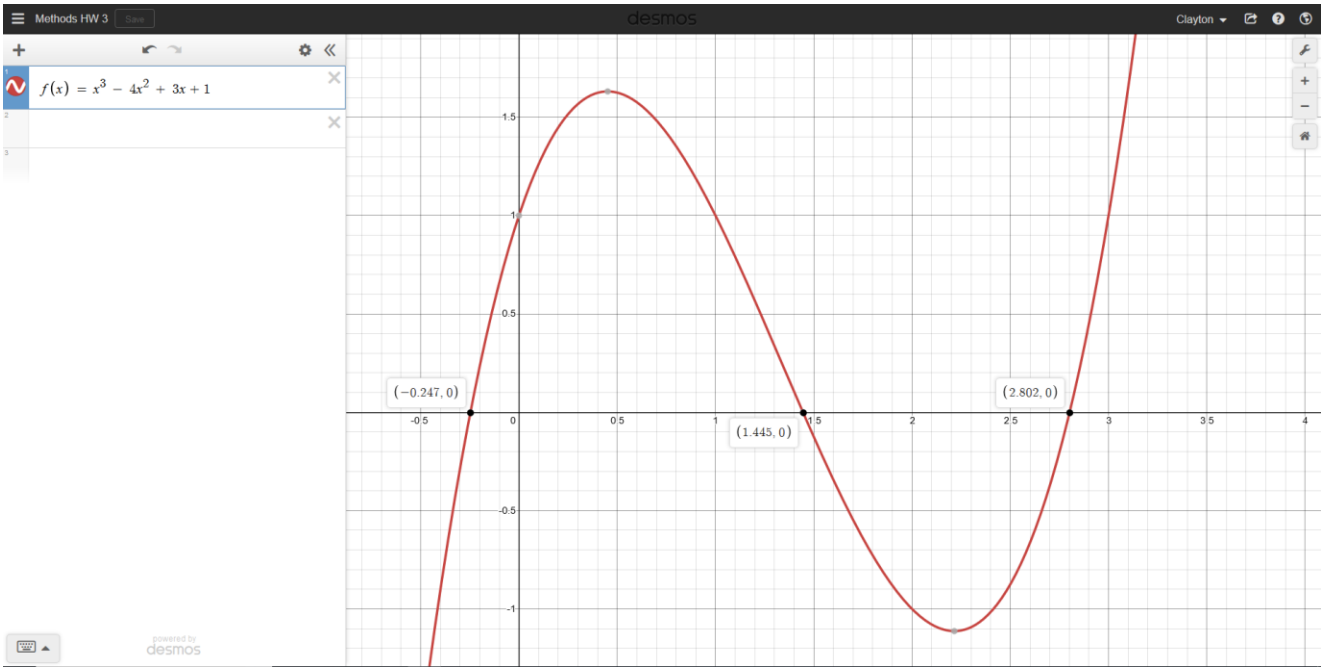
MATH 366 Methods of Applied Mathematics II

