Hey paul, was having trouble organizing the publisher function
with matlab, extra credit problems 5 and 6 printed out
after my Newtons function and problem two is at the end.
Thanks!

Table of Contents

Housekeeping

```
clear all; close all; clc;

%{
    CSCI 3656 HW6
    Author: Connor O'Reilly
    Email: coor1752@colorado.edu
    Last Edited: 10/14/2021
%}
```

Problem One:

```
%define functions
syms x [1 2]
f(1,1) = x(1).^3 - x(2).^3 + x(1);
f(2,1) = x(1).^2 + x(2).^2 - 1;
%cell array of functions
```

Problem 2

```
%included in pdf
```

Problem 3

```
%test different initial guesses to find a soluton

%this will probably take awhile
[r1, i(1)] = newtons_nonlin(f , x, [0 0], 1e-9, 100);
[r2, i(2)] = newtons_nonlin(f , x, [0 1], 1e-9, 100);
[r3, i(3)] = newtons_nonlin(f , x, [1 0], 1e-9, 100);
[r4, i(4)] = newtons_nonlin(f , x, [1 1], 1e-9, 100);
[r5, i(5)] = newtons_nonlin(f , x, [0.5 0], 1e-9, 100);
[r6, i(6)] = newtons_nonlin(f , x, [0 0.5], 1e-9, 100);
[r7, i(7)] = newtons_nonlin(f , x, [0.5 0.5], 1e-9, 100);
[r8, i(8)] = newtons_nonlin(f , x, [0.8 0.1], 1e-9, 100);
[r9, i(9)] = newtons_nonlin(f , x, [0.1 0.8], 1e-9, 100);
```

```
[r10, i(10)] = newtons_nonlin(f , x, [0.4 0.5], 1e-9, 100);
[r11, i(11)] = newtons_nonlin(f , x, [0.5 0.4], 1e-9, 100);
[r12, i(12)] = newtons_nonlin(f , x, [0.3 0.2], 1e-9, 100);
[r13, i(13)] = newtons_nonlin(f , x, [0.9 -0.9], 1e-9, 100);
[r14, i(14)] = newtons_nonlin(f , x, [-0.9 0.9], 1e-9, 100);
[r15, i(15)] = newtons_nonlin(f , x, [1.1 0.5], 1e-9, 100);

%display in plotting section
```

Problem 4

```
%starting point where newtons method failed was found with test points

%compute jacobian
jacob_4 = jacobian(f,x);
jacob_4 = double(subs(jacob_4, x, [1,0]));
det_p4 = det(jacob_4);
```

Problem 5

```
%CPU go brrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
X1_5 = linspace(-1, 1, 30);
X2_5 = X1_5;
[X1_5, X2_5] = meshgrid(X1_5,X2_5);

%question is answered in plotting section
x1_5 = [0 0 1 1 0.5 0 0.5 0.5 0.1 0.4 0.5 0.3 0.9 -0.9 1.1];
x2_5 = [0 1 0 1 0 0.5 0.5 0.1 0.8 0.5 0.4 0.2 -0.9 0.9 0.5];
```

Problem 6

```
X1_6 = linspace(-1, 1, 10);
X2_6 = X1_6;
[X1_6, X2_6] = meshgrid(X1_6, X2_6);


%jacobian computed for plotting
jacob = jacobian(f,x);

%plotting taken care of below
```

