Computational Engineering - Engr 8103
Problem Set #3

Allen Spain
avs81684@uga.edu

University of Georgia — 09 December 2019

Question 1

(5 pts.) Do this exercise with paper and pen, without using Matlab. Carry out the first four Newton iterations to find the root of $f(x) = x^3 - 2x + 2$ with an initial guess of $x_0 = 0$. Does it work? Explain the problem by sketching the graph of the function and the tangents.
Give Newton's Method $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, where n is the iteration

n	x_n	x_{n+1}
0	0	$x_1 = 0 - \frac{[(0)^3-2(0)+2]}{3(0)^2-2} = 0 - \frac{-2}{-2} = 1$
1	1	$x_2 = 1 - \frac{[(1)^3-2(1)+2]}{3(1)^2-2} = 1 - \frac{1}{1} = 0$
2	0	$x_3 = 0 - \frac{[(0)^3-2(0)+2]}{3(0)^2-2} = 0 - \frac{-2}{-2} = 1$
3	1	$x_4 = 1 - \frac{[(1)^3-2(1)+2]}{3(1)^2-2} = 1 - \frac{1}{1} = 0$

Newtons method doesn't converge on a root since the ratio of the derivative goes to zero near local minima.

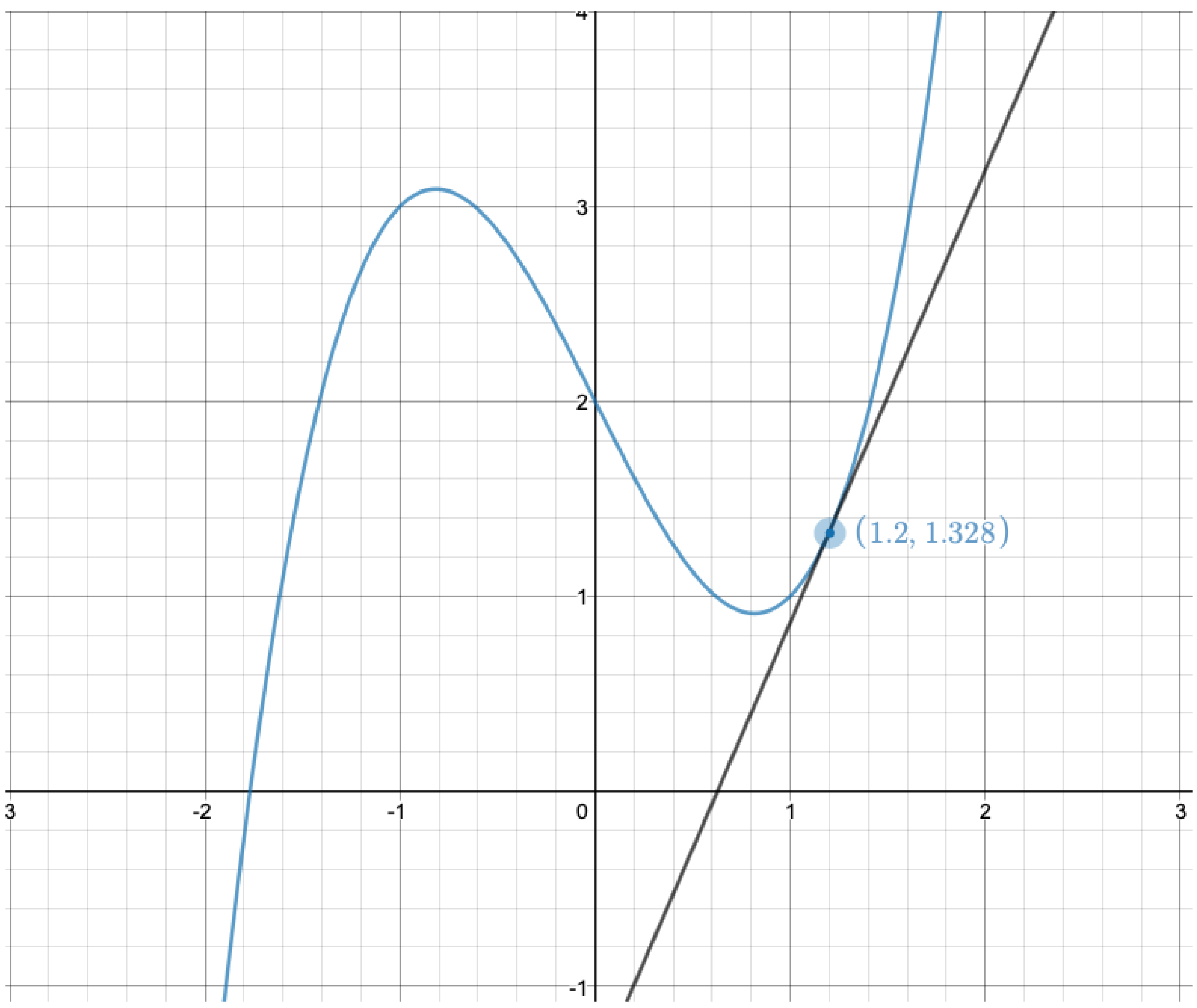

Figure 1: Tanget line of graph of $f(x)$

Question 2

The derivative of $sin(x)$ at $x = 2$ can be approximated using $\frac{1}{h}(sin(2 + h) - sin(2)$. Type the following three lines in Matlab to create a figure of the error $\mid cos(2) - \frac{1}{h}(sin(2 + h) - sin(2)) \mid$ for various values of $h = 1, 10^{-1}, 10^{-2}, ..., 10^{-20}$

(a) (2 pts.) Error is expected to decrease with h, as we observe on the right part of the plot. The left part of the plot however, shows that exactly the opposite occurs. Explain why the error increases with decreasing h beyond a certain point $(h \simeq 10^{-8})$.

As you approach ($\simeq 10^{-8}$) the error is no longer improved by decreasing the step size, and the roundoff error begins to accumulate due to worse and worse approximations. I.e the truncation error is only diminished to the point at which accurate approximations can be made.

