H212 CS Spring 2014
Programming Assignment: Newton-Raphson Method
Indiana University
Bloomington, IN

Mehmet M. Dalkilic

January 23, 2014

1 Introduction

Being a competent programming means being able to understand an algorithm by its successful implementation. In this assignment, I'll discuss how to find the root of an equation of a single variable, *i.e.,* x where $f(x) = 0$. The algorithm is developed from an observation of slope and some modest calculus. An initial algorithm is constructed, and your assignment is to follow the discussion and then implement it. I'd suggest experimenting with several functions and their respective derivatives too. Remember the honor code statement for your code.

- Implement Algorithm 1.
- Include a check to ensure the best guess is *always* within the initial interval.
- After you've successfully implemented the algorithm, place the header it in `#include "mymathfunc.h"` and the function, called `rn1(...)` in `mymathfunc.cpp`.
- Contemplate what the function should return if it's not working correctly.
- Contemplate how accurate x_best needs to be for finance.

To remind you, contemplation questions are about thinking–no written answers are required; however, questions about your ruminations can pop-up in various scenarios where you *will* have to commit to writing something.

2 Roots using Newton-Raphson

In financial engineering, you'll often need to know the root of a function. The first implementation will be rather simple–I won't ask you to worry about aberrant conditions. Implement this in Visual Studio 10 as a console application. Call this project `RN1`.

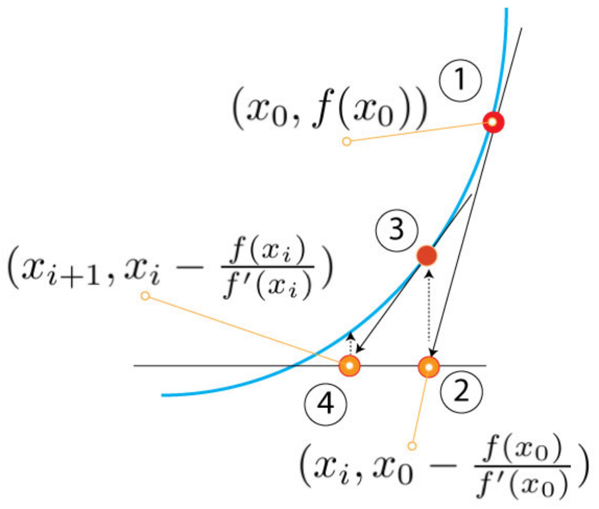
The most popular algorithm for finding roots of one parameter is the Newton-Raphson (`NR`). Informally, given an interval in which the the root exists, `NR` calculates the tangent whose value at the abscissa is the current guess. The point on the function at that value becomes the new locus for the tanget. See Fig. 1.

We leverage the Taylors series for our function f about some point $x = x_0 + \epsilon$. Symbolically,

$$f(x_0 + \epsilon) \quad = \quad f(x_0) + \epsilon f'(x_0) + \epsilon^2 (\frac{1}{2}) f''(x_0) + \ldots \tag{1}$$

Given the right conditions, we can ignore nonlinear terms in Eq. 1,

$$f(x_0 + \epsilon) \quad \approx \quad f(x_0) + \epsilon f'(x_0) \tag{2}$$



1
$(x_0, f(x_0))$
3
$(x_{i+1}, x_i - \frac{f(x_i)}{f'(x_i)})$
4
2
$(x_i, x_0 - \frac{f(x_0)}{f'(x_0)})$

Figure 1: A graphic showing the progression of the `NR` algorithm. The tangent (1) at $(x_0, f(x_0))$ is found, then the value on the abscissa (2) $(x_i, 0)$. The tangent at that point is found (3) and the next abscissa (4). The interval where the function is zero is also needed.

Returning to Fig. 1, observe that (1) is $(x_0, f(x_0))$ and I know the slope. How do I find where the function hits the abscissa? I have a point and the slope, so I can pick another point (x_i, y_i) and write

$$\frac{y_i - f(x_0)}{x_i - x_0} = f'(x_0) \tag{3}$$

$$y_i - f(x_0) = f'(x_0)(x_i - x_0) \tag{4}$$

I want the point (2) in Fig. 1, so $y_i = 0$. Continuing from Eq. 4 I find

Algorithm 1 Newton-Raphson find x, where $f(x) = 0$

```
1: Input $f(x)$, $f'(x)$, $[x_\ell, x_r]$, $\epsilon$, $maxIt$
2: // function, derivative, $x_{best} \in [x_\ell, x_r]$
3: // $\epsilon$ is precision or how close to slope of 0
4: // $maxIt$ is bound on iterations
5: Output $x_{best}$ where $f(x_{best}) = 0$
6: $x_{best} = (x_\ell + x_r)/2.0$
7: for $i = 0$ to $maxIt - 1$ do
8:     $f_{next} = f(x_{best})$
9:     $f'_{next} = f'(x_{best})$
10:    if $|f'_{next}| < \epsilon$ then
11:        break
12:    end if
13:    $x_{best} \leftarrow x_{best} - \frac{f_{next}}{f'_{next}}$
14: end for
15: return $x_{best}$
```

$$x_i = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{5}$$

```
struct func {
// original function
double f(double x) { return x*x - 7*x + 10;}
// derivative
double fp(double x) {return 2*x - 7;}
};

int main(void) {

    // create an instance
func myfunc;
cout << "function @ 3 = " << myfunc.f(3) << endl;
cout << "Slope @ 3 = " << myfunc.fp(3) << endl;
...
```

Figure 2: Using the `struct` construction for functions.

Eq. 6 gives an iterative method of finding the root:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{6}$$

An initial attempt at `NR` is shown in Algorithm 1.

I'll comment on a few of the details. First, I'll assume the there aren't any problems with the algorithm *per se*. This actually isn't the case–if the initial guess isn't good, for example, your answer will shoot into space. I also might go beyond the number of set iterations–and this means the accuracy isn't what I wanted. At this point, however, I didn't want to include more C++ than was necessary.

Line 1. gives the formal parameters. You'll need the function and its derivate. An easy way to do this is to use `struct`. in Fig. 2 and example is given using this construction. The ending condition is on 10-11. or 15. Line 10. is checking how close to slope of zero our estimate is. We need to use $abs|\cdot|$, which requires `#include <math.>`.

3 Summary

I've shown how to find a root of an equation by iteratively approximating the solution called Newton-Raphson. Root finding is needed in financial engineering, so having this algorithm is necessary. The algorithm itself needs to be improved by taking into account aberrant conditions. Additionally, you've been show a glimpse of object-oriented elements through a simple `struct`.