8.1.3

$\dot{x} = \mu x - x^2$ $\quad x_c = 0, \mu$

$\dot{y} = -y$ $\quad y_c = 0$

Jacobian: $\begin{pmatrix} \frac{\partial \dot{x}}{\partial x} & \frac{\partial \dot{x}}{\partial y} \\ \frac{\partial \dot{y}}{\partial x} & \frac{\partial \dot{y}}{\partial y} \end{pmatrix} = \begin{pmatrix} \mu - 2x & 0 \\ 0 & -1 \end{pmatrix}$

$J(x=0) = \begin{pmatrix} \mu & 0 \\ 0 & -1 \end{pmatrix}$ $\qquad$ $J(x_c = \mu) = \begin{pmatrix} -\mu & 0 \\ 0 & -1 \end{pmatrix}$

Eigenvalues: $(\mu, -1)$ $\qquad$ Eigenvalues $(\lambda_1, \lambda_2) = (-\mu, -1)$

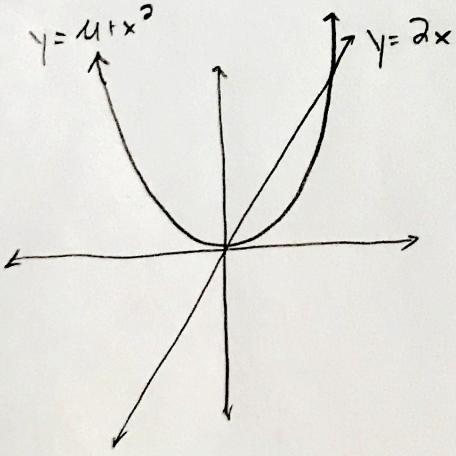
$\lambda_1 \to 0 \quad \mu \to 0$

8.1.6 $\quad \dot{x} = y - 2x$

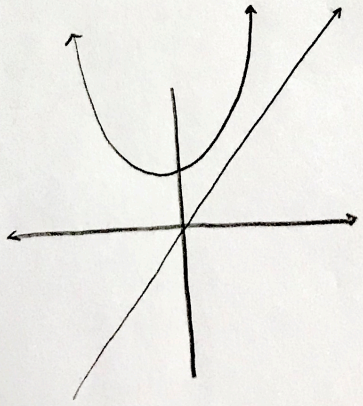
$\dot{y} = \mu + x^2 - y$

nullclines: $\dot{x} = 0 \Rightarrow y = 2x$

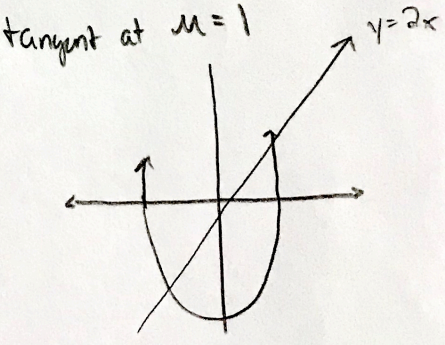