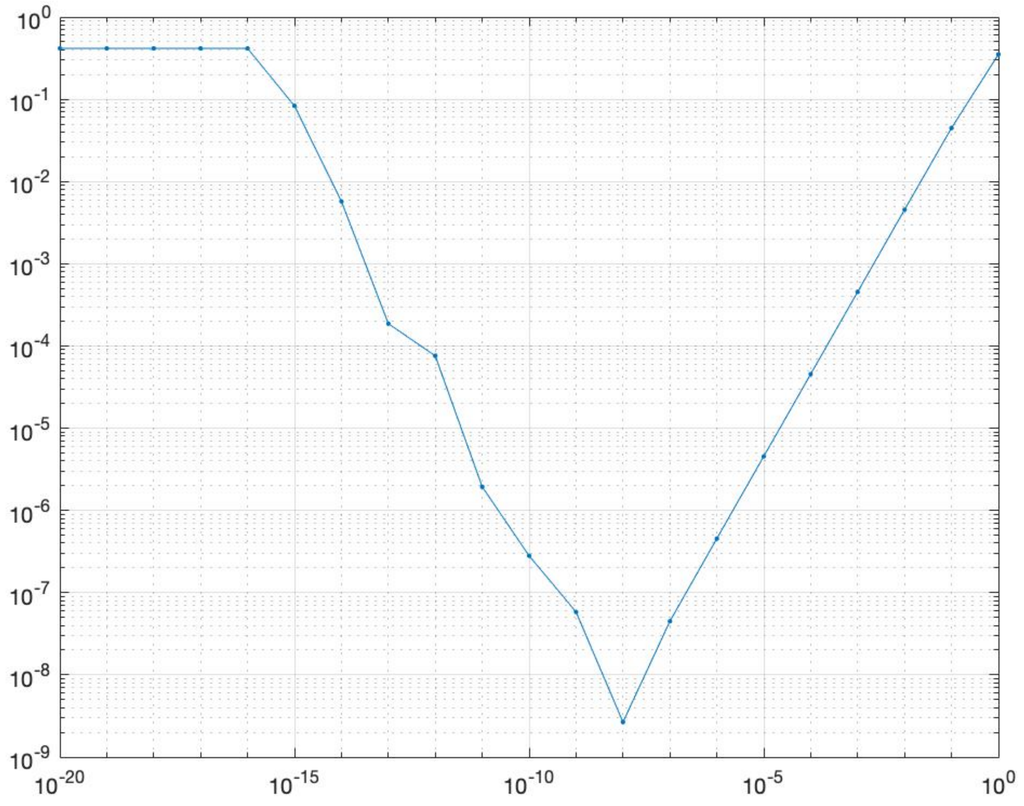

Figure 2: Tanget line of graph of $f(x)$

(b) (2pts.) Error appears to be exactly the same value ($\simeq 0.416147$) for $h = 10^{-16}, 10^{-17}, 10^{-18}, 10^{-19}, 10^{-20}$. Explain why. Where does this value come from?

Because $\epsilon_{mach} \simeq 10^{-16}$, therefore the function $\mid cos(2) - \frac{1}{h}(sin(2+h) - sin(2)) \mid$ will yield the same approximation, and thus the same error.

(c) (2 pts.) We know that this is a first order approximation to the first derivative. Since the figure shows how the error changes for various values of h, it is possible to estimate the order of this method using this figure. Verify that this is a first order method (or very close, it is an estimate after all) by using the figure. Feel free to print the figure and draw lines on it if it helps.

The graph is linear which shows uniform change in Y, as X changes. Emphrically when $h = 10^{-7}$, error $= 4.485x10^{-8}$ and $h = 10^{-8}$, error $= 4.547x10^{-7}$ so decreasing h by a factor of 10 decreases the error by $10x$ so this appoximation is first order.

d) (2 pts.) Replace the second line (errors = ...) to use the fourth order approximation to the first derivative, which you derived in the previous assignment:

$$f'^{(x)} \simeq \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h} + \mathcal{O}(h^4)$$

The minimum error you can achieve with the previous method is between 10^{-8} and 10^{-9}. What is the minimum error you can achieve change with this new method? Given that your computer's precision is fixed, clearly explain the reason for the improvement in the minimum error.

