Function Fitting

As discussed in the methodology section. The implementation of this project is capable of supporting 3 types of analyses. Despite the fact that the three of them perform differently and return different values, they all have a goal in common; to describe a set of observed data as a mathematical expression or function. The previously mentioned description will be called on this paper as “function fitting”. We will later notice that each function procedure has a specific evaluation procedure, which will be explained in the following subchapters. Also mention the previous function of all cases that helps filtering similar formulas.

Polynomial Fitting

At this point of the implementation we assume that the data points to be analyzed are actually coming from a polynomial function or procedure. As a polynomial function can be expressed with different coefficients and degrees. The purpose of the polynomial fitting algorithm is to actually attempt to fit the observed data points from a trace to a set of polynomial functions from degree 0 to degree *n*; where n is our upper bound or limit of degree. Where the desired function will be the one with the highest degree and least error among all attempted fitted functions. All details will be discussed on the Algorithm section of this chapter.

Algorithm

The library utilized in this implementation was the polynomialCurvFitter class from the math3 apache commons. It fits a set of observed data to a polynomial function of degree *x* (previously specified as a parameter) and returns the coefficients for the desired polynomial function of degree *x*. The procedure of the analysis is explained with the following algorithm:

----------- Polynomial algorithm ------------

The analysis always begins by fitting the information to a polynomial function of degree 0. The observed data is fitted to a polynomial function by first creating PolynomialCurvFitter object with the create method, with desired degree as a parameter (line 4). The coefficients parameters that best fit the previously specified degree are returned by the fit method from line 5. A temporal variable called *functionCandidate* is used in order to identify the maximum degree to be fitted and to perform the evaluation of the function. The maximum degree is reached when the last returned coefficient of the fit function is either 0 or a very close value. (line 6). Otherwise the function proceeds to be evaluated (line 7, Polynomial-function-evaluation algorithm).

For simplicity and order, let us take a look at the polynomial evaluation (figure x). We begin by first retrieving the first *x* and *y* data points from our observed data. Which represents the original information from the trace (line 3-4). Then we proceed to apply the current function candidate with the *x* parameter of the original trace and then compare the result with the exact value of the original trace, which is *y* (line 6-9). Afterwards, the error is calculated by retrieving the absolute value of the difference between the original value from the trace *y* and the value obtained from the candidate function with *x* as a parameter. Once the error is returned to the main polynomial algorithm, we then store the degree with least error, as seen from line 8 to 11. After the maximum degree is reached, the *chosenDegree* or the degree with the least error is then compared with the polynomial function that was calculated with the previous buffer information, if any; the previous polynomial function is kept when its error is below the established threshold (line 17 to 27). Also notice that the degree of an existing function is calculated by the number of its coefficients minus 1.

Polynomial function preparation/construction

Note: move PreparePolynomialFunction() from the code/pseudo algorithm to another procedure and explain how the obtained coefficients are put together into a polynomial function. We might change this part as we don’t have a recursive representation yet.

Exponential Fitting

Algorithm

Exponential function preparation/construction

Harmonic Oscillation Fitting

Algorithm

Harmonic function preparation/construction