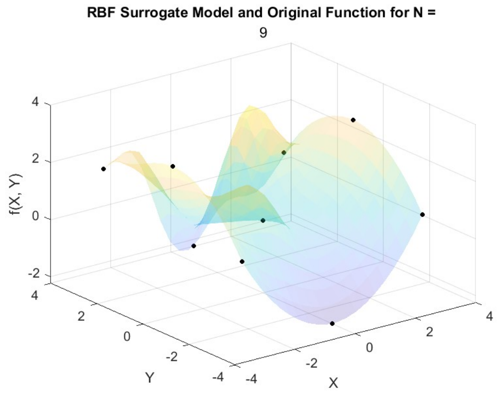

Figure 7: N = 9

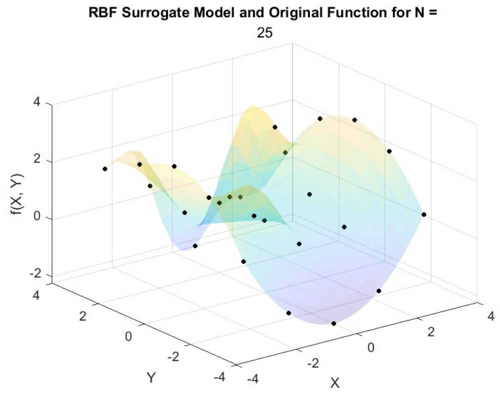

Figure 8: N = 25

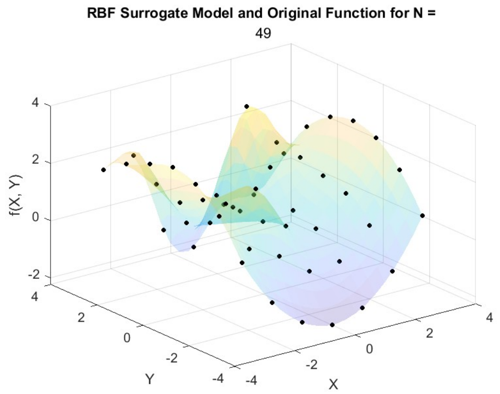

Figure 9: N = 49

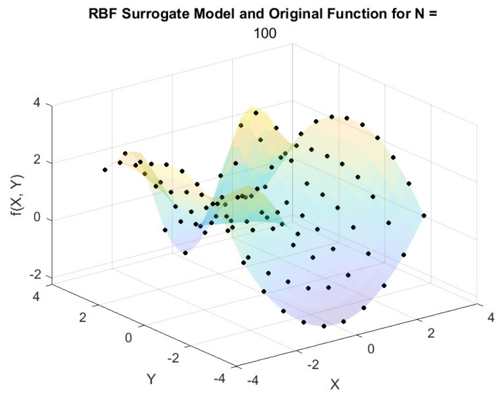

Figure 10: N = 100

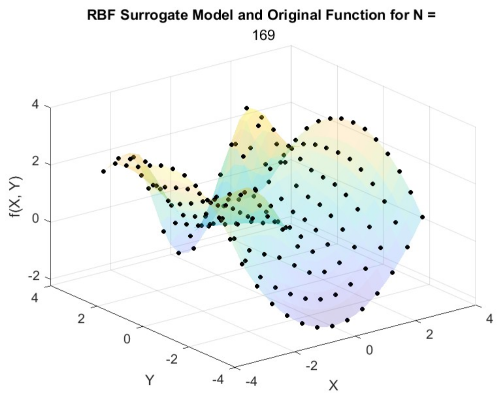

Figure 11: N = 169

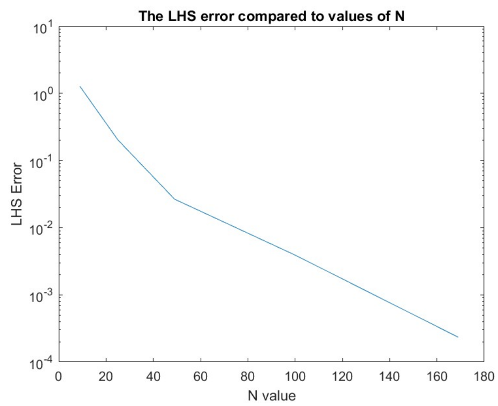

Figure 12: LHS error on a logarithmic scale compared to values of N