1) Construct the phase portrait for Problem 2.1g of Nayfeh and Mook.

The potential function for the following equation can be constructed by integrating the force function, $f(x)$. The equation of motion is defined as

$$\ddot{u} + u - \frac{\lambda}{a-u} = 0 \tag{1}$$

The force function in above equation is

$$f(u) = u - \frac{\lambda}{a-u} \tag{2}$$

Therefore, the potential function can be calculated as:

$$\begin{aligned} F(u) &= \int f(u)du \\ &= \int u - \frac{\lambda}{a-u} = \frac{u^2}{2} + \lambda \ln\left(|a-u|\right) \end{aligned}$$

Therefore, the potential function can be written as:

$$F(u) = \frac{1}{2}u^2 + \lambda \ln\left(|a-u|\right) \tag{4}$$

The mechanical energy consists of potential and kinetic energy. This can be written as:

$$E(u) = \frac{1}{2}\dot{u}^2 + \frac{1}{2}u^2 + \lambda \ln\left(|a-u|\right) \tag{5}$$

Furthermore, we can calculate the equilibrium points by setting the time derivative in Equation (1) equal to zero. The equilibrium points are calculated as follows:

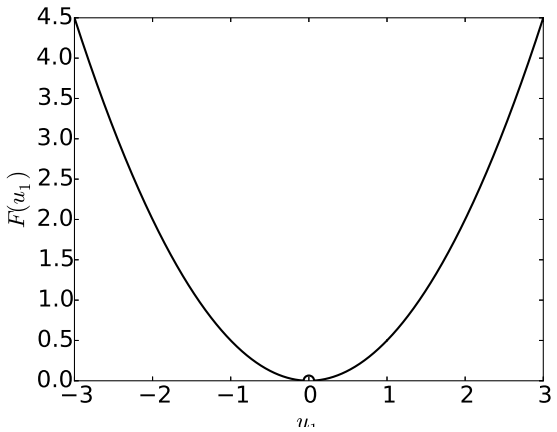
$$u_{eq_1} = \frac{a + \sqrt{a^2 - 4\lambda}}{2} \tag{6a}$$

$$u_{eq_2} = \frac{a - \sqrt{a^2 - 4\lambda}}{2} \tag{6b}$$

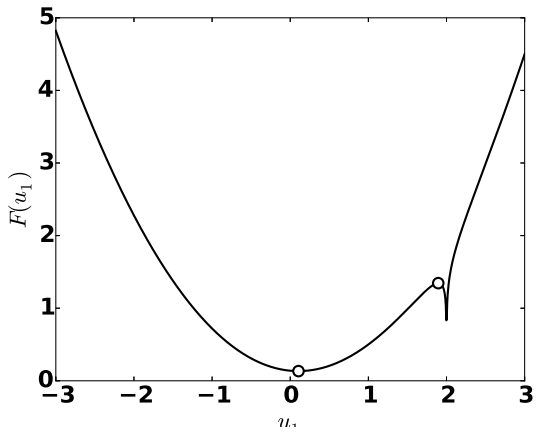
As can be seen in Equation (6), based on a, and λ we have three cases for the equilibrium points.

- $a^2 - 4\lambda > 0$: Two equilibrium points
- $a^2 - 4\lambda = 0$: One equilibrium points
- $a^2 - 4\lambda < 0$: No equilibrium points

