Programmeren (FEB22012)

Exercise 2

Deadline for submission: 2011-09-04 23:59 CET

Introduction to Newton's method

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 will be implemented is called Newton's method. In this section, we will briefly discuss the method and its limitations.

Basics

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 in [1] to get a basic idea on 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 a point where y is smaller than 0.0000001, one can decide to stop iterating.

Exercise

1. Without writing any Matlab code, manually apply Newton's method for the problem $x^2 = 777$. Start with an initial guess $x_0 = 25$. Create a table in a text file where you show the iteration number, x, $f(x)$, and the error, i.e., the difference between the computed value and the true value (in this case $f(x) - 777$). So the first line will look something like:

   ```
   iter.    x     f(x)    error
   0        25    625     -152
   ```

   You need to add three more iterations to this table. You may use a calculator to do the computations.

2. Implement Newton's method in Matlab. The implementation should be well documented and easily applicable to other problems. The implementation of the method that starts Newton's method should meet these requirements:

   - Any 'normalized form' function can be passed. This function takes one argument as input and returns exactly one value. For example, for $x^2 = 777$ one would pass the function $f(x) = x^2 - 777$. The part where the logic for Newton's Method is implemented should be independent of the chosen function. You can pass a function to another function by preceding it with an 'at sign', e.g., `@funcName` if the function is called `funcName`.
   - A threshold δ for the root value can be passed. This value indicates when the iterations should stop. For example, if $\delta = 0.01$, then the algorithm should stop after it found a solution for which $f(x) < 0.01$.
   - A threshold N that defines the maximum number of iterations can be passed. This means that the algorithm should stop when $f(x_n) < \delta$ or $n > N$.
   - The implementation should return a `struct` that contains 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. Look for `struct` in the Matlab documentation to see example uses and explanation.

   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 relatively small (e.g., $d = 1 \times 10^{-10}$).

3. Now you are going to add another feature to the implementation. Add the possibility to pass a parameter p with 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 of 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. The printing of the table in this way can be achieved using the `fprintf` function. Read the documentation in Matlab for more information on how to space the columns using equal distances and print values with a certain amount of decimals.

4. 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 such that it also returns the 'per iteration information', currently it returns a struct containing 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. A few tips:

   - you can use the function `plot`, you can make the point size larger by using `plot(..., '.k','MarkerSize',15)` as option arguments;
   - look at the `size` function in Matlab in order to determine the size of a matrix and/or vector;
   - with `1:10` you generate a vector that contains integers from 1 to 10 (inclusive).

References

[1] Wikipedia. Animation of newton's method. online. `http://upload.wikimedia.org/wikipedia/commons/e/e0/NewtonIteration_Ani.gif`.