$\dot{y} = 0 \Rightarrow y = \mu + x^2$



$\mu = 0$



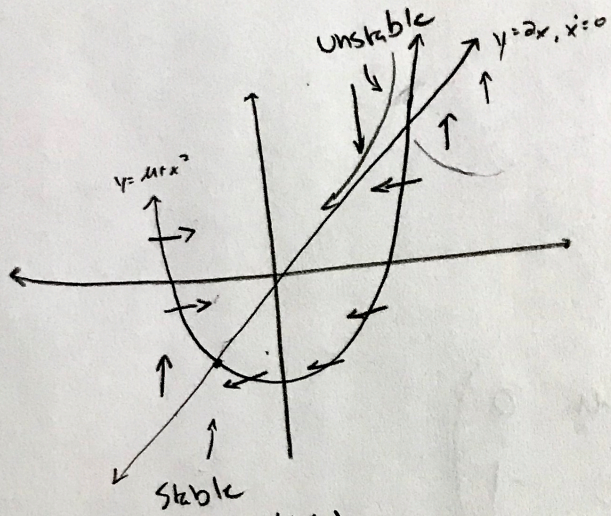
$\mu > 0$



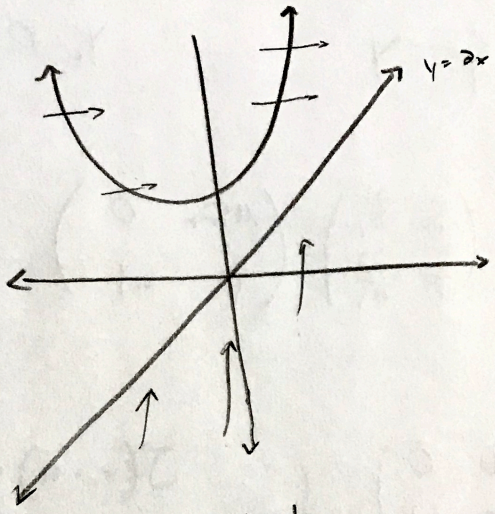
$\mu < 0$

b) bifurcation appears at $\mu = 1$

saddle node

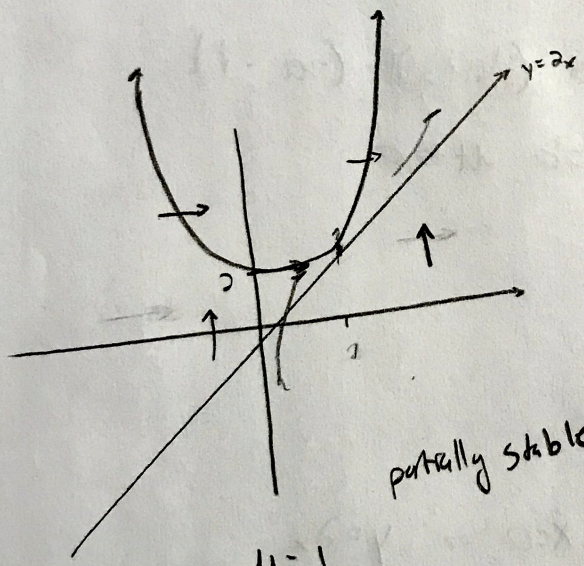


$\mu < 1$



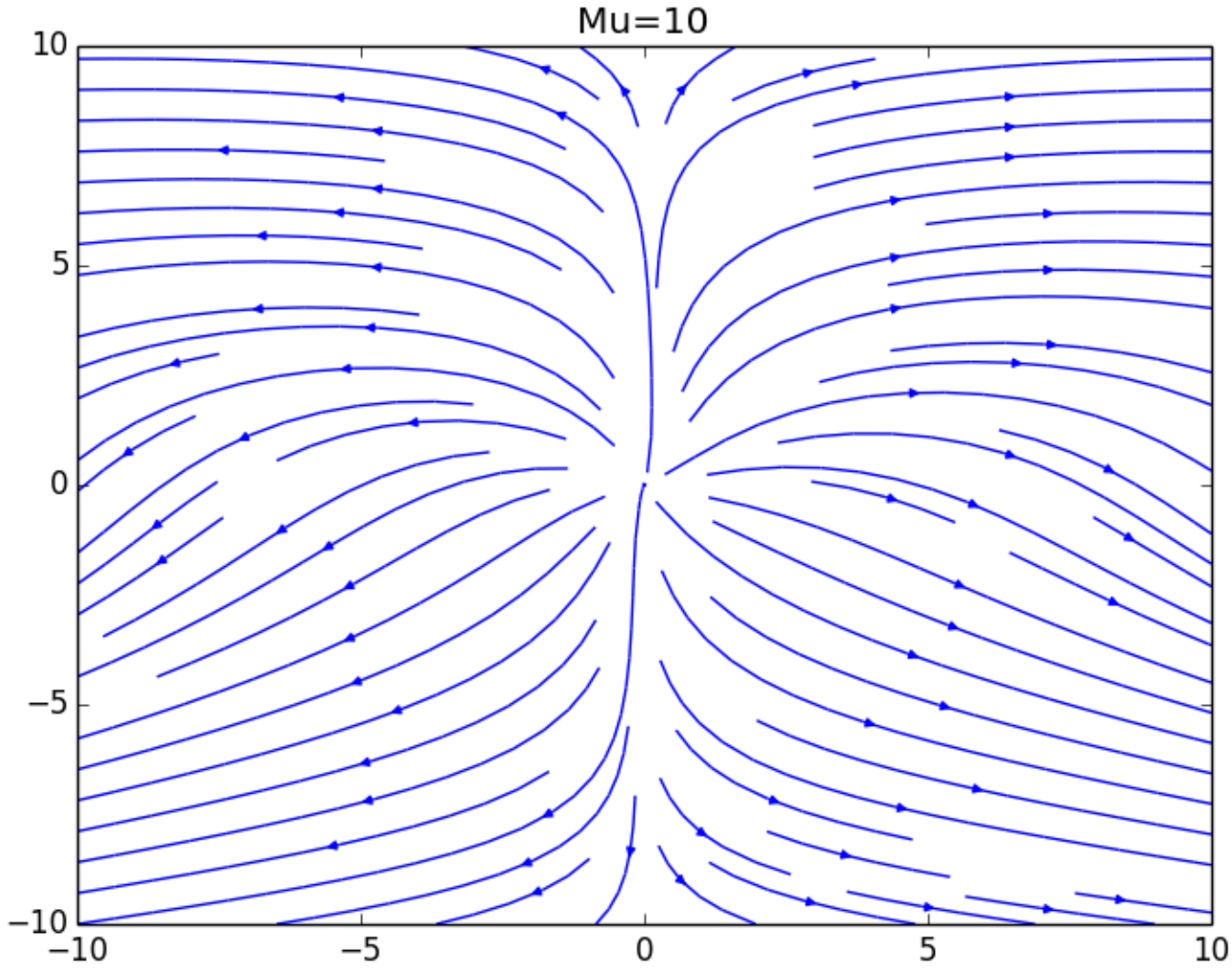
$\mu > 1$

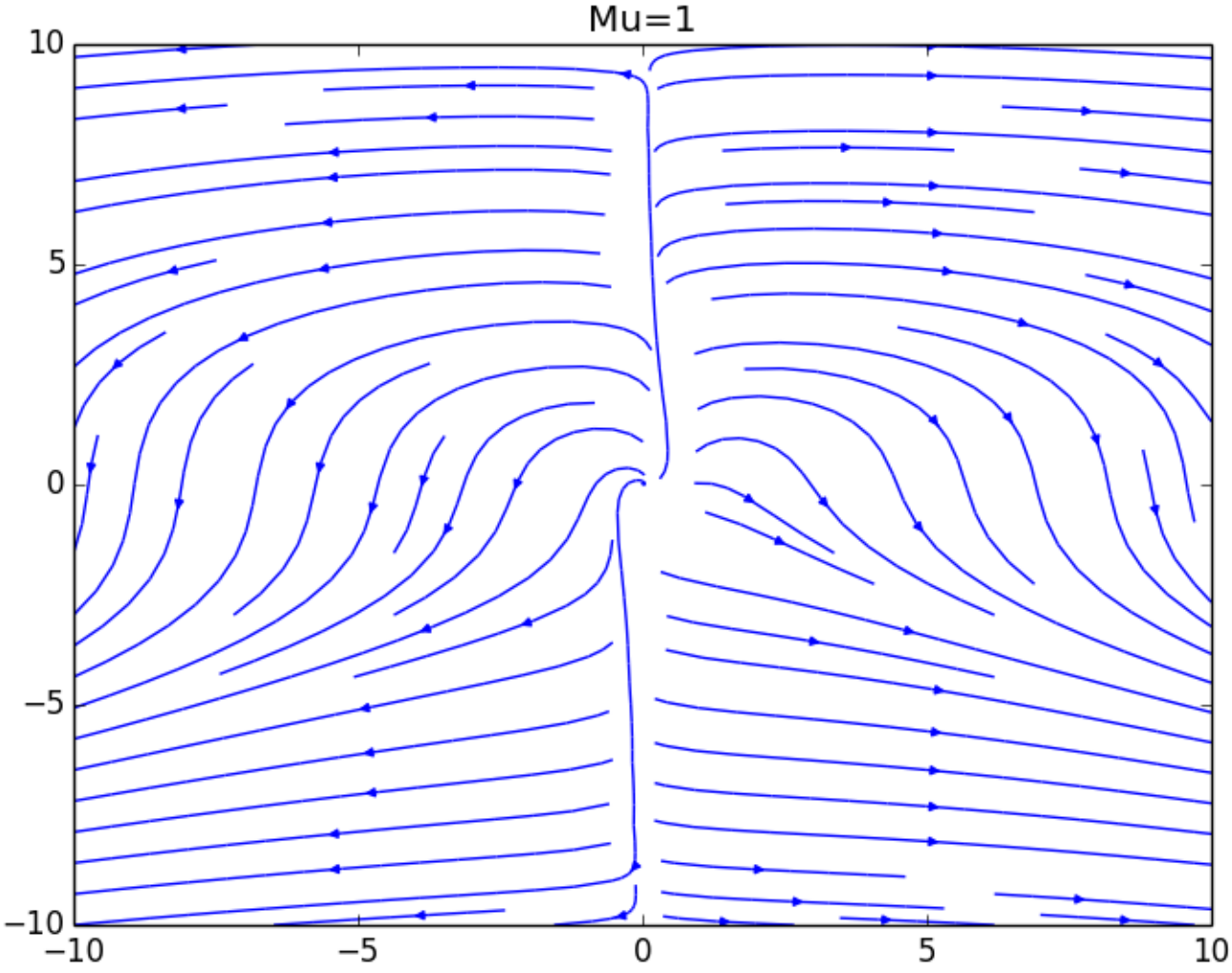
no fixed point

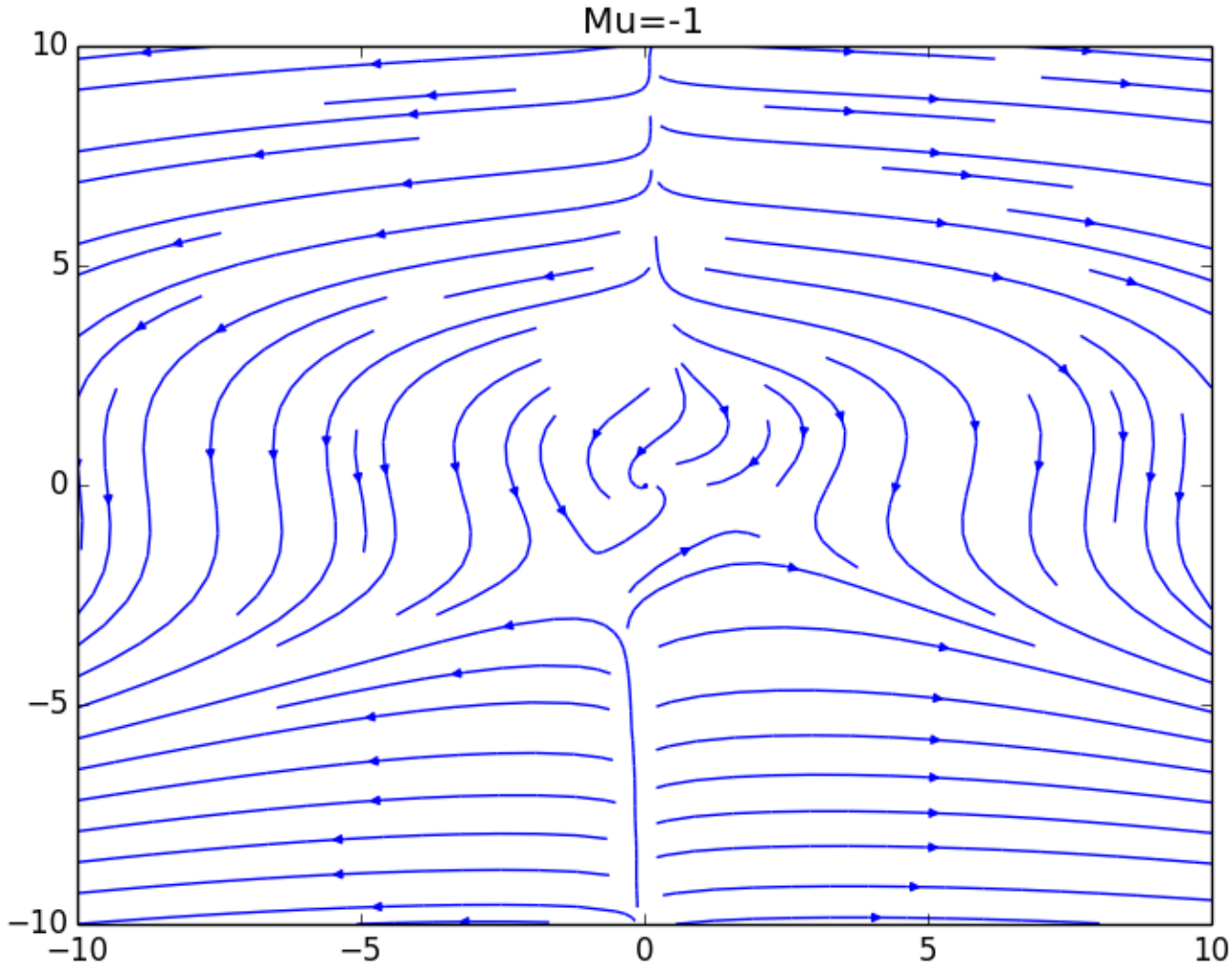


