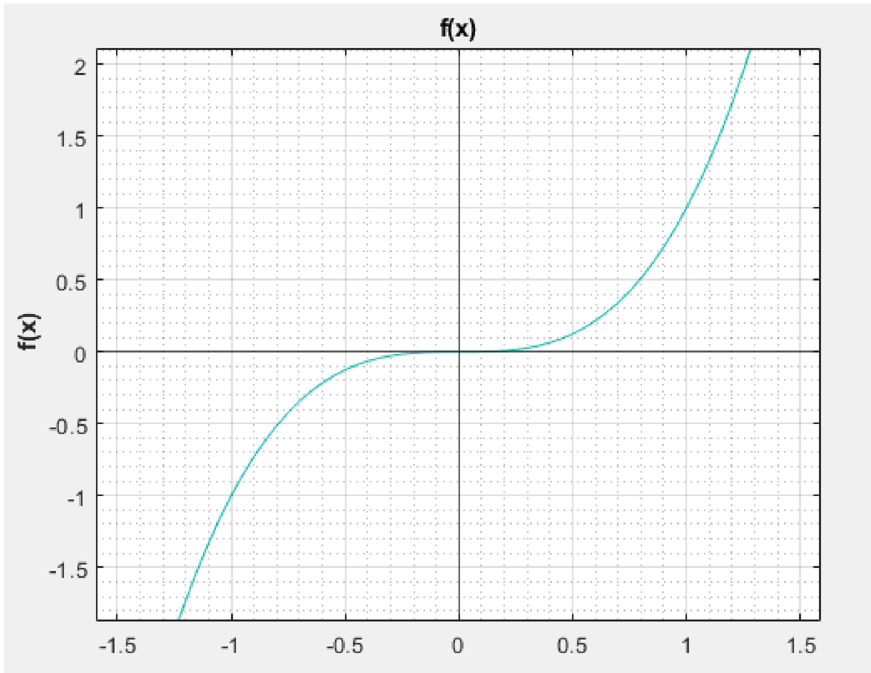
HW 6

MATH-375

Michael Tanguay

WRITTEN

1.1)



There is one zero. (One can imagine the plot shifting y for x^3 -y)

3.)

Bisection: root = 2.000000000029104, k =33

Newton: root = 2, k = 7

Secant: root =-2, k = 10.

Note my MATLAB runs the loop and additional time after a break and I have no idea why. This can explain why it could be off by an iteration.

4.)

	Bisection	
k	Error	Relative Rate
0	0.50000000000	0.5000000
1	0.25000000000	0.5000000
2	0.12500000000	0.5000000
3	0.06250000000	0.5000000
4	0.03125000000	0.5000000
5	0.01562500000	0.5000000
6	0.00781250000	0.5000000
7	0.00390625000	0.5000000
8	0.00195312500	0.5000000
9	0.00097656250	0.5000000
10	0.00048828125	0.5000000
11	0.00024414063	0.5000000
12	0.00012207031	0.5000000
13	0.00006103516	0.5000000
14	0.00003051758	0.5000000
15	0.00001525879	0.5000000
16	0.00000762939	0.5000000
17	0.00000381470	0.5000000
18	0.00000190735	0.5000000
19	0.00000095367	0.5000000
20	0.00000047684	0.5000000
21	0.00000023842	0.5000000
22	0.00000011921	0.5000000
23	0.00000005960	0.5000000
24	0.00000002980	0.5000000
25	0.00000001490	0.5000000
26	0.00000000745	0.5000000
27	0.00000000373	0.5000000
28	0.00000000186	0.5000000
29	0.00000000093	0.5000000
30	0.00000000047	0.5000000
31	0.00000000023	0.5000000
32	0.00000000012	0.5000000
33	0.00000000006	N/a