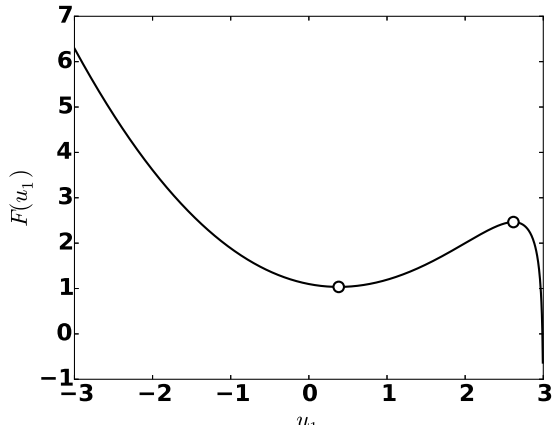
We plotted the potential energy function, $F(u)$, for $u \in [-3.0,\ 3.0]$ for different cases of a and λ in Figure 2. The equilibrium points are calculated buy Equation (6). As can be seen in the following figures, the equilibrium points are locations where the slope of the potential function is zero. For the first case, when there are two equilibrium points, one of them is stable and the other one is an unstable point.



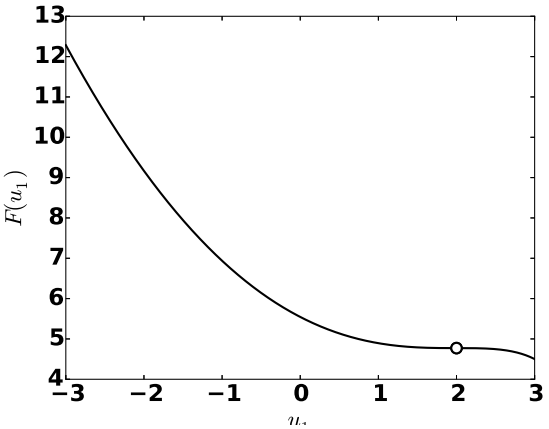
(a) $a = 1.0,\ \lambda = 0.0$



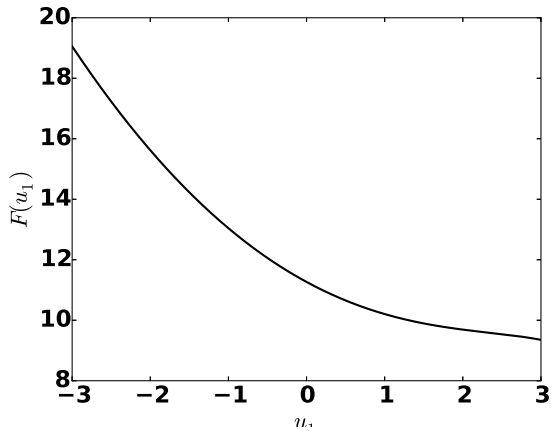
(b) $a = 2.0,\ \lambda = 0.2$



(c) $a = 3.0,\ \lambda = 1.0$

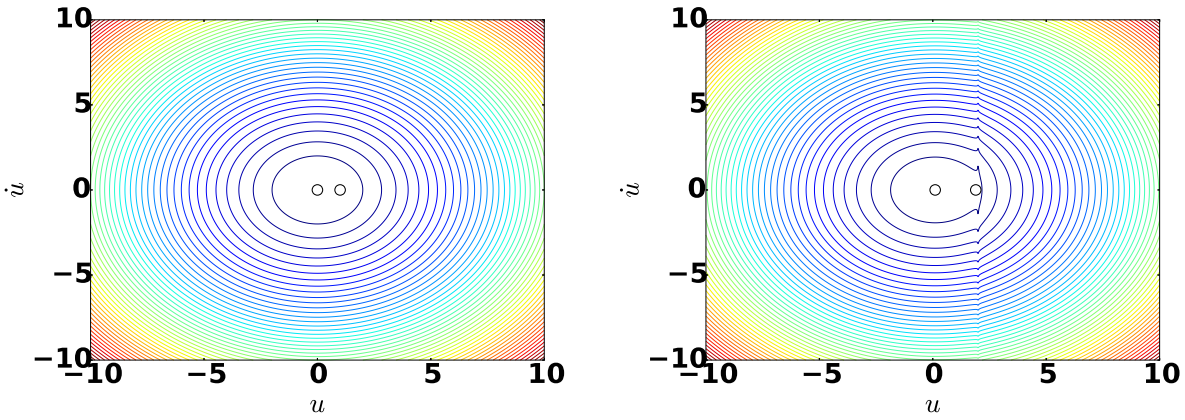


(d) $a = 4.0,\ \lambda = 1.0$



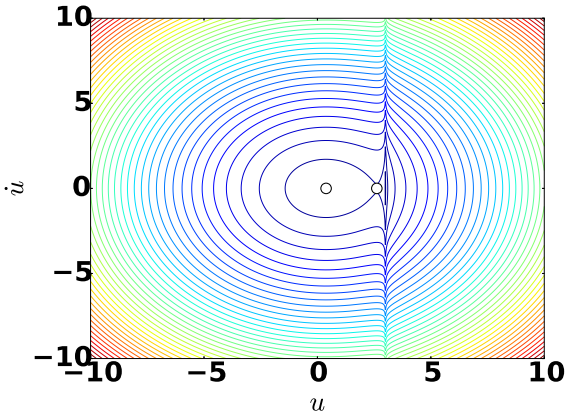
(e) $a = 5.0,\ \lambda = 7.0$

Figure 1: Potential energy vs. displacement for different values of a and λ. The white circle corresponds to the equilibrium point of the system.