Plotting

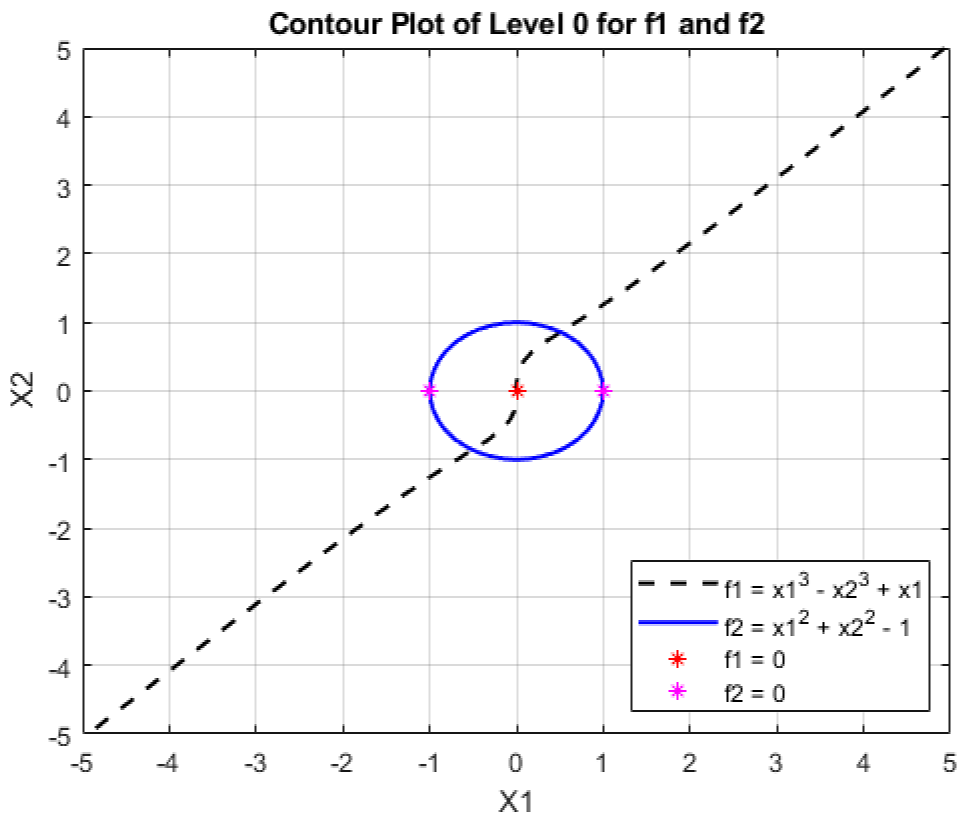
```
% Problem 1 parts (i) and (ii)
figure(1)
%level one gets plot similar to that provided by paul in class.
hand_f1 = fcontour(f(1),'k--', 'LineWidth', 1.5);
hand_f1.LevelList = 0;
hold on;
hand_f2 = fcontour(f(2), 'b', 'LineWidth', 1.5);
hand_f2.LevelList = 0;
```

```
grid on;
xlabel('X1')
ylabel('X2');
title('Contour Plot of Level 0 for f1 and f2');
%plot where f1 is equal to zero, easy when x1 and x2 are equal to zero
scatter(0,0,'r*')
%plot where f2 = 0, rewrite equation f2 so that x1^2 + x2 - 1 = 0 ->
% x1^2 + x2^2 = 1, x1 = +- 1 and x2 = 0 or x1 = 0 and x2 = +-1
scatter(1,0,'m*');
scatter(-1,0, 'm*');
legend('f1 = x1^3 - x2^3 + x1', 'f2 = x1^2 + x2^2 - 1', 'f1 = 0','f2 =
 0', 'Location', 'southeast');
hold off;
snapnow

fprintf('\n-----------------------------------------------------------------------
\n')
fprintf('Problem 1 parts (i) and (ii):')
fprintf('\n-----------------------------------------------------------------------
\n')
fprintf('\n Points that satisfy f1 = 0:\n x1 = 0, x2 = 0 \n \n Points
 which satisfy f2 = 0:\n x1 = 0 , x2 = +- 1 \n x1 = +- 1 , x2 = 0 \n
\n');
fprintf('Looking at the graph, there are no points where f1 and f2
 share a root.\n\n');

%problem 3
fprintf('\n-----------------------------------------------------------------------
\n')
fprintf('Problem 3:')
fprintf('\n-----------------------------------------------------------------------
\n\n')
fprintf('Implementing Newton''s Method for systems several intial
 guesses were used to determine points that satisfy f1 = 0 and f2 =
 0\n\n')
fprintf('Initial Guess ( x1, x2)  |  Newton''s answer ( r1 , r2)\n')
fprintf('--------------------------------------------------------------
\n')
fprintf('            (0 , 0)              |            (%0.4f , %0.4f)
     \n',r1)
fprintf('            (0 , 1)              |            (%0.4f , %0.4f)
     \n',r2)
fprintf('            (1 , 0)              |            (%0.4f , %0.4f)
     \n',r3)
fprintf('            (1 , 1)              |            (%0.4f , %0.4f)
     \n',r4)
fprintf('            (0.5 , 0)            |            (%0.4f , %0.4f)
    \n',r5)
fprintf('            (0 , 0.5)            |            (%0.4f , %0.4f)
    \n',r6)
fprintf('            (0.5 , 0.5)          |            (%0.4f , %0.4f)
   \n',r7)
fprintf('            (0.8 , 0.1)          |            (%0.4f , %0.4f)
   \n',r8)
```

```
fprintf('              (0.1 , 0.8)         |             (%0.4f , %0.4f)
            \n',r9)
fprintf('              (0.4 , 0.5)         |             (%0.4f , %0.4f)
            \n',r10)
fprintf('              (0.5 , 0.4)         |             (%0.4f , %0.4f)
            \n',r11)
fprintf('              (0.3 , 0.2)         |             (%0.4f , %0.4f)
            \n',r12)
fprintf('              (0.9 , -0.9)        |             (%0.4f , %0.4f)
            \n', r13)
fprintf('              (-0.9 , 0.9)        |             (%0.4f , %0.4f)
            \n', r14)
fprintf('              (-0.9 , -0.9)       |             (%0.4f , %0.4f)
            \n', r15)
fprintf('Solutions that do converge are converging to either  x1 =
 0.5080  x2 =  0.8614 or x1 = -0.5080  x2 =  -0.8614, which seems to
 be where\n f1 = f2.\n')
fprintf('Evaluating f1 and f2 at either return the following results:
\n')
fprintf('f1(0.5080, 0.8614) = %0.5f \n', double( subs(f(1), x,
 [0.5080, 0.8614]) ))
fprintf('f2(0.5080, 0.8614) = %0.5f \n', double( subs(f(2), x,