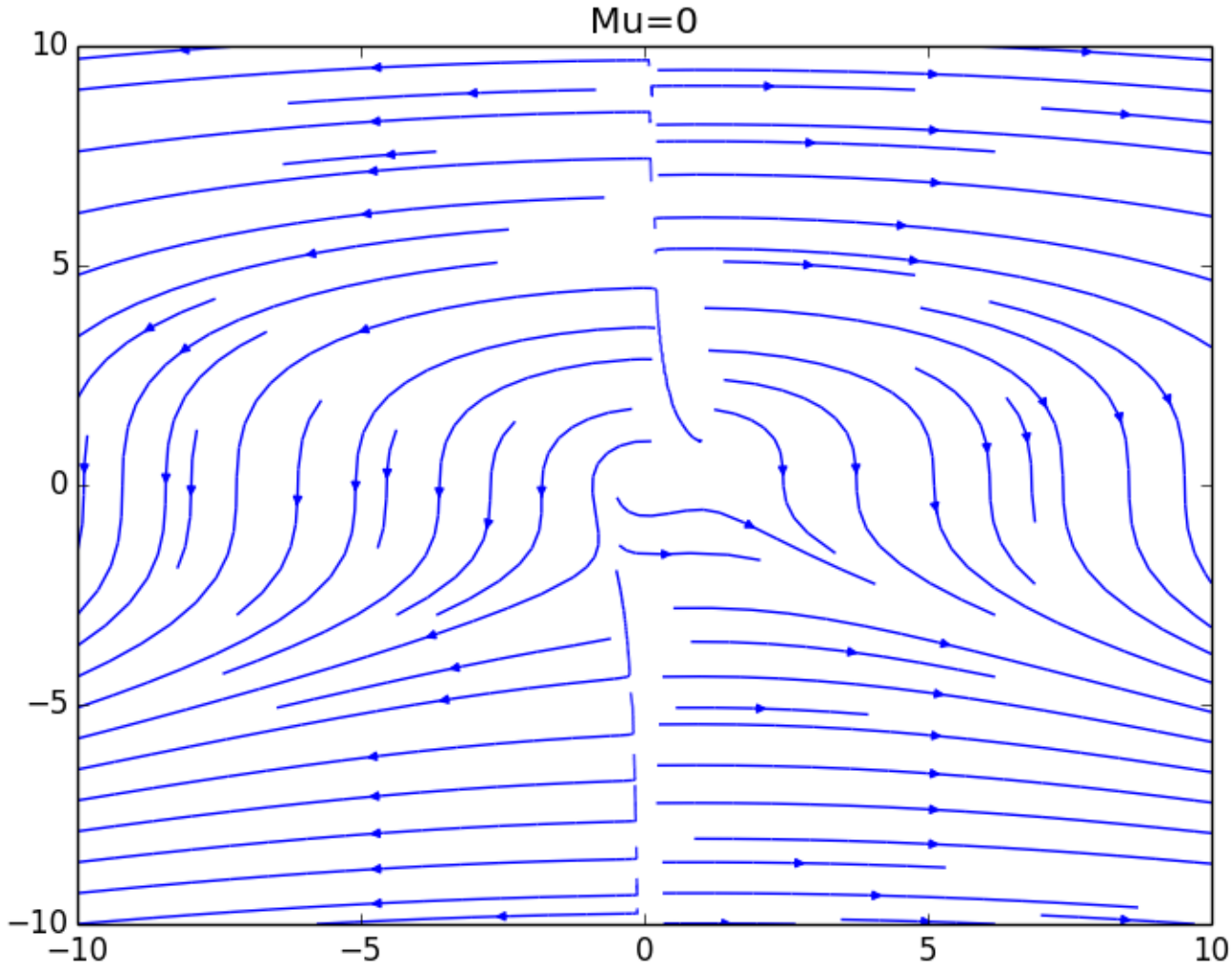
partially stable

$\mu = 1$


Mu=10
10
5
0
−5
−10
−10
−5
0
5
10

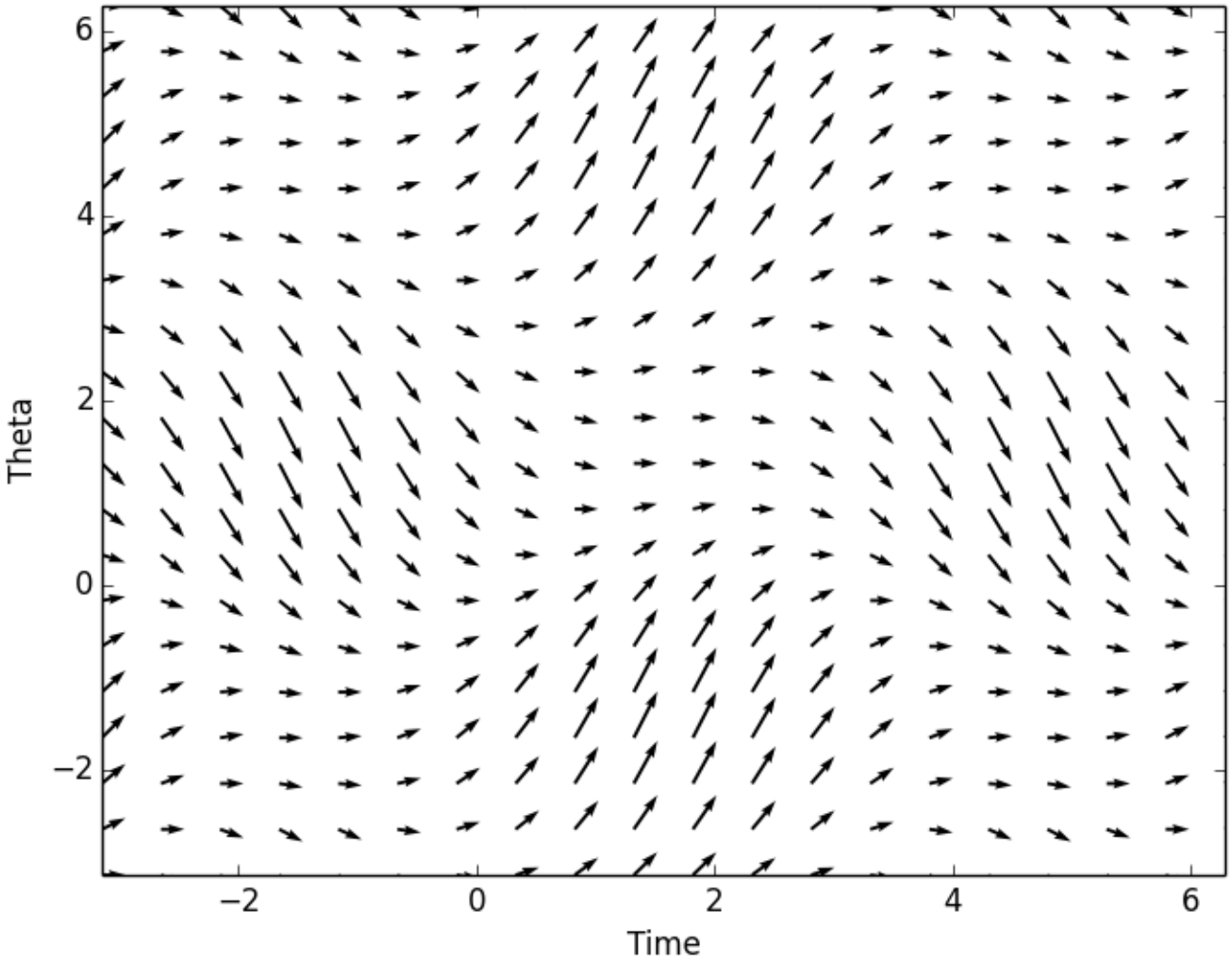

Mu=1
10
5
0
−5
−10
−10
−5
0
5
10

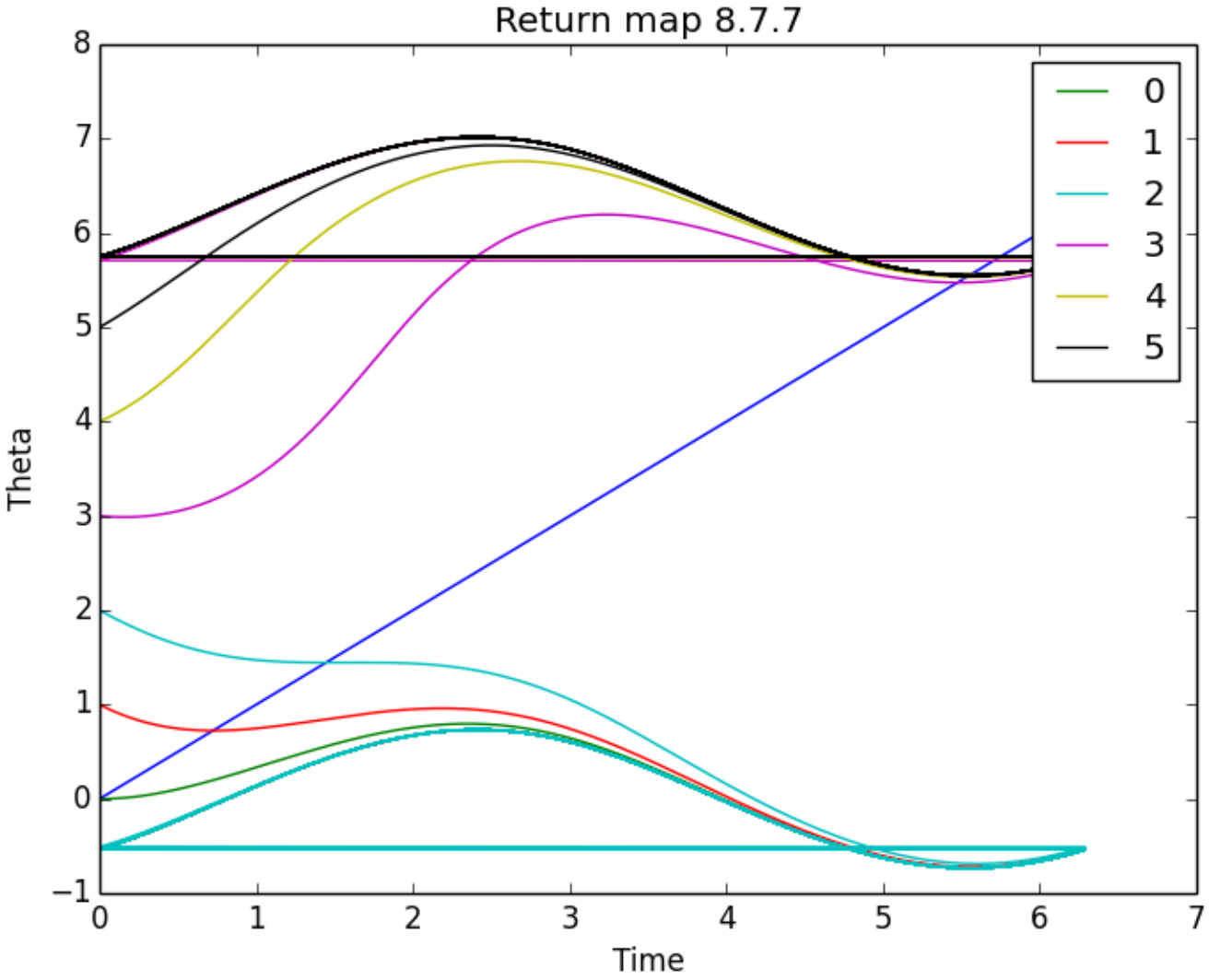

Mu=-1
10
5
0
−5
−10
−10
−5
0
5
10


Mu=0
10
5
0
−5
−10
−10
−5
0
5
10

streamplot.py — Edited

```
##written in python 2.7


import matplotlib.pyplot as plt
