NumComp - Fall 2022
Project #6

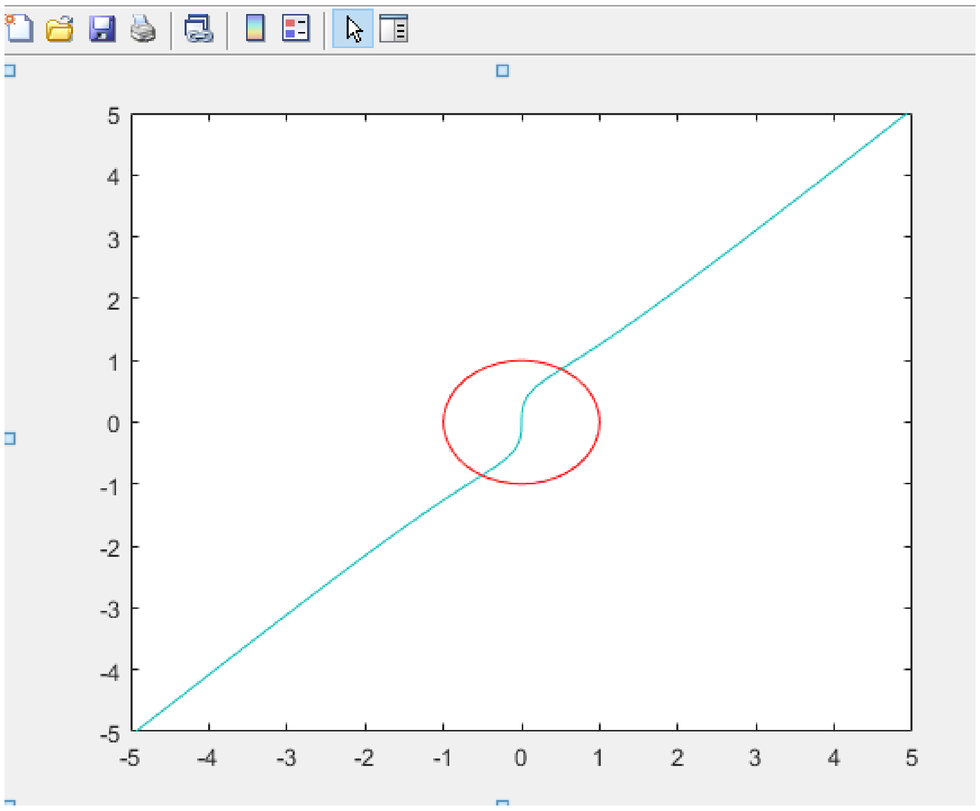
Due –

Consider the following nonlinear system of equations with two equations and two unknowns. The math problem can be stated as follows. Given $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$ defined as

$$\begin{aligned} f_1(x_1, x_2) &= x_1^3 - x_2^3 + x_1, \\ f_2(x_1, x_2) &= x_1^2 + x_2^2 - 1. \end{aligned} \tag{1}$$

Find r_1 and r_2 such that $f_1(r_1, r_2) = 0$ and $f_2(r_1, r_2) = 0$.

1. Make a plot that shows
   - all the points that satisfy $f_1 = 0$ and
   - all the points that satisfy $f_2 = 0$
   - the points that satisfy both

