FYS2130 - Project

Candidate - 15266

April 2018

Problems

Problem 1

We are asked to solve the equation

$$ma(t) + kx(t) = 0, \tag{1}$$

numerically using the Runge-Kutta 4 method where we have written the code ourself. We start by setting up an expression we can feed into our function, which is basicly just to solve for a

$$a(t) = -\frac{k}{m}x(t). \tag{2}$$

Using the constants and initial conditions given in the problem

Figure 1: Phase diagram of a harmonic oscilator.

Problem 2

Problem 3

$$mx'' + kx \tag{3}$$

Problem 4

Problem 5

Problem 6

Problem 7

Problem 8

Problem 9

Appendix

Python code

```

import numpy as np
import matplotlib.pyplot as plt


class Solver():
    def __init__(self, time, dt):
        self.time = time
        self.dt = dt
        self.N = int(self.time/self.dt)

        self.x = np.zeros(self.N)
        self.v = np.zeros(self.N)
        self.t = np.zeros(self.N)
        self.m = np.zeros(self.N)

        self.dm = 1.0


    def set_initial_conditions(self, x0, v0, m0):
        self.x[0] = x0
        self.v[0] = v0
        self.m[0] = m0


    def set_time(self, time):
        self.time = time
        self.N = int(self.time/self.dt)

        self.x = np.zeros(self.N)
        self.v = np.zeros(self.N)
        self.t = np.zeros(self.N)
        self.m = np.zeros(self.N)


    def set_dt(self, dt):
        self.dt = dt


```

```
    def set_dm(self, dm):
        self.dm = dm


    def set_diff_eq(self, f):
        self.diff_eq = f;


    def rk4_step(self, x0, v0, t0, mi):
        dt = self.dt

        a1 = self.diff_eq(x0, v0, t0, mi)
        v1 = v0

        x_half_1 = x0 + v1 * dt/2.0
        v_half_1 = v0 + a1 * dt/2.0

        a2 = self.diff_eq(x_half_1, v_half_1, t0+dt/2.0, mi)
        v2 = v_half_1

        x_half_2 = x0 + v2 * dt/2.0
        v_half_2 = v0 + a2 * dt/2.0

        a3 = self.diff_eq(x_half_2, v_half_2, t0+dt/2.0, mi)
        v3 = v_half_2

        x_end = x0 + v3 * dt
        v_end = v0 + a3 * dt

        a4 = self.diff_eq(x_end, v_end, t0 + dt, mi)
        v4 = v_end

        a_middle = 1.0/6.0 * (a1 + 2*a2 + 2*a3 + a4)
        v_middle = 1.0/6.0 * (v1 + 2*v2 + 2*v3 + v4)

        x_end = x0 + v_middle * dt
        v_end = v0 + a_middle * dt

        return x_end, v_end


    def solve(self):
        x = self.x; v = self.v; t = self.t; N = self.N; dt = self.dt; m = self.m; dm = self.dm
        for i in range(N-1):
            [x[i+1], v[i+1]] = self.rk4_step(x[i], v[i], t[i], m[i])
            t[i+1] = t[i] + dt
            m[i+1] = m[i] + dm

        return x, v, t, m



class Plotter():
    def __init__(self):
        a = 0
```

```
    def set_parameters(self, x_values, y_values, title, xlabel, ylabel, color):
        self.x_values = x_values
        self.y_values = y_values
        self.title = title
        self.xlabel = xlabel
        self.ylabel = ylabel
        self.color = color

    def show(self):
        plt.title(self.title)
        plt.xlabel(self.xlabel)
        plt.ylabel(self.ylabel)
        plt.plot(self.x_values, self.y_values, self.color)
        plt.show()


#
def diff_eq_1(x, v, t, m):
    """
    Differential equation for problem 1
    """
    m = 0.5
    k = 1.0

    a = -(1./.5)*x

    return a


#
def diff_eq_2(x, v, t, m):
    """
    Differential equation for problem 2
    """
    m = 0.5     # []
    k = 1.0     # []
    b = 0.1     # []

    a = -(k/m)*x - (b/m)*v

    return a


#
def diff_eq_4(x, v, t, m):
    """
    Differential equation for problem 4
    """
    m = 0.5     # []
    k = 1.0     # []
    F_D = 0.7   # N
    w_0 = np.sqrt(k/m)
    omega_D = 13.0/(8.0*w_0)

    a = (F_D*np.cos(omega_D*t)-k*x)/m
```

```

    return a


#
def diff_eq_5(x, v, t, m):
    """
    Differential equation for problem 5
    """
    m = 0.5      # []
    k = 1.0      # []
    b = 0.1      # []
    F_D = 0.7    # N
    w_0 = np.sqrt(k/m)
    omega_D = 13.0/(8.0*w_0)

    a = (F_D*np.cos(omega_D*t)-k*x-b*v)/m

    return a


#
def diff_eq_6(x, v, t, m):
    """
    Differential equation for problem 5
    """
    k = 0.475        # []
    b = 0.001        # []
    g = 9.81         #

    a = -(b*v + k*x + g)/m

    print m

    return a


solver = Solver(20.0, 1e-2)
plotter = Plotter()

problem_to_show = 6
if( problem_to_show == 1 ):
    solver.set_diff_eq(diff_eq_1)
    solver.set_initial_conditions(1.0, 0.0, 0.5)
    [x, v, t, m] = solver.solve();
    plotter.set_parameters(x, v, 'Testplot', 'x', 'y', 'r')
    plotter.show()
elif( problem_to_show == 2 ):
    solver.set_diff_eq(diff_eq_2)
    solver.set_initial_conditions(1.0, 0.0, 0.5)
    [x, v, t, m] = solver.solve();
    plotter.set_parameters(x, v, 'Testplot', 'x', 'y', 'r')
    plotter.show()
elif( problem_to_show == 4):
    solver.set_diff_eq(diff_eq_4)
    solver.set_time(200.0)
    solver.set_initial_conditions(2.0, 0.0, 0.5)
```

```
    [x, v, t, m] = solver.solve();
    plotter.set_parameters(x, v, 'Testplot', 'x', 'y', 'r')
    plotter.show()
elif( problem_to_show == 5):
    solver.set_diff_eq(diff_eq_5)
    solver.set_time(100.0)
    solver.set_initial_conditions(2.0, 0.0, 0.5)
    [x, v, t, m] = solver.solve();
    plotter.set_parameters(x, v, 'Testplot', 'x', 'y', 'r')
    plotter.show()
elif( problem_to_show == 6):
    solver.set_diff_eq(diff_eq_6)
    solver.set_time(3.0)
    solver.set_dt(1e-4)
    solver.set_dm(0.00055)
    solver.set_initial_conditions(0.001, 0.001, 0.00001)
    [x, v, t, m] = solver.solve();
    plotter.set_parameters(x, v, 'Testplot', 'x', 'y', 'r')
    plotter.show()
else:
    print "Please input a valid problem numer: [1,2,4]"
```