import numpy as np

x=y=np.linspace(-10,10)

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

mew=-1
xdot=-Y+mew*X+X*Y**2
ydot=X+mew*Y-X**2

plt.streamplot(X,Y,xdot,ydot)
plt.title("Mu=-1")
plt.show()
```


Theta
Time
6
4
2
0
−2
−2
0
2
4
6


Return map 8.7.7
Theta
Time
0
1
2
3
4
5

8_7_7_integration.py

```
#written using python2.7 (filename changed .py-->.txt)


import numpy as np
import matplotlib.pyplot as plt

def integrate(x):    #integrate as a function of theta (theta=x)
        dx=.01
        time=0   #initial time
        t_list=[]
        theta_list=[]
        dt=.01
        x0=x  #variable to store labels
        i=0
        while i<4000*np.pi:
                i+=1
                dxdt=np.sin(time)-np.sin(x)   #function to integrate
                x+= dxdt*dt     #euler stepping
                time+=dt
                t_list.append(time)
                theta_list.append(x)
                #plt.plot(time,x,'--bo')
                if time>2*np.pi:        ##if statement that repeats the loop continuing to
                                        ##iterate over theta but resetting time,
                                        ##effectively creating the return map for
                                        ##different cycles
                        time=0
        plt.plot(t_list,theta_list, label=x0)
        plt.xlabel("Time")
        plt.ylabel("Theta")
        plt.title("Return map 8.7.7")

q=np.linspace(0,2*np.pi)    #generate t=theta line
plt.plot(q,q)

n=1
for n in range(0,6):    #plots the function for several starting values of theta
        integrate(n)
plt.legend()
plt.show()
```