 [0.5080, 0.8614]) ))

fprintf('f1(-0.5080, -0.8614) = %0.5f \n', double( subs(f(1), x,
 [-0.5080, -0.8614]) ))
fprintf('f2(-0.5080, -0.8614) = %0.5f \n', double( subs(f(2), x,
 [-0.5080, -0.8614]) ))
fprintf('The above evaluations provide semiaccurate approximations for
 the roots r1 and r2 such that f1(r1,r2) = f2(r1,r2) = 0\n')
% problem 4
fprintf('\n-----------------------------------------------------------------------
\n')
fprintf('Problem 4:')
fprintf('\n-----------------------------------------------------------------------
\n\n')
fprintf('Jacobian using value of x1 = 1 and x2 = 0\n\n')
disp(jacob_4)
fprintf('\nDeterminant of Jacobian: %0.4f\n',det_p4);
fprintf('Determinant of jacobian is singular which is equivelant to
 the case for newtons method with a differentiable functin f(x)'' =
 0\n meaning the inverse does not exist. Causing the approximation
 to fail due to pi = -J(xi)^-1/f(xi) to not exist or be inaccurate.
 \nThese fails can be seen in the previous problem where the
 approximated root is evaluated to (NaN, Nan)\n')

%problem 5
fprintf('\n-----------------------------------------------------------------------
\n')
fprintf('Problem 5:')
fprintf('\n-----------------------------------------------------------------------
\n\n')
figure(2)
hold on;
```

```
for i = 1:30
    for j = 1:30
      [~ , it] = newtons_nonlin(f , x, [X1_5(i,j) X2_5(i,j)], 1e-9,
 100);

      if ( i < 15 )
          col = 'k';
      else
          col = 'm';
      end
      scatter3(X1_5(i,j) ,X2_5(i,j), it, col)
    end
end
%hopefully my machine crashes
grid on;
title('Iterates from Newton''s method in (x1, x2) space');
xlabel('X1')
ylabel('X2')
zlabel('Iterations i')
view([-5 -2 5])
hold off;
snapnow

 fprintf('\n Observing figure(2), it seems that as initial guess of X2
 approaches zero the number of iterations needed for newtons method to
 converge increases, \nmaybe im making this up at this point but as X2
 approaches zero teh determinant of the jacobian also approaches zero
 causing the method to decrease in accuracy')
%
% Problem 6

figure(3)
title('My sad attempt of creating a quiver plot of the Newton
 Directions');
hold on;
xlabel('x')
ylabel('f(x1, x2)');
grid on;
axis equal;
for i = 1:10
    for j = 1:10
      f_eval = double( subs(f, x, [X1_6(i,j), X2_6(i,j)]) );
      j_eval = double( subs(jacob, x, [X1_6(i,j) , X2_6(i,j)] ) );
      p_mesh = -j_eval \ f_eval;
      quiver(X1_6(i,j), X2_6(i,j) , p_mesh(1) , p_mesh(2))
      hold on
    end
end


%add f1 and f2 cause why not yolo
hand_f1 = fcontour(f(1),'k--', 'LineWidth', 1.5);
hand_f1.LevelList = 0;
hand_f2 = fcontour(f(2), 'b', 'LineWidth', 1.5);
```

```
hand_f2.LevelList = 0;

hold off;
snapnow

fprintf('This is another shot in the dark, but im assuming arrows show
 the magnitude in change of the next iteration for newtons method,\n
 similar to problem 5 arrows closer to the X2 axis where are larger
 in magnitude and reach far beyound any of the functions in the
 X1,X2 plane \n which is then reflected in the method becoming more
 inefficient and inaccurate\n');
```




```
--------------------------------------------------------------------------------
Problem 1 parts (i) and (ii):
--------------------------------------------------------------------------------

 Points that satisfy f1 = 0:
 x1 = 0, x2 = 0

 Points which satisfy f2 = 0:
 x1 = 0 , x2 = +- 1
 x1 = +- 1 , x2 = 0

Looking at the graph, there are no points where f1 and f2 share a
 root.
```

