Numerical Computing :: Project Six

Julia Troni

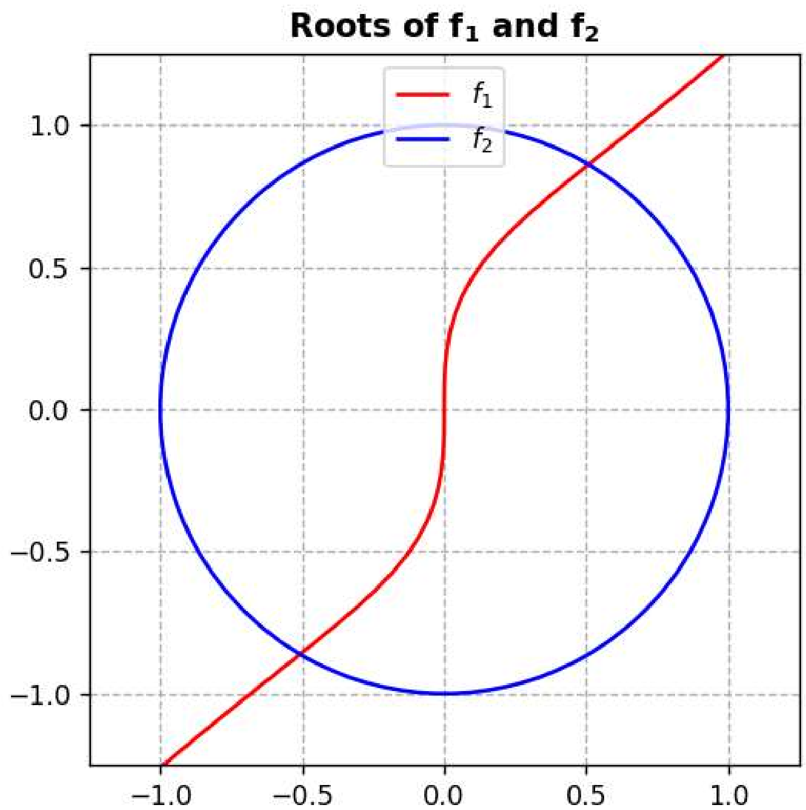
In [1]:

```
%matplotlib notebook
from matplotlib import pyplot
import matplotlib.pyplot as plt
import numpy as np
import math
```

1. Make a plot that shows (i) all the points that satisfy f1 = 0 and (ii) all the points that satisfy f2 = 0. Identify the points on the plot that satisfy both f1 = 0 and f2 = 0.

In [2]:

```
x = np.linspace(-3, 3, 100)
y = np.linspace(-3, 3, 100)

X, Y = np.meshgrid(x,y)

F = X**3 - Y**3 + X
F2 = X**2 + Y**2 - 1.0

fig, ax = plt.subplots()

CS1 = ax.contour(X,Y,F,[0], colors=['red'])
CS2 = ax.contour(X,Y,F2,[0], colors=['blue'])

ax.set_aspect(1)


plt.title('$\\bf{Roots\ of\ f_1\ and\ f_2}$', fontsize=12)
plt.xlim(-1.25,1.25)
plt.ylim(-1.25,1.25)
plt.grid(linestyle='--')

# adding labels to the graph was surprisingly obnoxious
labels = ['$f_1$', '$f_2$']
for i in range(len(labels)):
    if i == 0:
        CS1.collections[i].set_label(labels[i])
    else:
        CS2.collections[0].set_label(labels[i])
plt.legend(loc='upper center')

#labeling and finding intersections and roots proved more frusturating than I anticipat
#I opted to call this good enough and move on to the implementation and com back to try


plt.show()
```



- x,y coords should of intersection: (0.508,0.061) and (-0.508,-0.061)

- the roots of f_1 are x=1 and x=-1.

