Programming (ERIM)

6. Exercise

Deadline for submission: 2014-12-14 23:59 CET

Introduction

In this exercise you will implement an algorithm that finds the root (intersection with the x axis) of functions. Because each equation can be reduced to a 'find the root of a function' problem, this algorithm can be used to numerically solve equations that are analytically unsolvable. The method that we will implement is the Newton's method (see `http://en.wikipedia.org/wiki/Newton%27s_method`).

The basic idea of the method is that one starts with an initial guess (e.g., x_0) that is likely to be close to the true root of the function. Using this starting point x_0, the next point x_1 is the root of the tangent line at x_0. At this point you might want to look at the animation at the wikipedia page to get an idea of how the method works.

More formally, if $f(x)$ is a real differentiable function, then the relationship between a current point x_n and the next point x_{n+1} can be expressed in terms of the derivative of the function:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{1}$$

In this equation, x_{n+1} is the root of the line that is tangent with f at point x_0.

Using Equation (1), we can compute a series of solutions that approximate the true root of the function. The precision of the solution can be determined by setting a *threshold* for the algorithm. This threshold determines how much a solution needs to change from one iteration to another in order for the algorithm to continue. This can be either applied to the roots (the x values) or the corresponding y values (the values of $f(x_n)$. For example, if the algorithm reaches iteration n in which $|f(x_n)|$ is smaller than 0.0000001, one can decide to stop iterating.

Exercise

1. Implement the Newton's method in a test-first manner. The implementation should be well documented and easily applicable to other problems. The implementation of the method should have the following inputs:

   - A single argument function. This function takes one argument as input and returns exactly one value. For example, $f(x) = x^2 - 777$ is a valid example of such a function. The part where the logic for the Newton's Method is implemented should be independent of the chosen function.
   - A threshold δ for the root value. This value indicates when the iterations should stop. For example, if $\delta = 0.01$, then the algorithm should stop after it has found a solution for which $|f(x)| < 0.01$.
   - A threshold N that defines the maximum number of iterations. This means that the algorithm should stop when $|f(x_n)| < \delta$ or $n > N$.
   - The implementation should return the best solution x^*, i.e., for which $f(x^*)$ is the smallest, the value $f(x^*)$, and the number of iterations it took to complete.

   You can use an approximation for the implementation of the derivative. In other words, you can use the following equation:

   $$f'(x) = \frac{f(x+d) - f(x)}{d} \tag{2}$$

   where d is a small constant (e.g. $d = 1 \times 10^{-5}$), that can be passed as a parameter.

2. Add another feature to the implementation (remember to do this in a test-first manner!): the possibility to pass a parameter p within the range $[0, \infty]$. If p is not equal to zero, the implementation needs to print information on the screen about each iteration. The value of p then represents the number of decimals that should be used for the printing fractional numbers. A table like the following should be printed if $p > 0$:

```
iter.    x     f(x)
0        ..    ..
1        ..    ..
```

   The values in the columns `x` and `f(x)` should be printed with p decimals.

3. Using the above implementation, we are going to analyse the behaviour of Newton's Method by plotting the iteration results. Create a function which creates a plot of the different $f(x)$ values for each iteration. You have to modify the previous implementation so that it also returns the 'per iteration information' - currently it returns only the final three values (the best solution x^*, the value $f(x^*)$, and the number of iterations). It is up to you how you return this information. The plot should show on the x axis the iteration number and on the y axis the $f(x)$ value. Make a script that visualizes the process for the function $f(x) = x^2(x - 1000) + 1$.