Problem 3:

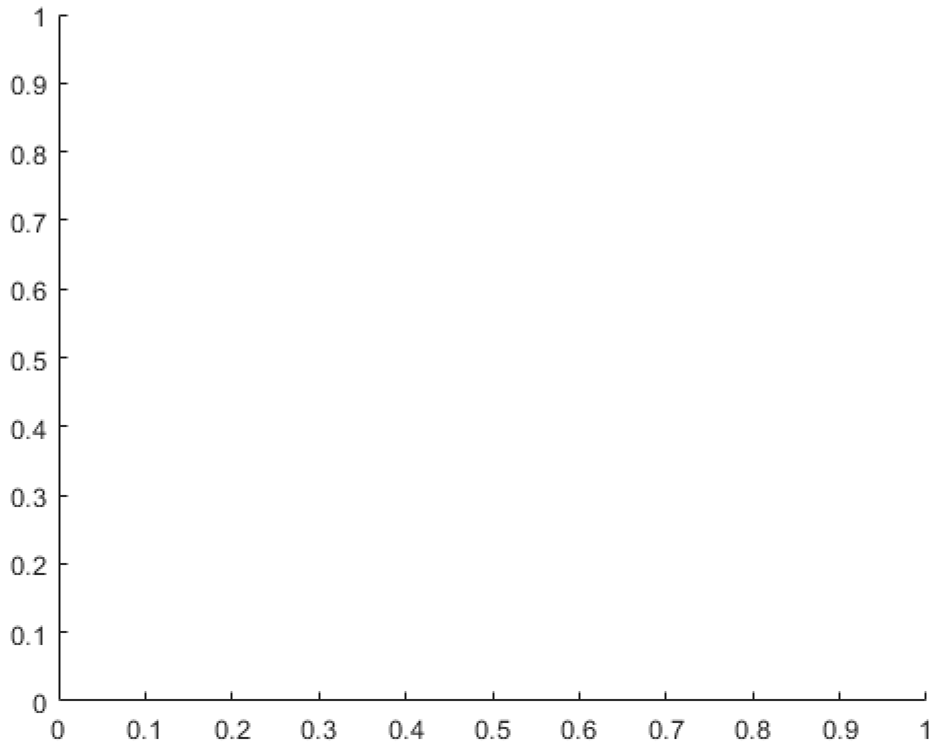
Implementing Newton's Method for systems several intial guesses were used to determine points that satisfy f1 = 0 and f2 = 0

Initial Guess (x1, x2)	Newton's answer (r1 , r2)
(0 , 0)	(NaN , NaN)
(0 , 1)	(0.5080 , 0.8614)
(1 , 0)	(NaN , NaN)
(1 , 1)	(0.5080 , 0.8614)
(0.5 , 0)	(NaN , NaN)
(0 , 0.5)	(0.5080 , 0.8614)
(0.5 , 0.5)	(0.5080 , 0.8614)
(0.8 , 0.1)	(0.5080 , 0.8614)
(0.1 , 0.8)	(0.5080 , 0.8614)
(0.4 , 0.5)	(0.5080 , 0.8614)
(0.5 , 0.4)	(0.5080 , 0.8614)
(0.3 , 0.2)	(-0.5080 , -0.8614)
(0.9 , -0.9)	(-0.5080 , -0.8614)
(-0.9 , 0.9)	(0.5080 , 0.8614)
(-0.9 , -0.9)	(0.5080 , 0.8614)

