Nicholas Livingstone HW #4 CS-241

1. Fixed points and Derivative

```
hold off
x = [-2, 1, 3];
y = x;
plot(x, y, 'o');
hold on
x_p = linspace(-4, 4);
y_pneg2 = 2.5 .* x_p + 3; %slope at negative 2
y_p3 = 0.5 * x_p + 1.5;
plot(x_p, y_p3);
plot(x_p, y_pneg2);
title('Problem 1')
xlabel('X')
ylabel('Y')
grid on
hold off
```

If there exists a sufficiently close x_0 that converges to $y = 3,\ f'(3) < 1$. And since $f'(-2) > 1,\ f'(1) = 1$.

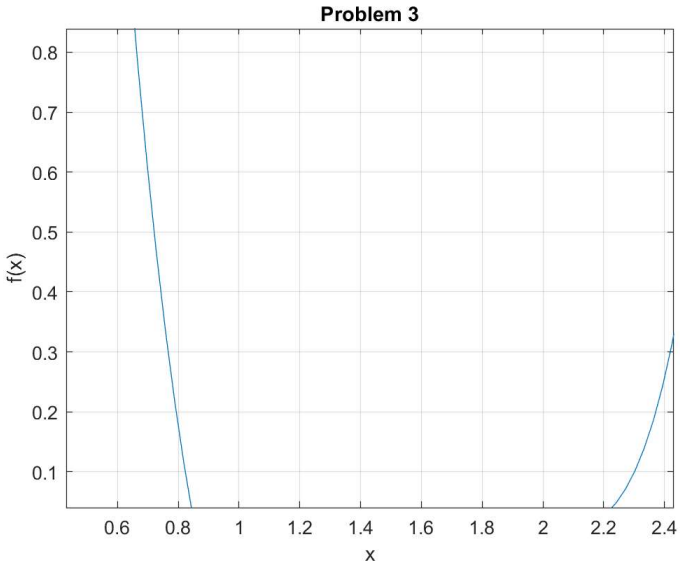
2. Theoretical Considerations of Root Finding Methods

```
x = linspace(-0.1, 0.6);
y = 8.*x.^4 - 12 .*x.^3 + 6.*x.^2 - x;
plot(x, y);
grid on;
hold on;
x = [0, 0.5];
y = [0, 0];
plot(x, y, 'o', 'LineWidth', 1);
hold off;
title('Problem 2');
xlabel('x');
ylabel('y');
ylim([-0.1, 0.2]);
```

Newton's method will likely have problems converging at $x = 0.5$ as the derivative appears to closely approach 0, therfore in this case it would be more beneficial to use the bisection algorithm since it is not dependent on derivative. However, in the case of $x = 0$, it makes more sense to use Newton's method. It's more obvious here that f(x) changes sign on both sides of the root and will have no problems converging to. The derivative is much less than one and will converge in much fewer iterations than the bisection algorithm. However, because the derivative apears to change shortly above $x = 0.1$, it's crucial that an inital guess lower than this value is chosen, otherwise it's likely that the algorithm will converge to the other root.

3. Convergence of Newton's Method

```
x = linspace(0,3);
f = @(x) 14 .*x.*exp(x - 2) - 12 .* exp(x-2) - 7.*x.^3 + 20.*x.^2 - 26 .* x + 12;
df = @(x) 40.*x + 2.*exp(x - 2) + 14.*x.*exp(x - 2) - 21.*x.^2 - 26;
ddf = @(x) 16.*exp(x - 2) - 42.*x + 14.*x.*exp(x - 2) + 40;
plot(x, f(x))
grid on;
xlabel('x');
ylabel('f(x)')
title('Problem 3');
ylim([-0.4, 0.4]);
```



Finding the first root

```
[root1, xvalues1, iterations1, e_i1] = newton(f, df, 1);
```

Iteration	X	ei	e_{i+1}/e_i^2	e_{i+1}/e_i
0	1.0000000000000	0.2642411176571140		
1	0.7627845835849	0.3143283473368120	4.5017643113876800	0.144605561977373
2	0.8408216918290	0.0454536273120585	0.4600462007406020	0.035973089364281
3	0.8565332220104	0.0016351073972274	0.7914239520931990	0.001466410903410
4	0.8571419618530	0.0000023977393155	0.8968284932821730	0.000002161790159
5	0.8571428571409	0.0000000000051834	0.9015951590249010	0.000342700479781
6	0.8571428571429	0.0000000000000018		

```
M = abs(ddf(root1)/(2*df(root1)))
```

```
M =
    2.413850894567765
```

Since M is not equal to the limit of the error ratio, Newton's method is not quadratically convergent to the first root. However, because the function is infinitely differentiable in the neighboorhood of the first root $(m-1)/m = \frac{\infty}{\infty}$ and is undefinied. Therefore it is neither quadratically or linearly convergent to the first root but faster.

