Dragos Pop (2618006)

Assignment 3 – Data Fitting

Concise method description

The function implemented in Matlab fits a linear combination of exponential functions to data. First, it minimizes the residue function R(λ) (computed according to the least square method in a secondary Matlab function) to find the optimal exponents λ. Then, it derives the optimal constants C_i by solving the system of linear equations [exp(λ*x) * C = y], obtained from the given function. Finally, it uses the estimated y values to output the total residue of the optimal fit and plots the fit over the data.

Functions syntax

Given the required name of the Matlab file, the primary function can be called by *Assignment_3_Pop()*, specifying the right arguments. These are:

- *data* = an (mx2) matrix, where x values are in the first column and y values in the second

* For this report, the functions were tested on the data sets in the file *expo-examples.mat*

- *initial_guesses* = the vector of initial guesses for the exponents (i.e. lambda values)

Accordingly, calling *Assignment_3_Pop(data1, [1, 1.2, 0, 1.5])* returns a vector lambda_opt = [-2.0811 -0.1576 0.0167 -0.4783], which stores the calculated optimal exponents lambda, a vector constant_opt = $[2.8977 \quad 0.7602 \quad -0.0945 \quad 1.4778]^T$, that has the computed optimal constants C_i values, and E = 39.0849, the residue of the optimal fit. Moreover, the function plots the fit over the data, as one can see in the picture below.

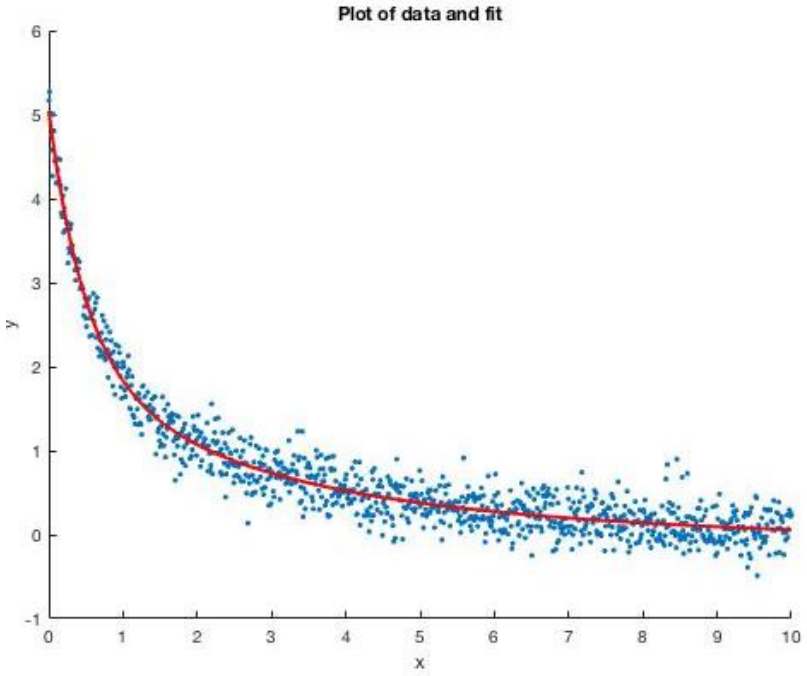

Figure 1 – Plot of the data1 and its fit generated with initial lambda guesses [1, 0.2, 1.2, 2]

The secondary function (the one that, given the initial lambda guesses, computes the corresponding residue R(λ) using least square method) is called as *residue()* and has the same arguments as *Assignment_3_Pop()*, namely *data* and *initial_guesses*. Consequently, calling *residue(data1, [1, 1.2, 0, 1.5])* generates E = 574.4634 ,the residue given the initial guesses.

Dragos Pop (2618006)

Restrictions

One limitation of the program is given by the Matlab built-in function *fminsearch()*, used to minimize the sum of squared residuals, which can get stuck in the local minimum. Consequently, the performance of the program is highly dependent on the initial guesses.

Program design

In the beginning, the *residue()* function is built as follows: first, the x and y values are extracted splitting the data, then the y estimates are calculated for the vector of initial lambdas converting the function $f(x) = C_1e^{\lambda_1 x} + C_2e^{\lambda_2 x} + \ldots + C_ne^{\lambda_n x}$ to matrix form. More exactly, the exponents are simply computed by multiplying the initial guesses with the x values. Afterward, the constants C are derived solving the system of linear equations [exp(λ*x) * C = y] with the help of Matlab function *mldivide()*, and eventually, the two terms are multiplied to form the estimates. Once found, they are subtracted from the real values of y, then squared and summed, according to the least square equation, to get the total residue.

In the second phase of the program, the main function is designed, starting in a similar fashion by splitting the data in the x and y vectors. Next, a function with initial guesses vector as the only parameter is derived from the previously built *residue()* method. Then, this new function R(λ) is minimized, returning the optimal lambda values. Subsequently, the optimal constants can be found by solving the same system of linear equation with optimal lambdas instead of the initial ones. From this point, the y estimates and the total residue are computed similarly as for *residue(),* however, the optimal vectors are used instead. Finally, the program generates a plot representing the fit over the data.

Figure interpretation

The illustration on the right was generated after calling *Assignment_3_Pop(data3, [1, 2])*. As one can easily observe, the red line fits the data well, suggesting that the function performed successfully. In order to better show the improvement of the fitting, the residue from the initial step (calculated using *residue(data3, [1, 2]))* was compared with the residue after optimization. Consequently, the first was evaluated at 63.5544, while the second was 30.3017, showing a significant improvement.

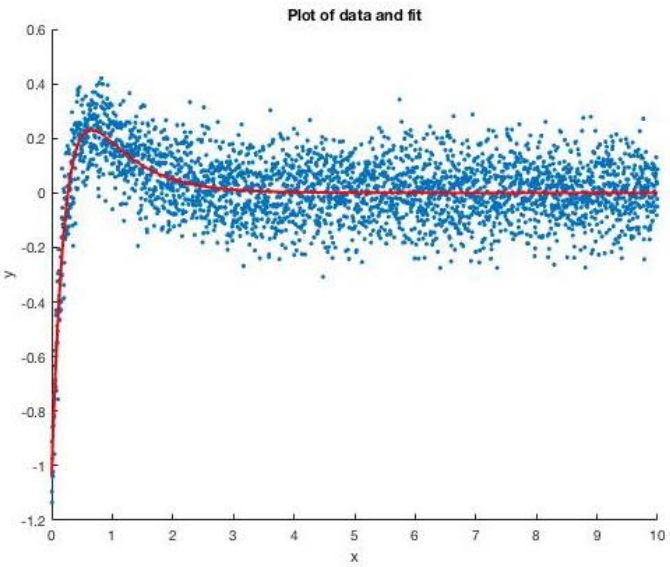

Figure 2 – Plot of the data3 and its fit generated with initial lambda guesses [1, 2]