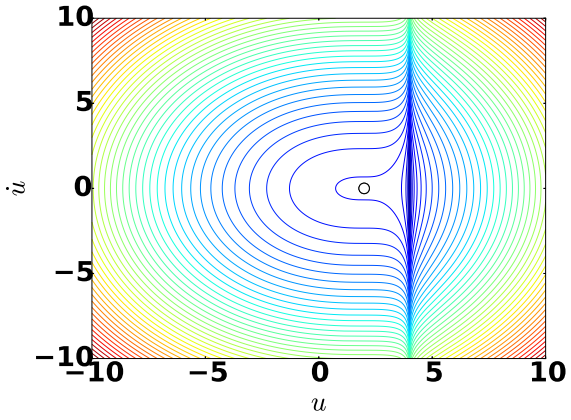


(a) $a = 1.0,\ \lambda = 0.0$

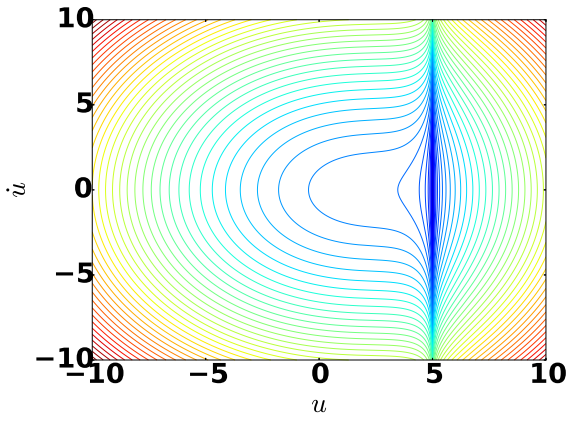
(b) $a = 2.0,\ \lambda = 0.2$



(c) $a = 3.0,\ \lambda = 1.0$



(d) $a = 4.0,\ \lambda = 1.0$



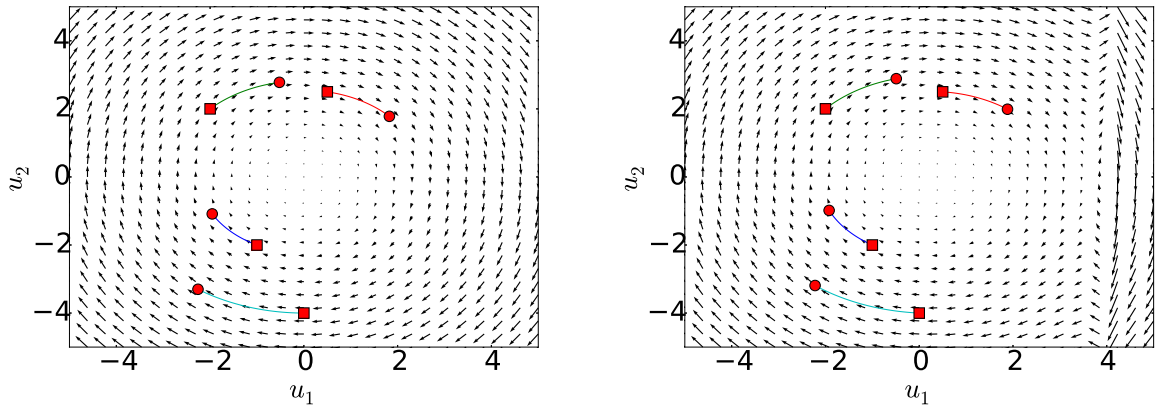
(e) $a = 5.0,\ \lambda = 7.0$

Figure 2: Contour plots of total mechanical energy (potential + kinetic energy) for different values of a and λ. The white circle corresponds to the equilibrium point of the system.

To draw the phase plot, the equation of motion needs to put into the state form. This is done by defining new variable u_1 and u_2 and rewriting Equation (1) in the following form.

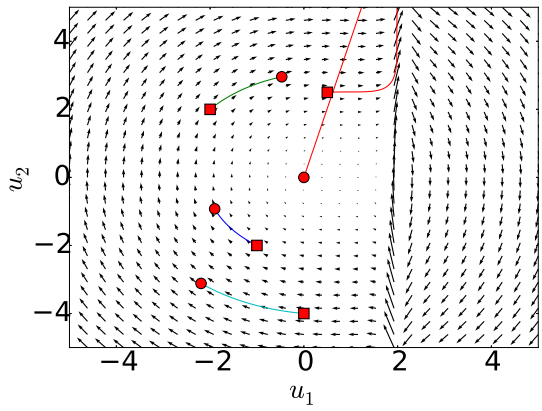
$$\begin{cases} \dot{u}_1 = u_2 \\ \dot{u}_2 = -u_1 + \dfrac{\lambda}{a - u_1} \end{cases} \tag{7}$$

The phase plot ($\dot{u}$ vs. u) of this system is shown in Figure 3. The lines represents different trajectories of the solution for different boundary condition. These are calculated by integrating Equation (7) using `odeint` function in `Python`.

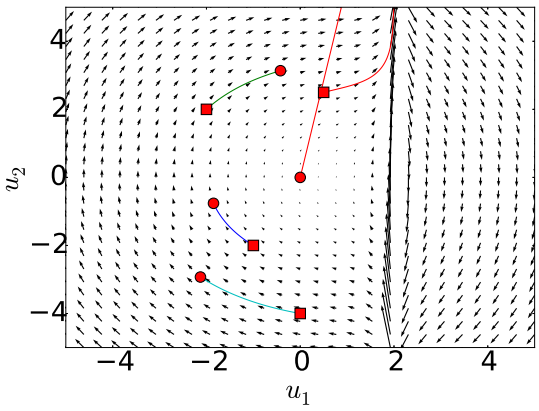


(a) $a = 4.0,\ \lambda = 0.0$

(b) $a = 4.0,\ \lambda = 1.0$



(c) $a = 2.0,\ \lambda = 1.0$



(d) $a = 2.0,\ \lambda = 2.0$

Figure 3: Phase plot and trajectories of the solution for different values of a, λ, and boundary conditions. $\square$ represents the initial point and $\circ$ represents the final point of the trajectory.

2) Construct the phase portrate for the following differential equations

The phase portrait is constructed by writing the equation of motion in the state-space and calculate the tangent vector to the solution at each point in the state-space at time zero. For all of the following equation we choose our states as follows

$$x_1 = x$$
$$x_2 = \dot{x}_1$$

After rewriting the equation is the state-space, we use Python's `numpy.meshgrid()` to generate a mesh over our state variables, x_1, x_2. By substituting these variables in our state equation, we can get the tangents to the solution at each point. These tangents are plotted using `matplotlib.pyplot.quiver` function in Python. A sample code is available in the appendix.