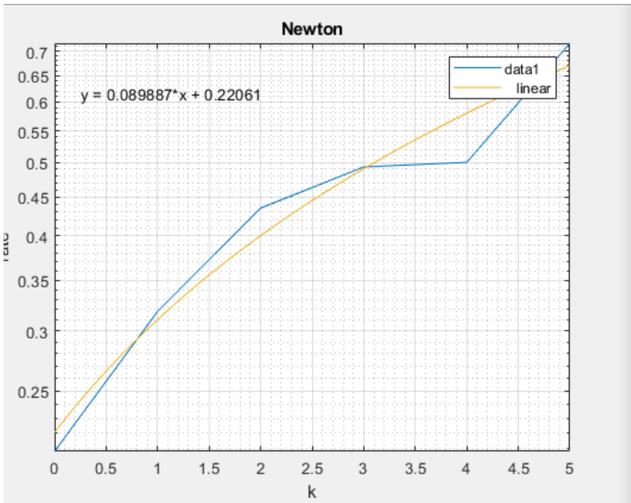
	Newton	
k	Error	Relative Rate
0	2.00000000000000000	0.20833333333
1	0.83333333333333300	0.31833910035
2	0.22106881968473700	0.43529603779
3	0.02127353680911260	0.49300191640
4	0.00022311460789837	0.49992563650
5	0.00000002488636230	0.71704660229
6	0.00000000000000044	0.00000000000
7	0.00000000000000000	n/a

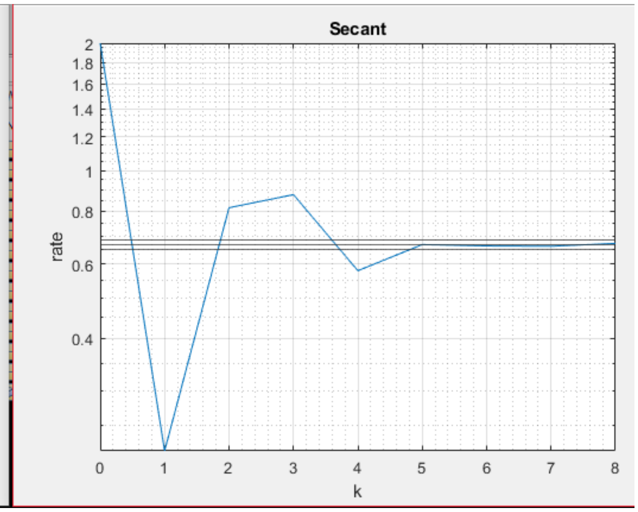
	Secant	
k	Error	Relative Rate
0	1.0000000000000	2.00000000000
1	2.0000000000000	0.21689030924
2	0.6666666666667	0.81599439731
3	0.4230769230769	0.87638744026
4	0.2175181351163	0.57905552721
5	0.0489173446794	0.66770142015
6	0.0050293534409	0.66274898401
7	0.0001252560653	0.66154156309
8	0.0000003154940	0.67201157064
9	0.0000000000198	0.00000000000
10	0.0000000000000	n/a

The rate of convergence for the methods are displayed in the tables attached.

For linear convergence, relative rates are consistent. This was the expectation of the bisectional method since the error was continuously cut into halves. Thus, resulting in a consistent relative rate

For both Newton and Secant it is difficult to discern a pattern (probably due to noise). Especially because the second to last iteration jumps straight to 0 when is should be a number close to that of the 3rd to last iteration. (Due to machine epsilon). To get an idea of the pattern, two log-scale plots were made. An average was taken for the secant and a linear fit was associated with Newton to attain the 2nd to last guess for the rate convergence. The average is around 0.66600088 for secant. For Newtons, the linear for a rate for k = 6 is around 0.759932. Again, these convergences will be more





5.) Convergence rate is 1.000001030145416 for Newtons Method.

