Numerical Mathematics Assignment 2 Peter-Jan Derks

Secant's Method

Description

The secant method is a root-finding algorithm that uses Secant lines to approximate a root of function f. A secant line is a straight line joining two points on a function. My function uses a succession of roots of secant lines to approximate a root of a function *f*. Starting with initial values x_0 and x_1, I construct a line through the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$. this line has the equation:

$$y = \frac{f(x2)-f(x1)}{x2-x1}(x - x1) + f(x1)$$

The root of the x values of this line at $y = 0$ can be found by solving the following equation for x:

$$0 = \frac{f(x2)-f(x1)}{x2-x1}(x - x1) + f(x1)$$

Solving for x gives me the equation I implemented in my function:

$$x = x2 - diff$$

with:

$$diff = f(x1)\frac{x2-x1}{f(x2)-f(x1)}$$

In my programm I run this iterative function in a while loop, until the difference is smaller than the tolerated difference or the number of iterations is larger than the max number of iterations. (Same as in Newton.m) I decided to use "diff" as my measure of precision because the diff becomes smaller as x approaches the root I am looking for. My function should be run with at least three inputs: the function, x1 and x2

Picture

The following input produces the following array with x-values, the root of the function and a graph showing the convergence to zero. I zoomed in on the graph to show the x values from 10 to 0.

```
>> Secant(@(x) (x-2)^5 - 600,30,10,0.0005,100)

xvalues =

  Columns 1 through 11

   30.0000   10.0000    9.9625    8.4145    7.6635    6.8836    6.3265    5.9221    5.6964    5.6110    5.5953

  Columns 12 through 13

    5.5944    5.5944

ans =

    5.5944
```

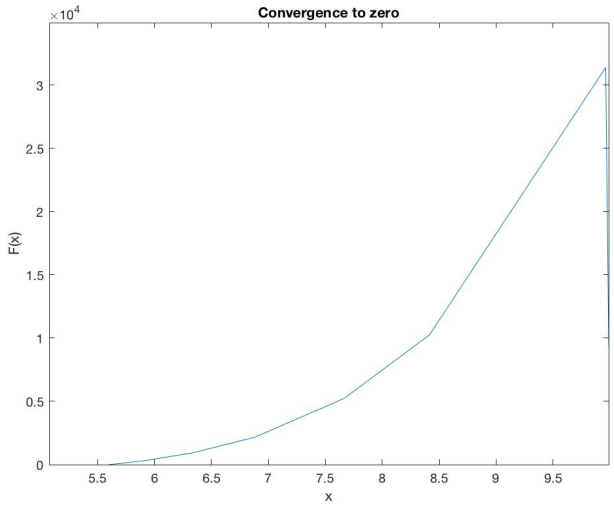

Figure 1

Questions

One of the drawbacks of using the secant method is that the initial guessed x0 and x1 need to be close enough to the root of the function for the algorithm to converge. A drawback of my program is that it is difficult for the user to estimate what kind of tolerated difference is needed for some precision.

The order of convergence of the secant method is approximately equal to 1.62. (Source: http://www1.maths.leeds.ac.uk/~kersale/2600/Notes/appendix_D.pdf). An advantage with respect to Newton's method is that only f(x) needs to be calculated at each x-value while for Newton's method both f(x) and f'(x) need to be calculated. The secant method may be faster than Newton's method if we assume the following:

1. Evaluating f takes as much time as evaluating its derivate.
2. We neglect the time all other calculations take.

If this is the case we can do two steps of the secant method for the same cost as one step of Newton's method. However in practice Newton's method is usually faster, with parallel

processing of the f(x) and f'(x) being one of the reason. The error depends on the requested tolerance by the user.

Modified Newton's Method

Description

The main challenge of this assignment was to find a formula to calculate the order of the zero. I derived the following formula:

When x is close to the root:

$$x_{n+1} = x_n$$

$$x_{n+1} - M\frac{f(x_{n+1})}{f'(x_n)} = x_n - M_0\frac{f(x_n)}{f'(x_n)}$$

$$M = M_0\frac{f(x_n)}{f'(x_n)} * \frac{1}{\frac{f(x_n)}{f'(x_n)} - \frac{f(x_{n+1})}{f'(x_{n+1})}}$$

$$M = M_0\frac{(x_n - x_{n+1})}{(x_n - x_{n+1}) - (x_{n+1} - x_{n+2})}$$

In my function I used the first three x-values (which I calculated using the classical Newton's method) to find the order of the zero.

The input fun = x^2 – 612, dfun = 2x, x0 = 50, tol = 0.0005 and nmax = 12 gives the following output: (order of zero, x – values and the root in figure 2 and the graph in figure 3 zoomed in in figure 4)

```
Command Window
>> modified_newton2(@(x) x^2 - 612, @(x) 2*x ,50,0.0005,12)

order =

    1.4354


zero =

   24.7388


xvalues =

  Columns 1 through 12

   50.0000   31.1200   25.3929   24.4658   24.8596   24.6864   24.7615   24.7287   24.7430   24.7368   24.7395   24.7383

  Column 13

   24.7388


ans =
```

Figure 2

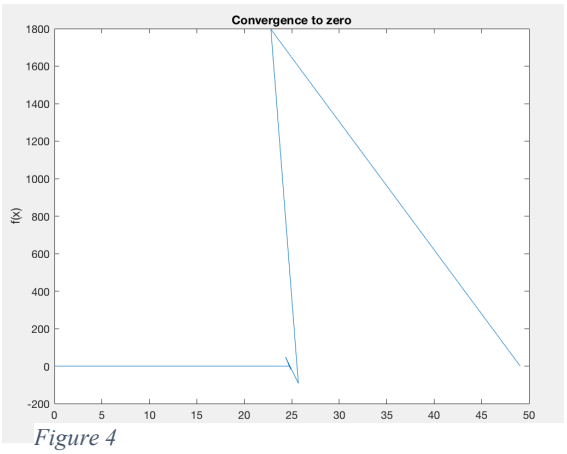

Figure 4

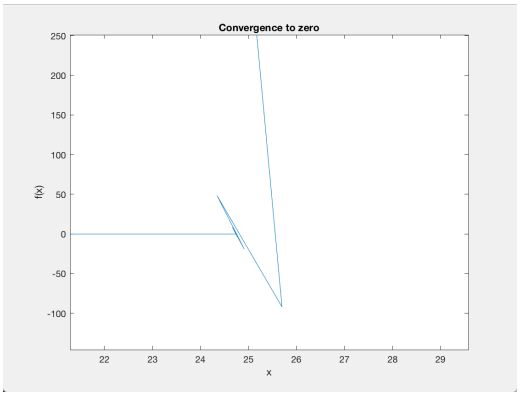

Figure 3

Questions

If the original Newton function is run with the same input as in figure 2, the zero is found much faster, in 5 iterations instead of 12.

```
>> newton(fun,dfun,50,0.0005,12)

niter =

     5


ans =

   24.7386
```

This means that my function doesn't have a higher order of convergence than Newton's function. As discussed in the lecture and course notes the goal of modifying Newton's method is to calculate an m such that φ_M '(x) = 1 – M/m = 0 if M = m.

The error depends on the (optional) tolerated input, I chose for it to be 0.005 if the user doesn't give any input.

In my code I tried calculating the order of convergence but could not figure it out. The formula I was trying to use is (the formula is derived in the course notes:

$$p \approx \frac{\ln\left(\frac{x_{n+1}-a}{x_n-a}\right)}{\ln\left(\frac{x_n-a}{x_{n-1}-a}\right)}$$

With a being the last x-value my function calculates.