The following are the state-space representation of the second order differential equations for this problem.

$$\ddot{x} + x + x^3 = 0 \longrightarrow \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_1 - x_1^3 \end{bmatrix} \tag{8}$$

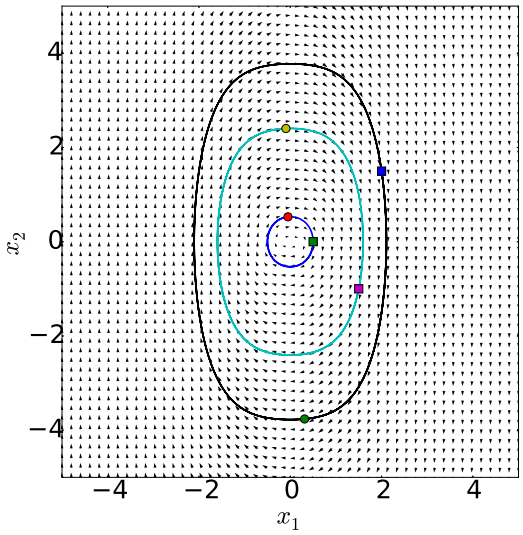
$$\ddot{x} + x - x^3 = 0 \longrightarrow \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_1 + x_1^3 \end{bmatrix} \tag{9}$$

$$\ddot{x} - x + x^3 = 0 \longrightarrow \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_1 - x_1^3 \end{bmatrix} \tag{10}$$

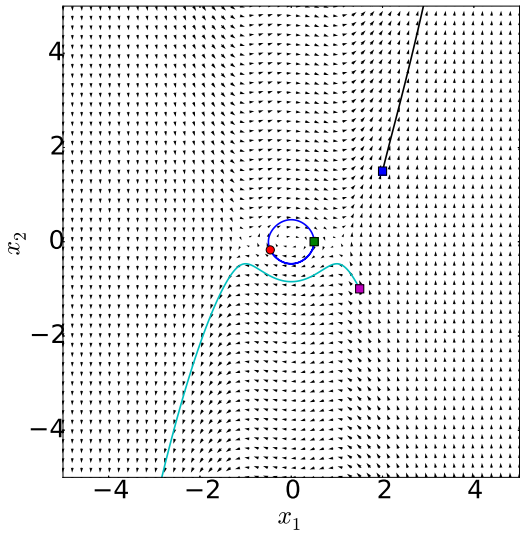
$$\ddot{x} - x - x^3 = 0 \longrightarrow \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_1 + x_1^3 \end{bmatrix} \tag{11}$$

$$\ddot{x} + x^3 = 0 \longrightarrow \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_1^3 \end{bmatrix} \tag{12}$$

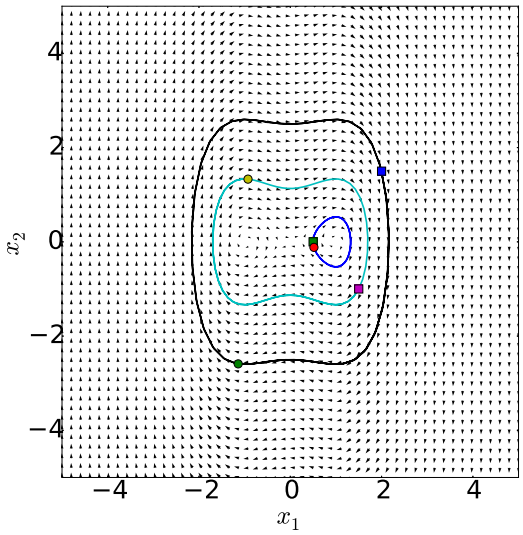
The phase plots for these equations are shown in the Figure 4. The lines plotted on the phase plot represent the trajectory of the system with different initial conditions. The blue, cyan, and black lines have initial location of 0.5, 1.5, 2.0 and initial velocity of 0.0, -1.0, 1.5 respectively. The square represents the origin of the trajectory and circle represents its end.