Clayton Johnson

HW 3: Ch 19.2 Newton's Method with Python and MATLAB

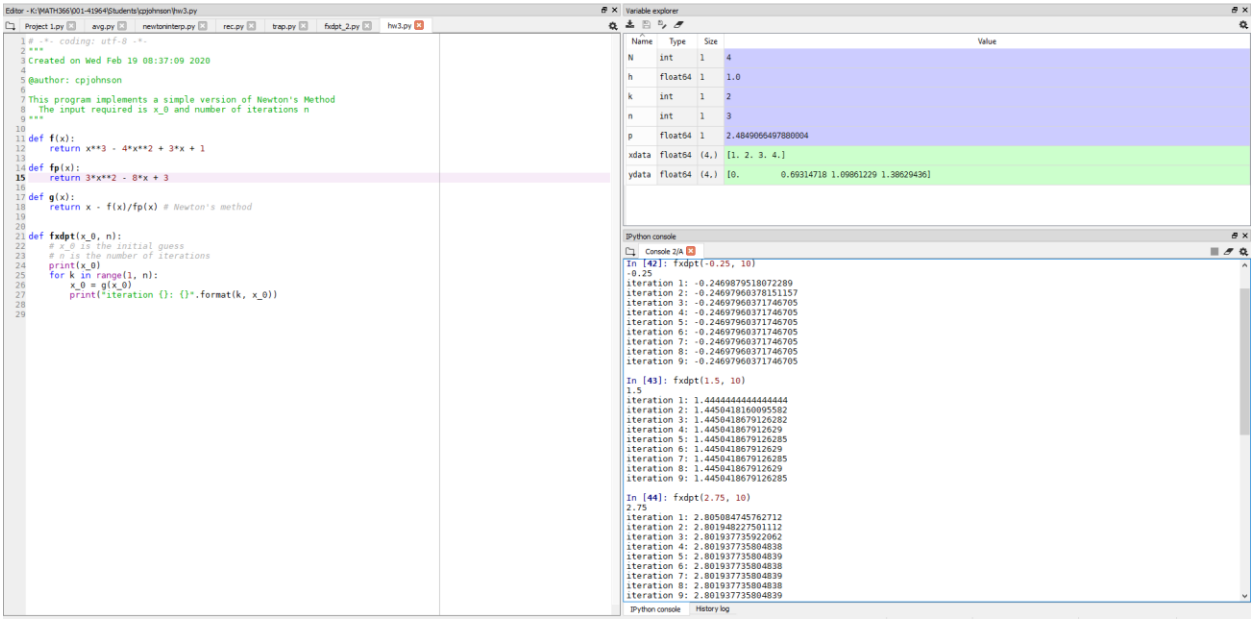
Desmos



The function *f* has three roots located around -0.25, 1.5, and 2.75. These guesses will be put into the Python and MATLAB programs.

Python

Spyder Program



Python Code

Editor - K:\MATH366\001-41964\Students\cpjohnson\hw3.py

Project 1.py | avg.py | newtoninterp.py | rec.py | trap.py | fxdpt_2.py | hw3.py

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Wed Feb 19 08:37:09 2020
4
5  @author: cpjohnson
6
7  This program implements a simple version of Newton's Method
8    The input required is x_0 and number of iterations n
9  """
10
11 def f(x):
12     return x**3 - 4*x**2 + 3*x + 1
13
14 def fp(x):
15     return 3*x**2 - 8*x + 3
16
17 def g(x):
18     return x - f(x)/fp(x) # Newton's method
19
20
21 def fxdpt(x_0, n):
22     # x_0 is the initial guess
23     # n is the number of iterations
24     print(x_0)
25     for k in range(1, n):
26         x_0 = g(x_0)
27         print("iteration {}: {}".format(k, x_0))
28
29
```

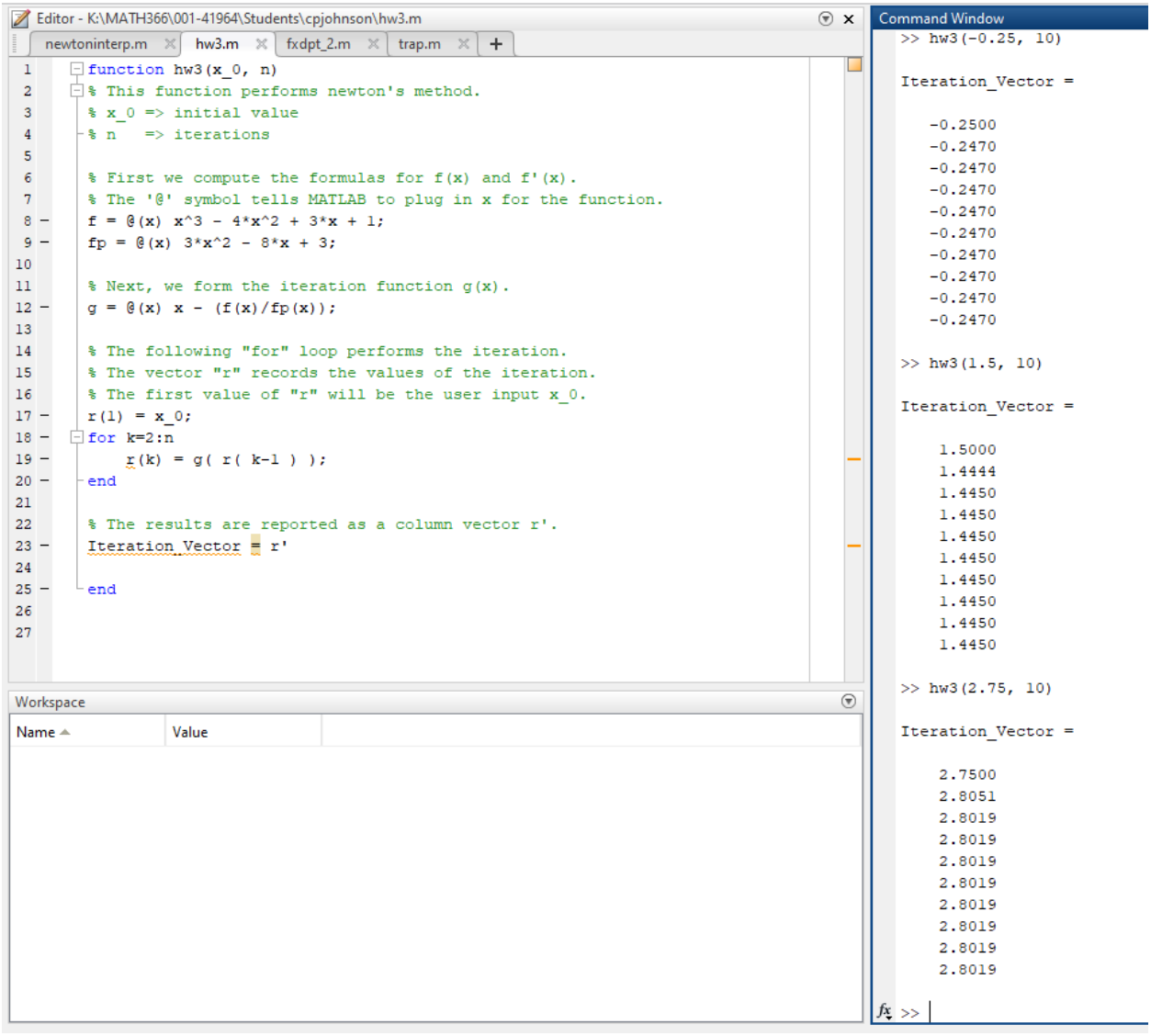
Python Output

