PHYS 410 Assignment 1
Parker Lloyd
17048142

Problem 1:

For Problem 1 I implemented 3 function files: f, dfdx, and hybrid. The function f(x) takes the argument x and returns the value $y = \mathrm{f(x)} = 128\mathrm{x}^8 - 256\mathrm{x}^6 + 160\mathrm{x}^4 - 32\mathrm{x}^2 + 1$. The function dfdx(x) takes the argument x and returns the derivative at that point. The function hybrid determines the root of the function f between a lower and upper bound (xmin, xmax) using the method of bisection until the solution is within the desired tolerance tol1. Once a solution is found within this first tolerance level, the hybrid function uses this current solution as an initial guess for performing Newton's method. Newton's method is applied until the solution is within a second tolerance level, tol2. I also created a script to iterate over the domain [-1 1] in 0.1 increments ($x_{max} - x_{min} = 0.1$) and call the hybrid function to determine the root in that range if $f_{max}*f_{min} < 0$.

The roots of the equation $f(x) = 128x^8 - 256x^6 + 160x^4 - 32x^2 + 1$ in the domain x ϵ [-1 1] were calculated to be:

x = [-0.980785280403231 -0.831469677945431 -0.555573074559169 -0.195090322016128
0.195090322016128 0.555573074559169 0.831469677945431 0.980785280403231]

using tolerance levels of tol1 = 10^{-2} and tol2 = 10^{-16}. To verify my code, I determined the roots of this equation using Wolfram and concluded that my code solved for the roots to the desired accuracy.

Problem 2:

For Problem 2 I implemented 3 function files: f, jac, and newtond. To define the function f(**x**) I took the system of equations provided by the question and moved all terms to one side of the equation and set them equal to zero. The function f takes a row vector, **x** = [x y z], and returns a column vector comprised of the residuals of each of the 3 new equations. The function jac(**x**) computes the Jacobian of the function f and evaluates it at the point x. The function newtond takes an initial guess, **x**, of the system of equations and computes the residual, f(**x**), and the derivative, jac(**x**) to determine **dx**: the amount **x** will be incremented by to help eliminate the residual and find a solution. This process is repeated with **x** replaced with **x**-**dx** until the root-mean-square value of **dx** falls below a specified tolerance or until 40 iterations are performed. I also created a script to call the newtond function with the required arguments.

The solution to the system of equations

$$x^2 + y^3 + z^4 = 1$$
$$sin(xyz) = x + y + z$$
$$x = yz$$

was calculated to be **x** = [x y z] = [2.31551841888053 -1.88117377526825 -1.23089022892122] using a tolerance of 10^{-16}. To verify, I determined the solution for this system of equations using Wolfram and concluded that my code found the solution to the desired accuracy.