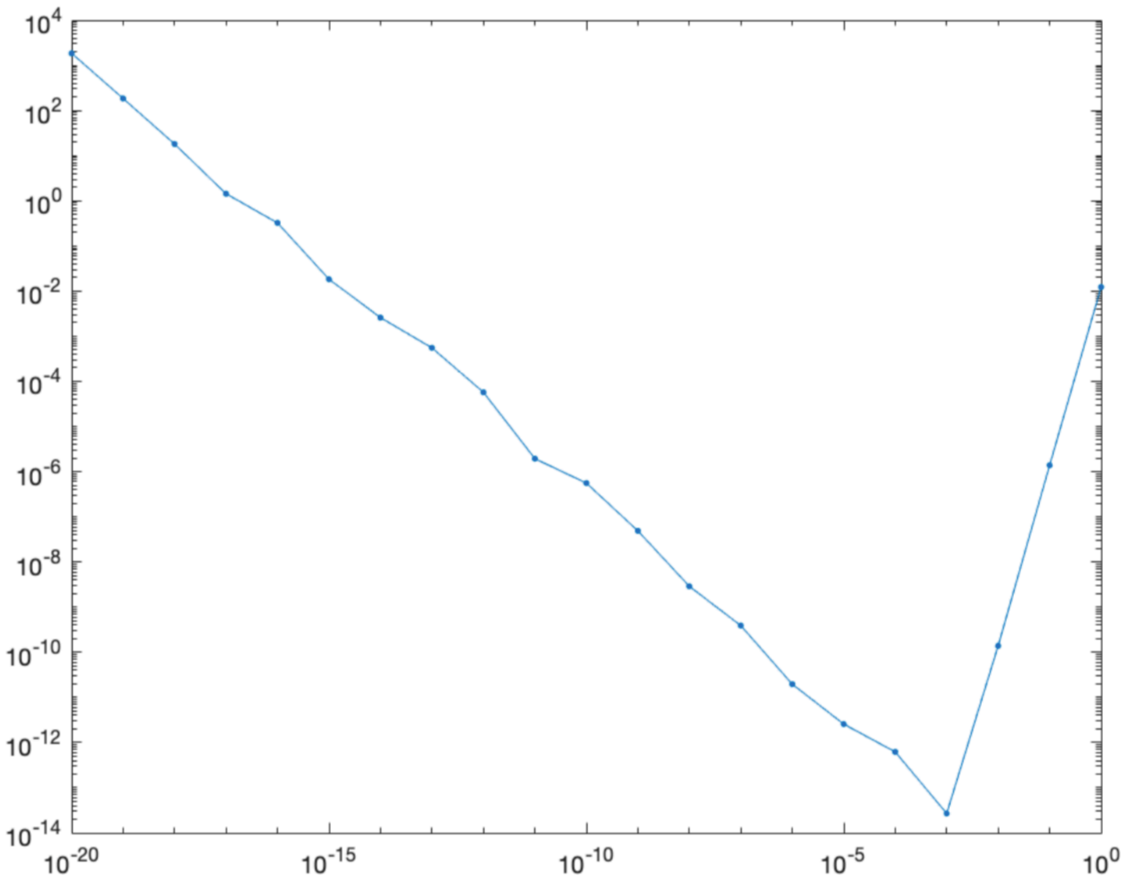

Figure 3: Fourth order error approximation

The new minimum error is $\simeq 10^{-14}$ the higher order method allows for better approximations as h decreases, this extra precision allows the number to have higher resolution.

(e) (2 pts.) Repeat part (c) for this new method.

Decreaasing h decrease by 10 times the errors decrease $\simeq 10^4$ times, so this system is 4_{th} order.

Question 3

(15pts.)The concentration of pollutant bacteria $c(t)$ in a lake decreases according to the following expression (unit is cfu/ml):

$$c = 75e^{-1.5t} + 20e^{-0.075t}$$

We'll use various methods to determine the time required for the bacteria concentration to be reduced to a safe level of $15cfu/ml$

(5 pts.) Write a Matlab code to solve this problem using Newton's method with an initial guess of $t_0 = 0$ and stopping criterion of 0.5% approximate relative error. At each iteration, your code should print (i) iteration number, (ii) approximated value of t, and (iii) approximate relative error. Print your code and the output of your code, and include them in your solutions. Name your code as newton.m and submit a