<u>**MATLAB CODES**</u>

PROBLEM 1 and 2)

```
%% 1
clear, clc, close all
syms x y
ezplot(x^3 -y); grid on; grid minor; yline(0), xline(0); title('f(x)'); ylabel( '\bf f(x)'); xlabel( '\bf x')
%% 2.a
clear, clc, close all, format long
f =@(x)  x-x.^(1/3) -2; %function
xmid = my_bisection(f,3,4,10^-4,10); %function f veiwed from a to b with a tolerance of 10^-4
%Amount of itterations k
fxmid = f(xmid);
%% 2.b
clear, clc, close all, format long
f = @(x)  x-x.^(1/3) -2; %function
df = @(x) 1 -1/3*x.^(-2/3); %derivitive of f
xmid = my_newton(f,df,3,10^-15,4); %function f veiwed from a to b with a tolerance of 10^-4  %Amount
of itterations k
fxmid = f(xmid);
%% 2.c
clear, clc, close all, format long
f =@(x)  x-x.^(1/3) -2; %function
xmid = my_secant(f,4,3,10^-15,5); %function f veiwed from a to b with a tolerance of 10^-4  %Amount
of itterations k
fxmid = f(xmid);
```

PLOBLEM 3

```
%% 1
clear, clc, close all
syms x y
ezplot(x^3 -y); grid on; grid minor; yline(0), xline(0); title('f(x)'); ylabel( '\bf f(x)'); xlabel( '\bf x')
%% 3.a
clear, clc, close all, format long
f =@(x)  x.^3-8; %function
xmid = my_bisection(f,1,4,10^-10,100); %function f veiwed from a to b with a tolerance of 10^-4
%Amount of itterations k
fxmid = f(xmid);
%% 3.b
clear, clc, close all, format long
f =@(x)  x.^3-8 %function
df = @(x) 3*x.^2; %derivitive of f
xmid = my_newton(f,df,4,10^-10,100); %function f veiwed from a to b with a tolerance of 10^-4
%Amount of itterations k
```

```
fxmid = f(xmid);
```

PROBLEM 4

```
%% 3.c
clear, clc, close all, format long
f =@(x)  x.^3-8; %function
xmid = my_secant(f,1,4,10^-10,100); %function f veiwed from a to b with a tolerance of 10^-4  %Amount
of itterations k
fxmid = f(xmid);
%% 1
clear, clc, close all
syms x y
ezplot(x^3 -8); grid on; grid minor; yline(0), xline(0); title('f(x)'); ylabel( '\bf f(x)'); xlabel( '\bf x')
%% 3.a
clear, clc, close all, format long
f =@(x)  x.^3-8; %function
xmid = my_bisection(f,1,4,10^-10,100); %function f veiwed from a to b with a tolerance of 10^-4
%Amount of itterations k
fxmid = f(xmid);
sizek = length(xmid);
errorn = (abs(xmid - 2))';
errornplus1 = (abs(xmid(2:end) - 2))';
relativerate = (errornplus1./errorn(1:end-1))'; relativerate = relativerate'; relativerate(end)
k = [1:length(xmid)] -1;
plot(k(1:end-1), log10(relativerate)); grid on; grid minor
%% 3.b
clear, clc, close all, format long
f =@(x)  x.^3-8 %function
df = @(x) 3*x.^2; %derivitive of f
xmid = my_newton(f,df,4,10^-10,100); %function f veiwed from a to b with a tolerance of 10^-4
%Amount of itterations k
fxmid = f(xmid);
sizek = length(xmid);
errorn = (abs(xmid - 2))';
errornplus1 = (abs(xmid(2:end) - 2))';
relativerate = (errornplus1./errorn(1:end-1).^2)'; relativerate = relativerate'; relativerate(end)
k = [1:length(xmid)] -1;
plot(k(1:end-1), (relativerate)); grid on; grid minor; xlim([0 5])
%% 3.c
clear, clc, close all, format long
f =@(x)  x.^3-8; %function
xmid = my_secant(f,1,4,10^-10,100); %function f veiwed from a to b with a tolerance of 10^-4  %Amount
of itterations k
```

```
fxmid = f(xmid);
sizek = length(xmid);
errorn = (abs(xmid - 2))';
errornplus1 = (abs(xmid(2:end) - 2))';
relativerate = (errornplus1./errorn(1:end-1).^1.62)'; relativerate = relativerate'; relativerate(end)
k = [1:length(xmid)] -1;
plot(k(1:end-1), (relativerate)); grid on; grid minor

clear, clc, close all
semilogy(k,y); grid on; grid minor; title('Newton'); xlabel('k'); ylabel('rate')

k2 = [0:8]
x2 = [1.0000000000000
2.0000000000000
0.6666666666667
0.4230769230769
0.2175181351163
0.0489173446794
0.0050293534409
0.0001252560653
0.0000003154940];
y2 = [2.00000000000
0.21689030924
0.81599439731
0.87638744026
0.57905552721
0.66770142015
0.66274898401
0.66154156309
0.67201157064]
figure(2)
semilogy(k2,y2); grid on; grid minor; title('Secant'); xlabel('k'); ylabel('rate')
average = mean(y2(2:end))
average2 = mean(y2(3:end))
average3 = mean(y2(4:end))
average4 = mean(y2(5:end))
average5 = mean(y2(6:end))

yline(average3); yline(average4); yline(average5);
% yline(average); yline(average2);

y3 = @(x) 0.089887*x + 0.22061
y3(6)
```

