In Class Exercise: Using Matlab, the Bisection and Newton's Method

Ritter Math 2335

Load both of the files bisect.m and Newt.m into the active directory in Matlab. Some commands you may wish to implement between runs are

- ”clc” this command clears the entire command window
- ”clear all” this command clears all assignments made to variable
- ”close all” if you've created any plots, this command will close all open figure windows

Exercise 1: Consider finding the root of the equation $x(1-x)+e^{-x}=0$ within a tolerance of 10^{-5}.

(a) Modify the code bisect.m so that the proper function is read. (Matlab's exponential is ”exp( )” with the argument between the parentheses.)

(b) Find the root using bisect.m with $a=1$ and $b=2$. Make note of how many iterations are required. (Does this match the estimate given by the formula $n \geq \ln((b-a)/\epsilon)/\ln(2)$?[1])

(c) Modify the code Newt.m to enter the proper function and its derivative.

(d) Use Newt.m to find the root with initial guesses of $x_0=1$, $x_0=2$, and with a really bad guess of $x_0=20$. Make note each time of how many iterations are required.

Exercise 2: Consider finding the root of the equation $x^2=0$ within a tolerance of 10^{-5}.

(a) Modify the code bisect.m so that the proper function is read.

(b) Find the root using bisect.m with $a=-1$ and $b=1$. Make note of how many iterations are required. (Does this match the estimate given by the formula $n \geq \ln((b-a)/\epsilon)/\ln(2)$?

[1]My code will always compute one more iterate after the tolerance is met. So, for example, if you determine that $n=15$ is needed, my code will compute $n=16$ iterations. You can modify the code to eliminate this if you wish.

(c) Modify the code Newt.m to enter the proper function and its derivative.

(d) Use Newt.m to find the root with initial guesses of $x_0 = -1$, $x_0 = 1$. Make note of how many iterations are required.

By considering the analysis we did in class regarding the speed of Newton's method, identify why convergence is so slow in this case? What's different about this function?

Exercise 3: Suppose we wish to find the one real root of the polynomial

$$f(x) = x^3 - 2x + 2.$$

Modify the Newton's method code for this function. Run the command from the Matlab command window Newt(0, 0.000001, 25). What does the output look like? Open the file "outdis.dat". Now compute x_1 and x_2 by hand. Can you see what is happening? Try other initial guesses x_0 until you find the actual root.

Exercise 4: Find all of the roots of the quartic equation

$$x^4 - 7.52x^3 + 6.9137x^2 + 33.9231x - 37.136 = 0.$$

Use any method you wish. It may be helpful to plot the quartic first; this will allow you to see about where the roots are. You can do this in Matlab by entering the following in the command window:

```
>> x = linspace(-3,6,1000);
>> y = x.^4-7.52*x.^3+6.9137*x.^2+33.9231*x-37.136;
>> plot(x,y)
>> grid on
```

The linspace command fills a vector x with real numbers from -3 to 6 using 1000 equally spaced points. In general, linspace(a, b, N) creates an array from a to b consisting of N equally spaced (of increment (b-a)/(N-1)) points.

Exercise 5: Find the true root of the equation $\sqrt[3]{x} = 0$. Try both methods. To obtain a real n^{th} root in Matlab, use the command "nthroot(a,n)" where a is the number you're finding the root of and n is a positive integer. If you ask Matlab to find $x^{1/3}$, it will compute the principle complex cube root of x. From the command window, enter both of the following to see the difference

```
>>(-1)^(1/3)
>>nthroot(-1,3)
```

Do both Newton's and bisection methods for find the solution of $\sqrt[3]{x} = 0$ work? Why or why not?

- Find an explicit formula for x_{n+1} when Newton's method is used. Simplify this as much as possible.
- For $x_0 \neq 0$, find $\lim_{n\to\infty} |x_{n+1}|$