soft copy as an e-mail attachment with subject line “ENGR8103 HW3” to ml64719@uga.edu.

```
function newton

    c = @(t) 75 * exp(-1.5 * t) + 20 * exp(-0.075 * t) - 15;
    c_dt = @(t) -112.5 * exp(-1.5 * t) + -1.5 * exp(-0.075 * t);
    e = @(t_prev, t_curr) abs((t_curr - t_prev) / t_curr);

    x = 0;
    t_n = 0;
    approx_error = 100;
    tolerance =  0.05; % 5 percent tolerance
    i=0;

    % disp('iterations     root          error')
    while (approx_error > tolerance)
        t_prev = t_n;
        t_n = t_n - (c(t_n)/c_dt(t_n));
        approx_error = e(t_prev, t_n);
        i = i + 1;
        fprintf('(i) iteration: %d \n',i);
        fprintf('(ii) approximated val: %f \n',t_n);
        fprintf('(iii)relative error: %f \n',approx_error);
        fprintf('----------------------------------------\n')
    end
```

```
OUTPUT:

----------------------------------------
(i) iteration: 1
(ii) approximated val: 0.701754
(iii)relative error: 1.000000
----------------------------------------
(i) iteration: 2
(ii) approximated val: 1.442789
(iii)relative error: 0.513613
----------------------------------------
(i) iteration: 3
(ii) approximated val: 2.253249
(iii)relative error: 0.359685
----------------------------------------
(i) iteration: 4
(ii) approximated val: 3.125065
(iii)relative error: 0.278975
----------------------------------------
(i) iteration: 5
(ii) approximated val: 3.805303
(iii)relative error: 0.178761
----------------------------------------
(i) iteration: 6
(ii) approximated val: 3.994018
(iii)relative error: 0.047249
----------------------------------------
```

```
function secant
```

```
c = @(t) 75 * exp(-1.5 * t) + 20 * exp(-0.075 * t) - 15;
c_dt = @(t) -112.5 * exp(-1.5 * t) + -1.5 * exp(-0.075 * t);
e = @(t_prev, t_curr) abs((t_curr - t_prev) / t_curr);

x = 0;
t_n0 = 0;
t_n1 = 1;
approx_error = 100;
tolerance =  0.05; % 5 percent tolerance
i=0;

% disp('iterations    root         error')
while (approx_error > tolerance)
    t_prev = t_n1;
    t_n1 = t_n1 - c(t_n1)*((t_n1 - t_n0)/(c(t_n1) - c(t_n0)));
    approx_error = e(t_prev, t_n1);
    i = i + 1;
    fprintf('(i) iteration: %d \n',i);
    fprintf('(ii) approximated val: %f \n',t_n1);
    fprintf('(iii)relative error: %f \n',approx_error);
    fprintf('----------------------------------------\n')
end
```

```
OUTPUT:
i) iteration: 1
(ii) approximated val: 1.339801
(iii)relative error: 0.253620

----------------------------------------
(i) iteration: 2
(ii) approximated val: 1.603115
(iii)relative error: 0.164252
----------------------------------------
(i) iteration: 3
(ii) approximated val: 1.819303
(iii)relative error: 0.118830
----------------------------------------
(i) iteration: 4
(ii) approximated val: 2.003240
(iii)relative error: 0.091820
----------------------------------------
(i) iteration: 5
(ii) approximated val: 2.163499
(iii)relative error: 0.074074
----------------------------------------
(i) iteration: 6
(ii) approximated val: 2.305466
(iii)relative error: 0.061578
----------------------------------------
(i) iteration: 7
(ii) approximated val: 2.432752
(iii)relative error: 0.052322
----------------------------------------
(i) iteration: 8
(ii) approximated val: 2.547900
(iii)relative error: 0.045193
----------------------------------------
```

```
function bisect

    % c = @(t) 75 * exp((-1.5 * t)) + (20 * exp((-0.075 * t))) - 15;
    % c = @(t) (75 * exp((-1.5*t)) + (20*exp(-0.075*t))) - 15;
    c = @(t) (75 * exp((-1.5 * t))) + (20 * exp((-0.075 * t))) - 15;
    % c_dt = @(t) -112.5 * exp(-1.5 * t) + -1.5 * exp(-0.075 * t);
    % e = @(t_prev, t_curr) abs((t_curr - t_prev) / t_curr);
    e = @(x_1,x_2) abs((x_2 - x_1)/ 2);

    x_1 = 0
    x_2 = 10
    approx_error = 100;
    tolerance =  0.05; % 5 percent tolerance
    i=1;

    % disp('iterations    root         error')
    while (approx_error > tolerance)
        tmp = (x_1 + x_2)/2;
        if (c(x_1) * c(x_2) < 0)
            x_2 = tmp;
        else
            x_1 = tmp;
        end
        approx_error = e(x_1, x_2);

        fprintf('(i) iteration: %d \n',i);
        fprintf('(ii) approximated val: %f \n',tmp);
        fprintf('(iii)relative error: %f \n',approx_error);
        fprintf('----------------------------------------\n')
        i = i + 1;
    end

    % function cfunc = c(t)
    % cfunc = (75 * exp((-1.5 * t)) + (20 * exp((-0.075 * t))) - 15;
    % end
```