Problem 5

```
%% 3.b
clear, clc, close all, format long
f =@(x)  x.^3 %function
df = @(x) 3*x.^2; %derivitive of f
xmid = my_newton(f,df,4,10^-10,100); %function f veiwed from a to b with a tolerance of 10^-4
%Amount of itterations k
fxmid = f(xmid);
sizek = length(xmid);
errorn = (abs(xmid - 2))';
errornplus1 = (abs(xmid(2:end) - 2))';
relativerate = (errornplus1./errorn(1:end-1))'; relativerate = relativerate'; relativerate(end)
```

Functions used in every problem

<u>Bisectional Method</u>

```
function [x_arr] = my_bisection(f,a,b,tol,k)
%where f is a function pointer to the function in question,
%a, b are the initial brackets,
%and tol is the halting tolerance The array
%x_arr is the return value, an array of the root guesses. That is
%the first entry of x arr will be the initial root guess, and the last
%entry will be the final (and most accurate) root guess.
%k is the number of iterations

for k = 1:k
   xm = a + (b - a)/2;
   toli = (b-a)/2; %Current Toleranne
      if sign (f(xm)) == sign (f(a))
         a = xm;
      else
         b = xm;
      end
      if abs(toli) < tol
         disp('Current tolereance exceeds specified tolerance')
         break
      end
       if  abs(f(xm)) < eps
         disp('Machine epsilon tolerance')
         break
      end
      x_arr(k) = xm;
end
```

end

Newtons's Method

```
function [x_arr] = my_newton(f, df, x_0,tol, k)
%where f is still a function pointer, df is a function pointer to the
%derivative of f, x 0 is the initial guess to a root, and tol is the
%halting tolerance. %k is the number of itterations
x(1) = x_0;
for k = 1:k
 x(k+1) = x(k) - f(x(k))/df(x(k));;
 toli = x(k+1) - x(k);
  if abs(toli) < tol
      disp('Current tolereance exceeds specified tolerance')
      break
  end
   if  abs(f(x(k+1))) < eps
       disp('Machine epsilon tolerance')
      break
   end

end
 x_arr = x;
end
```

Secant Method

```
function [x_arr] = my_secant(f, x_0, x_1,tol, k)

x(1)  = x_0; x(2) = x_1;
for k = 1:k
 x(k+2) = x(k+1) - f(x(k+1))*(x(k+1) - x(k))/(f(x(k+1)) - f(x(k)));
  toli = x(k+2) - x(k+1); % Current Tolerance
   if abs(toli) < tol
      disp('Current tolereance exceeds specified tolerance')
      break
  end
  if  abs(f(x(k+2))) < eps
      disp('Machine epsilon tolerance')
      break
  end
end
 x_arr = x;
end
```