```
In [42]: fxdpt(-0.25, 10)
-0.25
iteration 1: -0.2469879518072289
iteration 2: -0.24697960378151157
iteration 3: -0.24697960371746705
iteration 4: -0.24697960371746705
iteration 5: -0.24697960371746705
iteration 6: -0.24697960371746705
iteration 7: -0.24697960371746705
iteration 8: -0.24697960371746705
iteration 9: -0.24697960371746705

In [43]: fxdpt(1.5, 10)
1.5
iteration 1: 1.4444444444444444
iteration 2: 1.4450418160095582
iteration 3: 1.4450418679126282
iteration 4: 1.445041867912629
iteration 5: 1.4450418679126285
iteration 6: 1.445041867912629
iteration 7: 1.4450418679126285
iteration 8: 1.445041867912629
iteration 9: 1.4450418679126285

In [44]: fxdpt(2.75, 10)
2.75
iteration 1: 2.805084745762712
iteration 2: 2.801948227501112
iteration 3: 2.801937735922062
iteration 4: 2.801937735804838
iteration 5: 2.801937735804839
iteration 6: 2.801937735804838
iteration 7: 2.801937735804839
iteration 8: 2.801937735804838
iteration 9: 2.801937735804839
```

We see that Newton's method has done a fairly good job of finding the roots of *f* (double checking with Desmos).

MATLAB

MATLAB Program



MATLAB Code

Editor - K:\MATH366\001-41964\Students\cpjohnson\hw3.m

newtoninterp.m | hw3.m | fxdpt_2.m | trap.m | +

```
function hw3(x_0, n)
% This function performs newton's method.
% x_0 => initial value
% n   => iterations

% First we compute the formulas for f(x) and f'(x).
% The '@' symbol tells MATLAB to plug in x for the function.
f = @(x) x^3 - 4*x^2 + 3*x + 1;
fp = @(x) 3*x^2 - 8*x + 3;

% Next, we form the iteration function g(x).
g = @(x) x - (f(x)/fp(x));

% The following "for" loop performs the iteration.
% The vector "r" records the values of the iteration.
% The first value of "r" will be the user input x_0.
r(1) = x_0;
for k=2:n
    r(k) = g( r( k-1 ) );
end

% The results are reported as a column vector r'.
Iteration_Vector = r'

end
```

MATLAB Output

```
>> hw3(-0.25, 10)

Iteration_Vector =

   -0.2500
   -0.2470
   -0.2470
   -0.2470
   -0.2470
   -0.2470
   -0.2470
   -0.2470
   -0.2470
   -0.2470

>> hw3(1.5, 10)

Iteration_Vector =

    1.5000
    1.4444
    1.4450
    1.4450
    1.4450
    1.4450
    1.4450
    1.4450
    1.4450
    1.4450

>> hw3(2.75, 10)

Iteration_Vector =

    2.7500
    2.8051
    2.8019
    2.8019
    2.8019
    2.8019
    2.8019
    2.8019
    2.8019
    2.8019
```

We see that the output here also matches the output from Desmos.