- the root of f_2 is x=0

2. By hand, calculate the 2 × 2 Jacobian matrix of the system (f1, f2)

$$J(x_1, x_2) = \frac{\partial(f_1,f_2)}{\partial(x_1,x_2)} = \begin{vmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} \end{vmatrix}$$

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

3. Use Newton's method for systems to find the two solutions to the sys tem of equations (f1 = 0, f2 = 0). Try several (10 or so) different initial guesses. Make a table of the answer that Newton's method gives—something like

In [3]:

```
import numpy as np

def Newton_system(F, J, x, eps):
    """
    Solve nonlinear system F=0 by Newton's method.
```

```
    J is the Jacobian of F. Both F and J must be functions of x.
    At input, x holds the start value. The iteration continues
    until ||F|| < eps.
    """
    F_value = F(x)
    F_norm = np.linalg.norm(F_value, ord=5)  # l2 norm of vector
    iteration_counter = 0
    print("initial guess", x)
    print("initial fval", F_value)
    print("initial fnorm", F_norm)
    while abs(F_norm) > eps and iteration_counter < 100:
        delta = np.linalg.solve(J(x), F_value)
        x = x + delta
        F_value = F(x)
        F_norm = np.linalg.norm(F_value, ord=2)
        iteration_counter += 1
        print("iter count: " , iteration_counter)
        print("x", x)
        print("fval", F_value)
        print ("x,itercount  " , x,iteration_counter)
    # Here, either a solution is found, or too many iterations

        # Here, either a solution is found, or too many iterations
        ##code is always stopping here
    if abs(F_norm) > eps:
        iteration_counter = -1
        print("returning ")
        return "Does not converge try a new initial guess", x, iteration_counter

    return "converges final answer" ,x,iteration_counter
```

In [4]:

```
def F(x):
    return np.array(
        [x[0]**3 - x[1]**3 + x[0],
         x[0]**2 + x[1]**2 -1 ])

def J(x):
    return np.array(
        [[3*x[0]**2+1, 3*x[1]**2],
         [2*x[0], 2*x[1] ]])

Newton_system(F, J, x=np.array([.8,.8]), eps=0.0001)
```

```
initial guess [0.8 0.8]
initial fval [0.8  0.28]
initial fnorm 0.8008385900728692
iter count:  1
x [1.264 0.511]
fval [3.15005491 0.858817  ]
x,itercount   [1.264 0.511] 1
iter count:  2
x [ 1.91031094 -0.24737285]
fval [8.89672307 2.71048121]
x,itercount   [ 1.91031094 -0.24737285] 2
iter count:  3
x [ 2.65120668 -0.00441721]
fval [21.28626504  6.02891636]
x,itercount   [ 2.65120668 -0.00441721] 3
```

```
iter count:  79
x [ 1.78538255e+17 -1.04409805e+17]
fval [6.82928290e+51 4.27773159e+34]
x,itercount   [ 1.78538255e+17 -1.04409805e+17] 79
iter count:  80
x [ 2.67807383e+17 -1.56614707e+17]
fval [2.30488298e+52 9.62489607e+34]
x,itercount   [ 2.67807383e+17 -1.56614707e+17] 80
iter count:  81
x [ 4.01711074e+17 -2.34922061e+17]
fval [7.77898005e+52 2.16560162e+35]
x,itercount   [ 4.01711074e+17 -2.34922061e+17] 81
iter count:  82
x [ 6.02566611e+17 -3.52383092e+17]
fval [2.62540577e+53 4.87260364e+35]
x,itercount   [ 6.02566611e+17 -3.52383092e+17] 82
iter count:  83
x [ 9.03849916e+17 -5.28574637e+17]
fval [8.86074446e+53 1.09633582e+36]
x,itercount   [ 9.03849916e+17 -5.28574637e+17] 83
iter count:  84
x [ 1.35577487e+18 -7.92861956e+17]
fval [2.99050126e+54 2.46675559e+36]
x,itercount   [ 1.35577487e+18 -7.92861956e+17] 84
iter count:  85
x [ 2.03366231e+18 -1.18929293e+18]
fval [1.00929417e+55 5.55020008e+36]
x,itercount   [ 2.03366231e+18 -1.18929293e+18] 85
iter count:  86
x [ 3.05049347e+18 -1.78393940e+18]
fval [3.40636784e+55 1.24879502e+37]
x,itercount   [ 3.05049347e+18 -1.78393940e+18] 86
iter count:  87
x [ 4.5757402e+18 -2.6759091e+18]
fval [1.14964915e+56 2.80978879e+37]
x,itercount   [ 4.5757402e+18 -2.6759091e+18] 87
iter count:  88
x [ 6.86361030e+18 -4.01386365e+18]
fval [3.88006586e+56 6.32202478e+37]
x,itercount   [ 6.86361030e+18 -4.01386365e+18] 88
iter count:  89
x [ 1.02954154e+19 -6.02079548e+18]
fval [1.30952223e+57 1.42245557e+38]
x,itercount   [ 1.02954154e+19 -6.02079548e+18] 89
iter count:  90
x [ 1.54431232e+19 -9.03119322e+18]
fval [4.41963752e+57 3.20052504e+38]
x,itercount   [ 1.54431232e+19 -9.03119322e+18] 90
iter count:  91
x [ 2.31646848e+19 -1.35467898e+19]
fval [1.49162766e+58 7.20118135e+38]
x,itercount   [ 2.31646848e+19 -1.35467898e+19] 91
iter count:  92
x [ 3.47470271e+19 -2.03201847e+19]
fval [5.03424337e+58 1.62026580e+39]
x,itercount   [ 3.47470271e+19 -2.03201847e+19] 92
iter count:  93
x [ 5.21205407e+19 -3.04802771e+19]
fval [1.69905714e+59 3.64559806e+39]
x,itercount   [ 5.21205407e+19 -3.04802771e+19] 93
```

```
iter count:  94
x [ 7.81808111e+19 -4.57204157e+19]
fval [5.73431784e+59 8.20259563e+39]
x,itercount   [ 7.81808111e+19 -4.57204157e+19] 94
iter count:  95
x [ 1.17271217e+20 -6.85806235e+19]
fval [1.93533227e+60 1.84558402e+40]
x,itercount   [ 1.17271217e+20 -6.85806235e+19] 95
iter count:  96
x [ 1.75906825e+20 -1.02870935e+20]
fval [6.53174641e+60 4.15256404e+40]
x,itercount   [ 1.75906825e+20 -1.02870935e+20] 96
iter count:  97
x [ 2.63860237e+20 -1.54306403e+20]
fval [2.20446441e+61 9.34326908e+40]
x,itercount   [ 2.63860237e+20 -1.54306403e+20] 97
iter count:  98
x [ 3.95790356e+20 -2.31459604e+20]
fval [7.44006739e+61 2.10223554e+41]
x,itercount   [ 3.95790356e+20 -2.31459604e+20] 98
iter count:  99
x [ 5.93685534e+20 -3.47189406e+20]
fval [2.51102275e+62 4.73002997e+41]
x,itercount   [ 5.93685534e+20 -3.47189406e+20] 99
iter count:  100
x [ 8.90528301e+20 -5.20784110e+20]
fval [8.47470177e+62 1.06425674e+42]
x,itercount   [ 8.90528301e+20 -5.20784110e+20] 100
returning
```