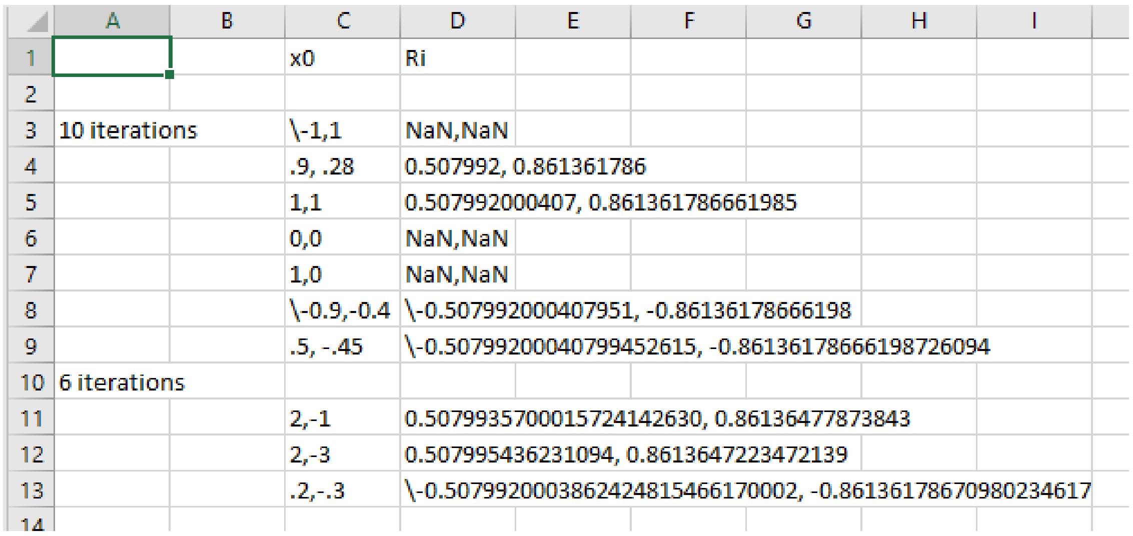


2. Calculate the jacobian 2x2 by hand

$$f(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} = \begin{bmatrix} x_1^3 - x_2^3 + x_1 \\ x_1^2 + x_2^2 - 1 \end{bmatrix}$$

$$J_f(x_1, x_2) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 3x_1^2 + 1 & -3x_2^2 \\ \\ 2x_1 & 2x_2 \end{bmatrix}$$

3. Use Newton's method for systems to find the two solutions to the system of equations ($f_1 = 0$, $f_2 = 0$).



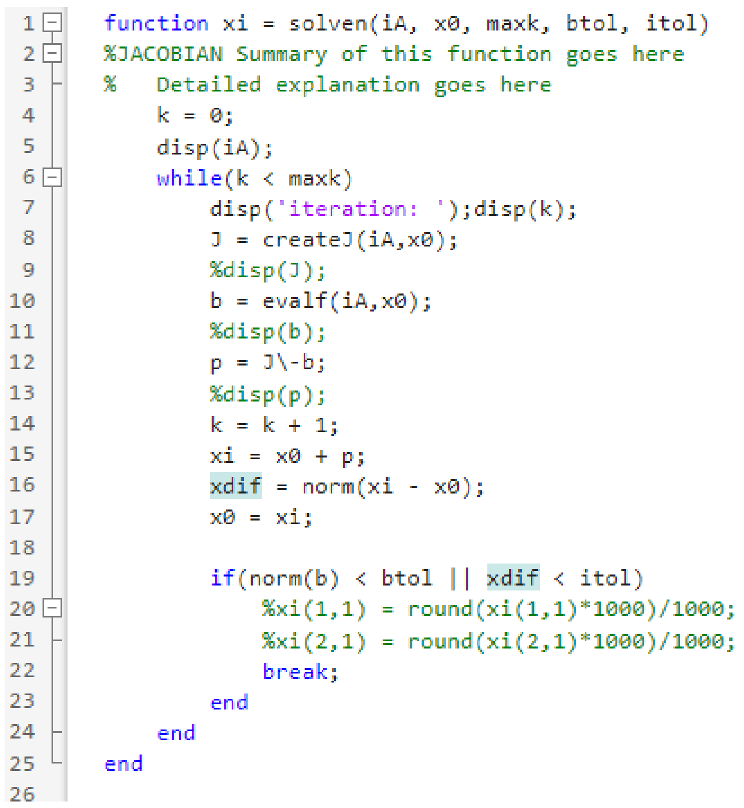
	A	B	C	D
1			x0	Ri
2				
3	10 iterations		\-1,1	NaN,NaN
4			.9, .28	0.507992, 0.861361786
5			1,1	0.507992000407, 0.861361786661985
6			0,0	NaN,NaN
7			1,0	NaN,NaN
8			\-0.9,-0.4	\-0.507992000407951, -0.86136178666198
9			.5, -.45	\-0.50799200040799452615, -0.86136178666198726094
10	6 iterations			
11			2,-1	0.50799357000157241426 30, 0.86136477873843
12			2,-3	0.507995436231094, 0.8613647223472139
13			.2,-.3	\-0.50799200038624248154661700 02, -0.86136178670980234617
14				