OUTPUT

```
i) iteration: 1
(ii) approximated val: 5.000000
(iii)relative error: 2.500000
----------------------------------------
(i) iteration: 2
(ii) approximated val: 2.500000
(iii)relative error: 1.250000
----------------------------------------
(i) iteration: 3
(ii) approximated val: 1.250000
(iii)relative error: 0.625000
----------------------------------------
(i) iteration: 4
(ii) approximated val: 1.875000
(iii)relative error: 0.312500
----------------------------------------
(i) iteration: 5
(ii) approximated val: 2.187500
(iii)relative error: 0.156250
```

```
----------------------------------------
(i) iteration: 6
(ii) approximated val: 2.343750
(iii)relative error: 0.078125
----------------------------------------
(i) iteration: 7
(ii) approximated val: 2.421875
(iii)relative error: 0.039062
----------------------------------------
```

Question 4

(7 pts.) Consider the following equation system:

$$x + y + z = 0$$
$$x^2 + y^2 = 5$$
$$x(x + y) = -1$$

To find a solution this nonlinear equation system, carry out the first two Newton iterations starting with $(x_0, y_0, z_0) = (1, 1, 1)$ to find (x_1, y_1, z_1) and (x_2, y_2, z_2). Show all your work clearly. Feel free to use a calculator to find the inverse of the matrix.

From Newton's method for section

$$X_{n+1} = X_n - [\nabla f(X_n)]^{-1} f(X_n)$$

$$\begin{bmatrix} \frac{\partial f(x,y,z)_1}{\partial x} & \frac{f(x,y,z)_1}{\partial y} & \frac{f(x,y,z)_1}{\partial z} \\ \frac{\partial f(x,y,z)_2}{\partial x} & \frac{\partial f(x,y,z)_2}{\partial y} & \frac{\partial f(x,y,z)_2}{\partial z} \\ \frac{\partial f(x,y,z)_3}{\partial x} & \frac{\partial f(x,y,z)_3}{\partial y} & \frac{\partial f(x,y,z)_3}{\partial z} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 2x & 2x & 0 \\ y+z & x & x \end{bmatrix}$$

Taking the inverse of the matrix:

$$[\nabla f(X_n)]^{-1} = \begin{bmatrix} -\frac{x}{x+z-x} & 0 & \frac{1}{y+z-x} \\ \frac{x^2}{y(y+z-x)} & \frac{1}{2y} & -\frac{x}{y(y+z-x)} \\ \frac{y(y+z)-x^2}{y(y+z-x)} & -\frac{1}{2y} & -\frac{y-x}{y(y+z-x)} \end{bmatrix}$$

For the first iteration $(x_0, y_0, z_0) = (1, 1, 1)$

$$\therefore X_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} -1 & 0 & 1 \\ 1 & 0.5 & -1 \\ 1 & -0.5 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix}$$

$$\Rightarrow X_1 = \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix}$$

For the second iteration $(x_1, y_1, z_1) = (2, -1, -1)$

$$\therefore X_1 = \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0.5 & 0 & -0.25 \\ 1 & -0.5 & -0.5 \\ -0.5 & 0.5 & 0.75 \end{bmatrix} \begin{bmatrix} 0 \\ 5 \\ -4 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ -0.5 \\ -0.5 \end{bmatrix}$$

$$\Rightarrow X_2 = \begin{bmatrix} 1 \\ -0.5 \\ -0.5 \end{bmatrix}$$