Out[4]:

```
('Does not converge try a new initial guess',
 array([ 8.90528301e+20, -5.20784110e+20]),
 -1)
```

Ok I am sorry I spent all weekend trying to fix this and it is making me frusturated. Ideally I will set up an office hours appointment but this is midterm week so I am submitting for now and hopefully the solution comes to me when I am in the shower or something and/or I meet with Paul a week after my midterms

Anyhow, I know the x,y coords should of intersection should be (0.508,0.061) and (-0.508,-0.061) and the roots of f_1 should be x=1 and x=-1. For f_2 the root should be should be x=0

I have tried multiple variations of this code as well as multiple other functions, although this one did overall better than my other attempts

References

- https://www.youtube.com/watch?v=Cs1g4qhgGxg for how one might do this by hand, helped me understand what I want my dumb code to do, unfortunately I didn't get there with my code YET

- http://hplgit.github.io/prog4comp/doc/pub/._p4c-solarized-Python031.html although now after wasting so much time I think this might be incorrect and its irritating me

- Lectures were informative and I thought this homework would be quite easy, however I clearly thought wrong and got stuck somewhere
- https://www.cmu.edu/math/undergrad/suami/pdfs/2014_newton_method.pdf
- graphed on sym lab to check my graph and determine the intersections https://www.symbolab.com/solver/non-linear-system-of-equations-calculator/x%5E%7B3%7D-y%5E%7B3%7D%2Bx%3D0%2C%20x%5E%7B2%7D%2By%5E%7B2%7D%3D1?or=input

In [ ]:

```

```