4. Find a starting point where Newton's method fails. Why did it fail?

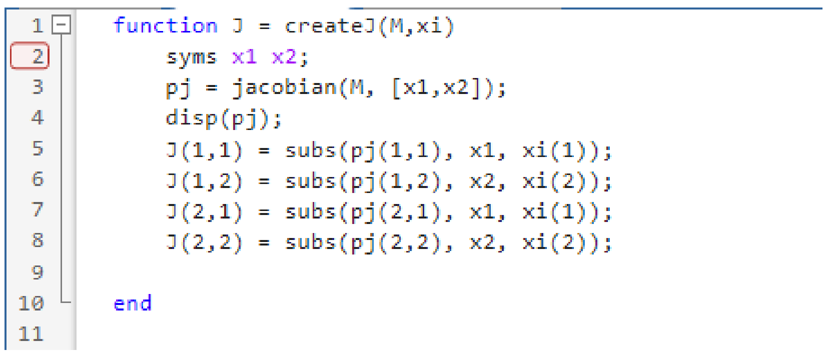
All failures are at non differentiable points of either function. Seems convergence depends on differentiation. As x_0 grows and gets close to these non differentiable points, the more time and iterations are needed to solve the system accurately.

The one thing I could not figure out is if my code is even remotely optimized– Each operation operates on each number as a fraction rather than a decimal, which is likely making my code take forever. I needed to convert each result into a less precise float in order to make it readable. Non-rounded outputs of my code would be fractions larger than the max line of the CLI. Regardless, the precision of my code is still pretty good despite being slow.

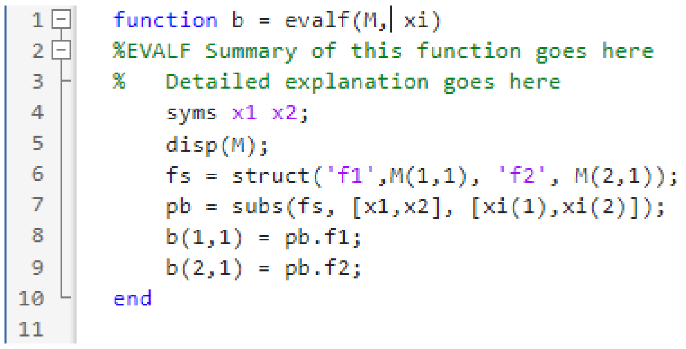
Code follows!



```
function xi = solven(iA, x0, maxk, btol, itol)
%JACOBIAN Summary of this function goes here
%   Detailed explanation goes here
    k = 0;
    disp(iA);
    while(k < maxk)
        disp('iteration: ');disp(k);
        J = createJ(iA,x0);
        %disp(J);
        b = evalf(iA,x0);
        %disp(b);
        p = J\-b;
        %disp(p);
        k = k + 1;
        xi = x0 + p;
        xdif = norm(xi - x0);
        x0 = xi;

        if(norm(b) < btol || xdif < itol)
            %xi(1,1) = round(xi(1,1)*1000)/1000;
            %xi(2,1) = round(xi(2,1)*1000)/1000;
            break;
        end
    end
end

```



```
function J = createJ(M,xi)
    syms x1 x2;
    pj = jacobian(M, [x1,x2]);
    disp(pj);
    J(1,1) = subs(pj(1,1), x1, xi(1));
    J(1,2) = subs(pj(1,2), x2, xi(2));
    J(2,1) = subs(pj(2,1), x1, xi(1));
    J(2,2) = subs(pj(2,2), x2, xi(2));

end

```



```
function b = evalf(M, xi)
%EVALF Summary of this function goes here
%   Detailed explanation goes here
    syms x1 x2;
    disp(M);
    fs = struct('f1',M(1,1), 'f2', M(2,1));
    pb = subs(fs, [x1,x2], [xi(1),xi(2)]);
    b(1,1) = pb.f1;
    b(2,1) = pb.f2;
end

```