Solutions that do converge are converging to either x1 = 0.5080 x2 = 0.8614 or x1 = -0.5080 x2 = -0.8614, which seems to be where f1 = f2.
Evaluating f1 and f2 at either return the following results:
f1(0.5080, 0.8614) = -0.00007
f2(0.5080, 0.8614) = 0.00007
f1(-0.5080, -0.8614) = 0.00007
f2(-0.5080, -0.8614) = 0.00007
The above evaluations provide semiaccurate approximations for the roots r1 and r2 such that f1(r1,r2) = f2(r1,r2) = 0

Problem 4:

```
Jacobian using value of x1 = 1 and x2 = 0

     4     0
     2     0


Determinant of Jacobian: 0.0000
Determinant of jacobian is singular which is equivelant to the case
 for newtons method with a differentiable functin f(x)' = 0
 meaning the inverse does not exist. Causing the approximation to fail
 due to pi = -J(xi)^-1/f(xi) to not exist or be inaccurate.
These fails can be seen in the previous problem where the approximated
 root is evaluated to (NaN, Nan)

----------------------------------------------------------------------------------
Problem 5:
----------------------------------------------------------------------------------
```

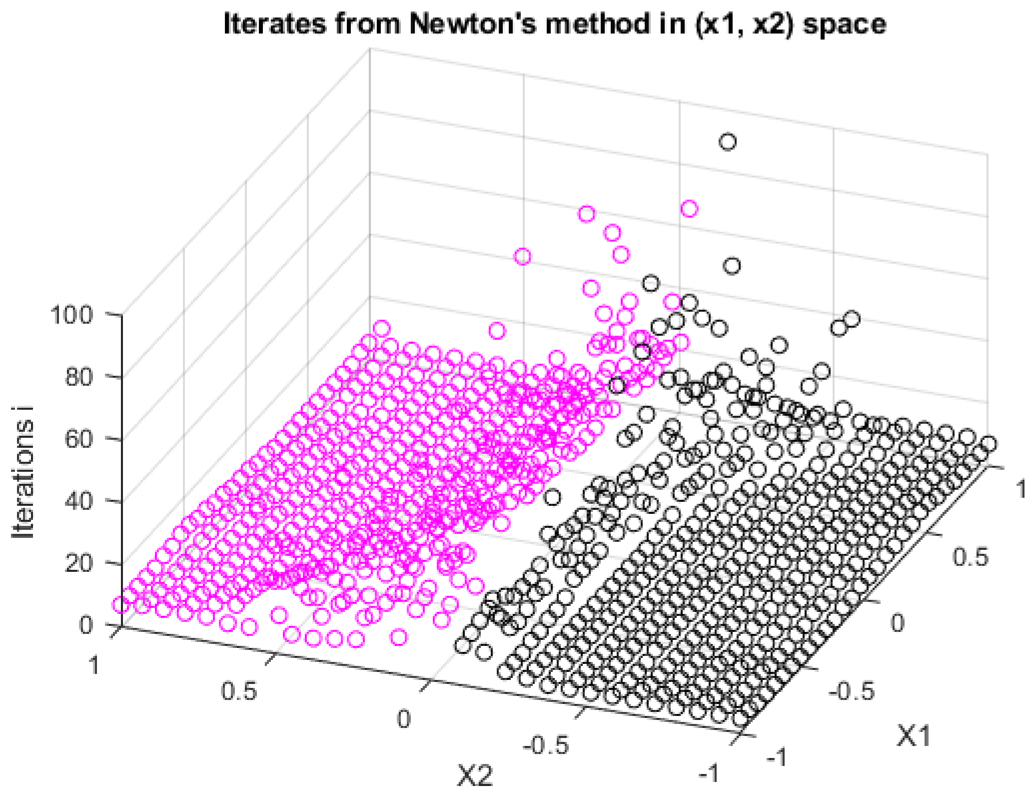


Fucntions

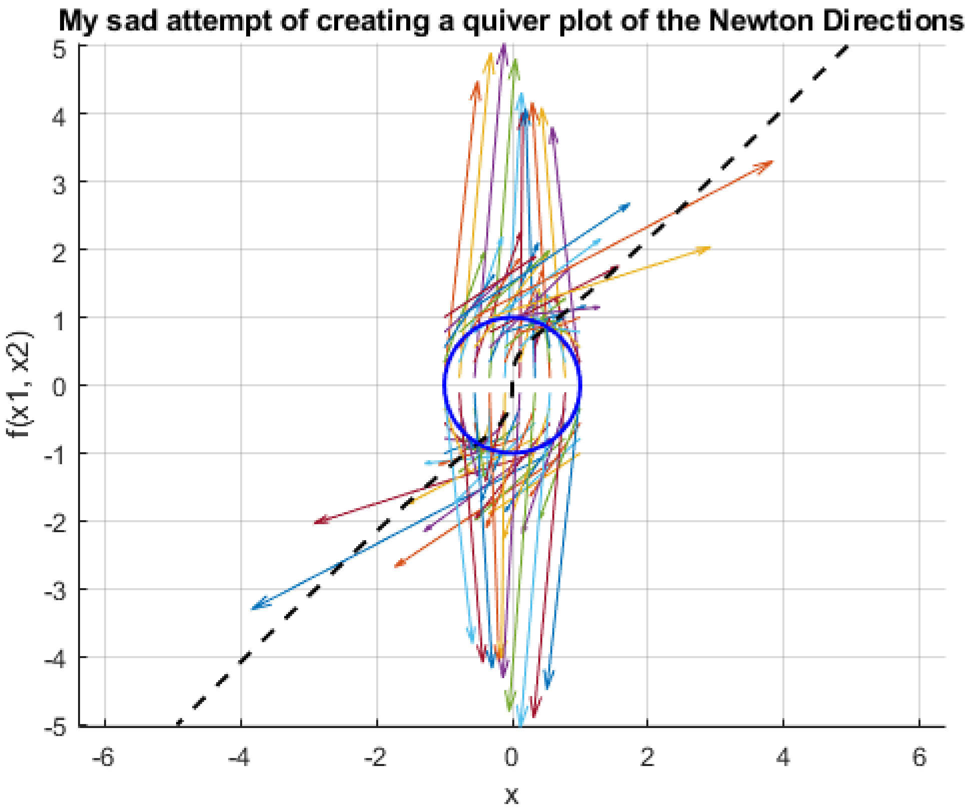
how to evaluate array of functions subs(f,[x(1),x(2)],[0 ,0])

```
function [r,i] = newtons_nonlin(f, vars, r0, tol, maxit)
%{
```

```
    Purpose: MATLAB implementation of Newton's Method for non linear
    systems.

    Inputs:
        f: array of symbolic functions (column, i dont think it
 matters but im too tired to care oops)
        vars: array of symbolic scalar variables
        r0: vector of initial guesses
        tol: tolerance
        maxit: maximum iteration count

    Outputs:
        r: array of approximation values for root of functions