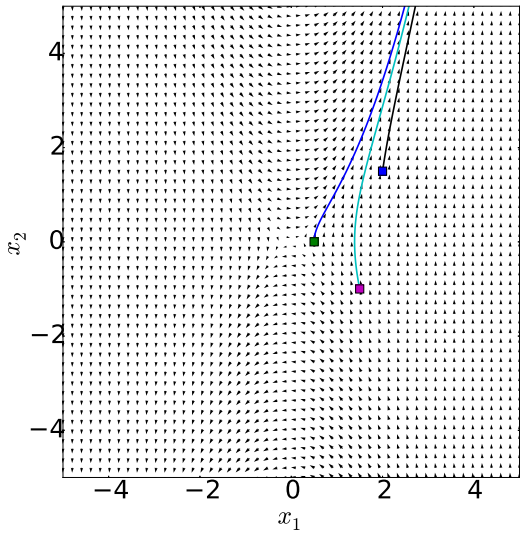
(a) $\ddot{x} + x + x^3 = 0$



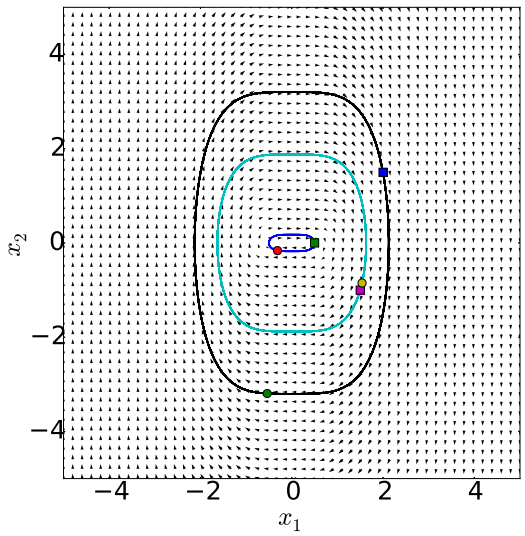
(b) $\ddot{x} + x - x^3 = 0$



(c) $\ddot{x} - x + x^3 = 0$



(d) $\ddot{x} - x - x^3 = 0$



(e) $\ddot{x} + x^3 = 0$

Figure 4: Phase portrait plot.

As can be seen from the phase plots in Figure 4, Equations (1), (3), and (5) are stable system. It is also clear than Equations (1) and (5) have one equilibrium solution whereas (3) has two. Equation (2) is asymptotically stable. It has stable solution for initial conditions near the equilibrium point and becomes unstable when the the initial point moves further. Equation (4) is unstable as can be

seen in Figure 4.

3) Determine when the following set of equations are dissipative and when the are conservative

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy \end{aligned} \tag{13}$$

The divergence of above vector field can be written as follows:

$$\frac{\partial \dot{x}}{\partial x} + \frac{\partial \dot{y}}{\partial y} + \frac{\partial \dot{z}}{\partial z} = -\sigma - 1 - \beta \tag{14}$$

Depending on the sign of the Equation (14), the system can be dissipative or conservative. If above equation is negative the system is dissipative, and if it is zero the system is conservative. This is can be written as follows:

$$\begin{cases} \sigma + \beta &= -1, \quad \text{conservative system} \\ \sigma + \beta &< -1, \quad \text{disipative system} \end{cases} \tag{15}$$

ρ is a free parameter for this problem.

