MATH 446
Project 4

March 28, 2023

Q1. We rewrite the given equations as

$$f_1(x,y) = \frac{x^2}{9} + \frac{xy}{3} + y^2 - 1 = 0$$
$$f_2(x,y) = x^2 - y - 1 = 0$$

Thus,

$$F(x,y) = \begin{bmatrix} \frac{x^2}{9} + \frac{xy}{3} + y^2 - 1 \\ x^2 - y - 1 \end{bmatrix}$$

Q2. The Jacobian Matrix is

$$DF(x,y) = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix}$$

which is given by

$$DF(x,y) = \begin{bmatrix} \frac{2x}{9} + \frac{y}{3} & \frac{x}{3} + 2y \\ 2x & -1 \end{bmatrix}$$

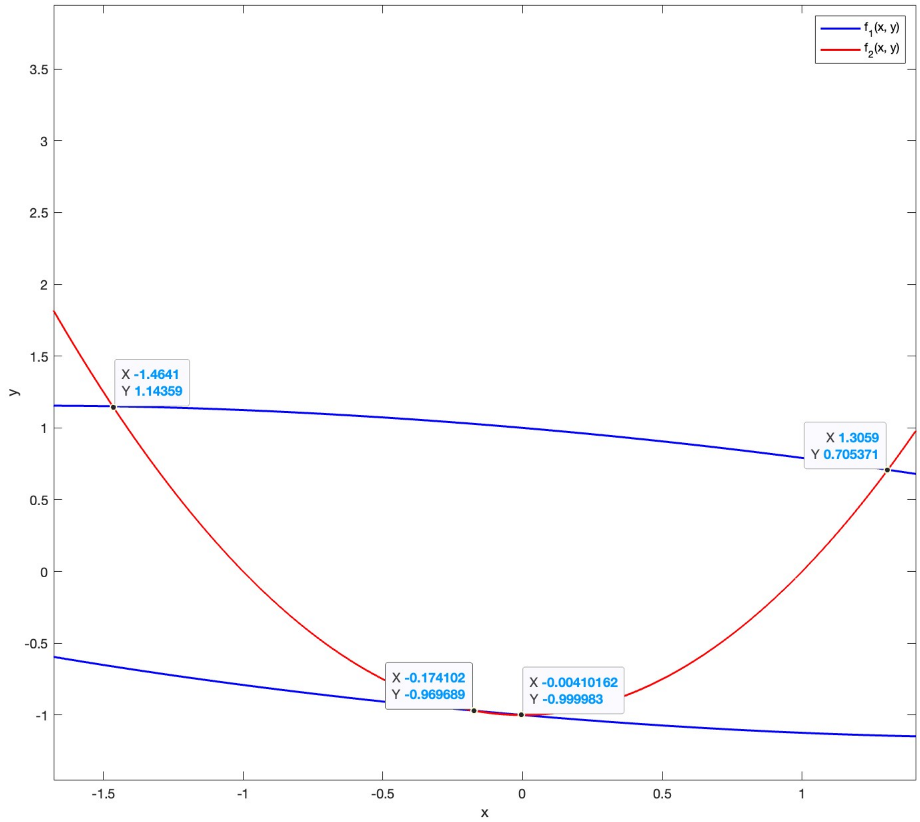
Q3. Using Newton's method implemented MATLAB, the intersection points are computed. The results are shown below. A tolerance value of 1e-12 was chosen to check for convergence. The initial guesses used are listed in Q5 below.

```
Command Window
Intersection Point #1 = (1.3070, 0.7083)
F(1.3070, 0.7083) = (-0.00000000, 0.00000000)

Intersection Point #2 = (-1.4664, 1.1504)
F(-1.4664, 1.1504) = (-0.00000000, 0.00000000)

Intersection Point #3 = (-0.0000, -1.0000)
F(-0.0000, -1.0000) = (0.00000000, 0.00000000)

Intersection Point #4 = (-0.1739, -0.9698)
F(-0.1739, -0.9698) = (0.00000000, 0.00000000)

>>
```

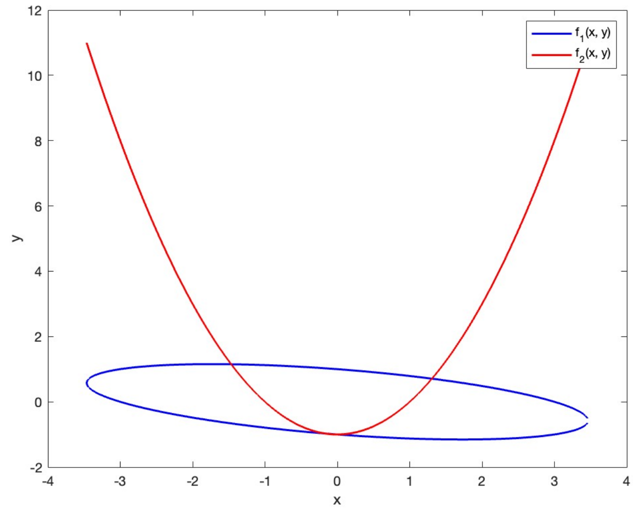
The intersection points are tabulated below.

#	x	y
1	1.3070	0.7083
2	-1.4664	1.1504
3	-0.0000	-1.0000
4	-0.1739	-0.9698

Q4. The graphical solution is shown below for comparison with the computed solution.



The zoomed out plot is shown below.



Q5. The initial guesses used to compute the intersection points are tabulated below, respectively in the order of obtained solutions.

#	x	y
1	0.2	0.3
2	-1.0	1.0
3	2.5	-1.0
4	-3	-3

The MATLAB Code follows.

MATLAB Code - proj_4.m

```
clc; clear; close all

% define Function
F = @(x, y) [x.^2/9 + x.*y/3 + y.^2 - 1 ;
x.^2 - y - 1];

% define Jacobian
DF = @(x, y) [2*x/9 + y/3 , x/3 + 2*y ;
2*x , -1      ] ;

% initial guesses
g1 = [0.2 ; 0.3];
g2 = [-1  ;   1];
g3 = [2.5 ;  -1];
g4 = [-3  ;  -3];
init_guesses = [g1 , g2 , g3 , g4];

% find intersection points using Newton's Method
tol = 1e-12;
for i = 1:length(init_guesses)

old_guess = init_guesses(:, i);
new_guess = old_guess;
while (true)
% compute new guess
new_guess = old_guess - DF(old_guess(1), old_guess(2))\ ...
F(old_guess(1), old_guess(2));
```

```
% check for convergence
if (norm(new_guess - old_guess) < tol)
break
end

% update old guess for next iteration
old_guess = new_guess;
end

fprintf('Intersection Point #%d = (%.4f, %.4f)\n', i, ...
new_guess(1), new_guess(2))

% post computation check
fprintf('F(%.4f, %.4f) = (%.8f, %.8f)\n\n', new_guess(1), new_guess(2), ...
F(new_guess(1), new_guess(2)))
end


% plot both curves
x  = -sqrt(12) : 0.01 : sqrt(12);
% ellipse
y1 = sqrt(1 - x.^2/12) - x/6;
y2 = -sqrt(1 - x.^2/12) - x/6;
% parabola
y3 = x.^2 - 1;

plot(x, y1, 'b', 'linewidth', 1.5)
hold on
plot(x, y2, 'b', 'linewidth', 1.5)
plot(x, y3, 'r', 'linewidth', 1.5)
xlabel('x')
ylabel('y')
legend('f_1(x, y)', '', 'f_2(x, y)')
```