Feyzi Can Eser – Brief Explanation

For both methods I used a while loop to do at most MAX ITERATION number of steps. Inside the while loops:

- For Newtons I put an if statement after calculating the Jacobian inverse's denominator term first so that the program may exit if the Jacobian is singular
- I use variables to house f and g for Newtons so the code is easier to read
- For Fixed point Iteration I first check if any of the square roots are < 0 in an if statement
- For both methods I store the previous x and y values in variables so I can later check if the tolerance condition is met with another if statement
- If the while loop concludes without the function returning early then it exits with status 1 since max number of iterations has been met

Why the code works

- The code works because the erroneous cases of singular Jacobian and negative square roots are caught early on in the while loop with the if statements
- The mathematical equations are translated directly into the code properly, so solutions are found when the methods are indeed able to find solutions
- For newtons, the solution found depends on the given x and y values such that x = 0.866 when x_0 >0 and x = -0.866 when x<0, y=0.5 when y_0 >0 and y=-0.5 when y_0 <0
- For fixed points we naturally only get to the solution with x = 0.866 and y=0.5 so long as the inside of the square roots aren’t negative

Reflection

- In the first submission I had written a part of the equation for g incorrectly for newtons, and forgot to use the previous x and y values when doing the fixed point iteration stepping.
- I felt the most challenging aspect was correctly translating the equations to code and avoiding minor typing errors I have used c before so was mostly comfortable with it.