```python
import numpy as np
import pylab
import math

vp = 5
r1 = 1e3
r2 = 4e3
r3 = 3e3
r4 = 2e3
i0 = 3e-9
vt = 0.05

def Newton2(f1, f2, x1_init, x2_init, delta=1e-4):
    h = 0.05
    df1x1 = lambda x: 1/r1+1/r2+(i0/vt)*np.exp((x[0]-x[1])/vt)
    df1x2 = lambda x: (-i0/vt)*np.exp((x[0]-x[1])/vt)
    df2x1 = lambda x: (-i0/vt)*np.exp((x[1]-x[0])/vt)
    df2x2 = lambda x: 1/r3+1/r4+(i0/vt)*np.exp((x[0]-x[1])/vt)
    # df1x1 = lambda x: (f1(x[0] + h, x[1]) - f1(x[0] - h, x[1])) / (2 * h)
    # df1x2 = lambda x: (f1(x[0], x[1] + h) - f1(x[0], x[1] - h)) / (2 * h)
    # df2x1 = lambda x: (f2(x[0] + h, x[1]) - f2(x[0] - h, x[1])) / (2 * h)
    # df2x2 = lambda x: (f2(x[0], x[1] + h) - f2(x[0], x[1] - h)) / (2 * h)
    X = [x1_init, x2_init]
    # while np.fabs(f1(X[0], X[1])) > delta or np.fabs(f2(X[0], X[1])) > delta:
    diff = [1,1]
    while np.fabs(diff[0]) > delta or np.fabs(diff[1]) > delta:
        J = np.array([[df1x1(X), df1x2(X)],[df2x1(X), df2x2(X)]])
        F = np.array([f1(X[0], X[1]), f2(X[0], X[1])])
        DX = np.linalg.solve(J,F)
        X[0] -= DX[0]
        X[1] -= DX[1]
        diff = DX
    return X[0], X[1]


def main():
    f1 = lambda v1, v2: (v1-vp)/r1+v1/r2+i0*(np.exp((v1-v2)/vt)-1)
    f2 = lambda v1, v2: (v2-vp)/r3+v2/r4+i0*(np.exp((v2-v1)/vt)-1)
    res = Newton2(f1, f2, 4.5, 4.5, 1e-12)
    print(res)
    print(np.fabs(res[0] - res[1]))
    print(f1(res[0], res[1]), f2(res[0], res[1]))
    print(f1(2.66, 2.00), f2(2.66, 2.00))
    # pylab.plot(np.linspace(0, 5,100), [])

if __name__ == "__main__":
    main()
```