Finding the second root

```
[root2, xvalues2, iterations2, e_i2] = newton(f, df, 2.1);
```

Iteration	X	ei	e_{i+1}/e_i^2	e_{i+1}/e_i
0	2.1000000000000	0.0029739745162587		
1	2.0678025694470	0.0008956461818883	101.2656123740400000	0.299731919629489
2	2.0457395265384	0.0002684537494062	334.6543821551920000	0.298682078588080
3	2.0307429387881	0.0000801823238774	1112.6016278362100000	0.297932602025231
4	2.0206098527386	0.0000238889283892	3715.6892893344000000	0.297408629490091
5	2.0137919059181	0.0000071047734522	12449.6429745346000000	0.297047747007110
6	2.0092180512705	0.0000021104569470	41809.6015312088000000	0.296801775339277
7	2.0061558900898	0.0000006263873686	##########################	0.296635318327352
8	2.0041086339912	0.0000001858086165	##########################	0.296523190343430
9	2.0027411905751	0.0000000550965638	##########################	0.296447890111001
10	2.0018283970609	0.0000000163332601	##########################	0.296398141211867
11	2.0012193487417	0.0000000048411479		

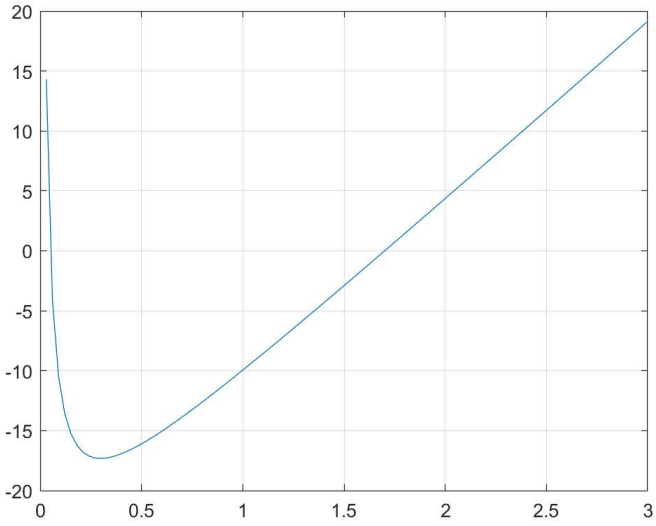
Newton's method is linearly convergent with a convergence rate of ≈ 0.3.

4. Ideal Gas Law

```
n = 1; %mol of oxygen
T = 320; %Degrees K
P = 15; %Pressure (atm)
R = 0.0820578; %Gas Constant
%oxygen constants
a = 1.36;
b = 0.003183;

v = linspace(0, 3);
vander = @(V) (P + ((a * n^2)./V.^2)).*(V - n*b) - n*R*T;
dvander = @(V) 34/(25*V^2) - (68*(V - 1834874574581797/576460752303423488))/(25*V^3) + 15; %derivative produced using diff function
plot(v, vander(v));
```

```
grid on;
```



Inital guess based on graph = 1.5;

```
root = newton(vander, dvander, 1.5);
fprintf("%.2f", root)
```

```
1.70
```

```
function [root, xval, iter, errs] = newton(f, df, x0, tol, kmax)
%Newton's method root finding algorithm
    switch nargin
        case 3,
            tol = 1e-8;
            kmax = 1e5;
        case 4,
            kmax = 1e5;
        case 5
            %do nothing
        otherwise,
            error('Invalid number of arguments')
    end

    errs = [];
    xval = [];
    iter = [];

    for k = 0:kmax
        iter = [iter k];
        y = f(x0);
        x = x0 - y/df(x0);
        xval = [xval x0];
        err = abs(y);
        errs = [errs err];
        x0 = x;
        if err < tol
            errs = [errs abs(f(x))];
            root = x;
            return;
        end
    end
end
```