        i: number of iterations
%}
%get jacobian with f and x
jacob = jacobian(f, vars);
x0 = r0; %x0
for i = 0:maxit
    %p0 = -J(x0) \ f(x0)
    b = subs(f,vars, x0);
    A = subs(jacob, vars, x0);
    p0 = -double(A) \ double(b);
    x0 = x0.';
    x1 = x0 + p0;

    %stopping criteria
    if ( norm(x1 - x0) < tol ) || ( norm(x1) < tol )
        break;
    else
        %swap and repeat
        x0 = x1.';
    end
end

r = x0.';
end
```



Observing figure(2), it seems that as initial guess of X2 approaches zero the number of iterations needed for newtons method to converge increases,
maybe im making this up at this point but as X2 approaches zero teh determinant of the jacobian also approaches zero causing the method to decrease in accuracy




```
This is another shot in the dark, but im assuming arrows show the
 magnitude in change of the next iteration for newtons method,
 similar to problem 5 arrows closer to the X2 axis where are larger
 in magnitude and reach far beyound any of the functions in the X1,X2
 plane
 which is then reflected in the method becoming more inefficient and
 inaccurate
```

Published with MATLAB® R2021a

Homework 6

Friday, October 15, 2021 1:40 PM

Problem 2

$$f = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix}$$

$$= \begin{bmatrix} x_1^3 - x_2^3 + x_1 \\ x_1^2 + x_2^2 - 1 \end{bmatrix}$$

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}$$

$$\frac{\partial f_1}{\partial x_1} = 3x_1^2 + 1$$

$$\frac{\partial f_1}{\partial x_2} = -3x_2^2$$

$$\frac{\partial f_2}{\partial x_1} = 2x_1 - 1$$

$$\frac{\partial f_2}{\partial x_2} = 2x_2$$

$$\boxed{J_f = \begin{bmatrix} 3x_1^2 + 1 \ , & -3x_2^2 \\ 2x_1 - 1 \ , & 2x_2 \end{bmatrix}}$$