4) Determine when the following set of equations are dissipative and when the are conservative

$$\begin{aligned} \dot{x} &= -(y+z) \\ \dot{y} &= x + ay \\ \dot{z} &= b + (x-c)z \end{aligned} \tag{16}$$

The divergence of above vector field can be written as follows:

$$\frac{\partial \dot{x}}{\partial x} + \frac{\partial \dot{y}}{\partial y} + \frac{\partial \dot{z}}{\partial z} = a + x - c \tag{17}$$

Depending on the sign of the Equation (17), the system can be dissipative or conservative. If above equation is negative the system is dissipative, and if it is zero the system is conservative. This is can be written as follows:

$$\begin{cases} c - a = x, & \text{conservative system} \\ c - a > x, & \text{disipative system} \end{cases} \tag{18}$$

b is the free parameters for this problem.

5) Determine when the following map is dissipative

$$\begin{aligned} x_{n+1} &= a_{11}x_n + a_{12}y_n \\ y_{n+1} &= a_{21}x_n + a_{22}y_n \end{aligned} \tag{19}$$

A map is said to be dissipative at $\mathbf{x}_k = \mathbf{x}_0$ if

$$|\det D_{\mathbf{x}_k}\mathbf{F}| < 1 \quad \text{at} \quad \mathbf{x}_k = \mathbf{x}_0 \tag{20}$$

For Equation (19) we have

$$|\det D_{\mathbf{x}_k}\mathbf{F}| = \left|\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\right| = a_{11} \times a_{22} - a_{12} \times a_{21} \tag{21}$$

For the system to be dissipative we should have

$$a_{11} \times a_{22} - a_{12} \times a_{21} < 0 \tag{22}$$

Appendix

```
'''
Python code for first differential equation in problem (2)
'''
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

# Define the font size for the figures
font = {'size'   : 62}
plt.rc('font', **font)

# Define the state space matrix
def RHS2A(X, t=0.0):
    x1, x2 = X
    x1dot = x2
    x2dot = -x1 - x1**3
    return [x1dot, x2dot]

# Defining a mesh where the state matrix is evaluated
x1 = np.linspace(-5.0, 5.0, 50)
x2 = np.linspace(-5.0, 5.0, 50)
X1, X2 = np.meshgrid(x1, x2)
X1dot = np.zeros(X1.shape)
X2dot = np.zeros(X2.shape)

# For nomalizing the state space vectors
normalize = True
for ni in range(0, len(x1)):
    for nj in range(0, len(x2)):
        X1dot[ni, nj], X2dot[ni, nj] = RHS2A([X1[ni, nj], X2[ni, nj]])
        if normalize:
            X1dot[ni, nj] = X1dot[ni, nj] / (X1dot[ni, nj]**2.0 + X2dot[ni, nj]*
            X2dot[ni, nj] = X2dot[ni, nj] / (X1dot[ni, nj]**2.0 + X2dot[ni, nj]*

# ploting the phase portrait
plt.figure(figsize=[20, 20])
plt.quiver(X1, X2, X1dot, X2dot)
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.xlim([x1[0], x1[-1]])
plt.ylim([x2[0], x2[-1]])

# Define initial conditions for the trajectories
x10 = [0.5, 1.5, 2.0]
x20 = [0.0, -1.0, 1.5]
t = np.linspace(0.0, 10.0, 1000)
t = np.linspace(0.0, 10.0, 1000)
for nInit in range(0, len(x10)):
    sol = odeint(RHS2A, [x10[nInit], x20[nInit]], t)
    # Filtering the results that goes to infinity
    for nSol in range(0, len(sol)):
        if sol[nSol, 0] > 100.0:
            break
        elif sol[nSol, 1] > 100:
```

```
            break
        elif sol[nSol, 0] < -100:
            break
        elif sol[nSol, 1] < -100:
            break
    plt.plot(sol[0:nSol, 0], sol[0:nSol, 1], lw=4.0)
    plt.plot(sol[0, 0], sol[0, 1], 's', ms=20)
    plt.plot(sol[nSol, 0], sol[nSol, 1], 'o', ms=20)

plt.show()
```