

FYS2130 - Project

Candidate - 15266

April 2018

Problems

Problem 1

We are asked to solve the equation

$$ma(t) + kx(t) = 0, \quad (1)$$

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 basically just to solve for a

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

Using the constants and initial conditions given in the problem

Figure 1: Phase diagram of a harmonic oscillator.

Problem 2

Problem 3

$$mx'' + kx \quad (3)$$

Problem 4

Problem 5

Problem 6

Problem 7

Problem 8

Problem 9

Appendix

Python code

```
1
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5
6 class Solver():
7     def __init__(self, time, dt):
8         self.time = time
9         self.dt = dt
10        self.N = int(self.time/self.dt)
11
12        self.x = np.zeros(self.N)
13        self.v = np.zeros(self.N)
14        self.t = np.zeros(self.N)
15        self.m = np.zeros(self.N)
16
17        self.dm = 1.0
18
19
20    def set_initial_conditions(self, x0, v0, m0):
21        self.x[0] = x0
22        self.v[0] = v0
23        self.m[0] = m0
24
25
26    def set_time(self, time):
27        self.time = time
28        self.N = int(self.time/self.dt)
29
30        self.x = np.zeros(self.N)
31        self.v = np.zeros(self.N)
32        self.t = np.zeros(self.N)
33        self.m = np.zeros(self.N)
34
35
36    def set_dt(self, dt):
37        self.dt = dt
38
39
```

```

40     def set_dm(self, dm):
41         self.dm = dm
42
43
44     def set_diff_eq(self, f):
45         self.diff_eq = f;
46
47
48     def rk4_step(self, x0, v0, t0, mi):
49         dt = self.dt
50
51         a1 = self.diff_eq(x0, v0, t0, mi)
52         v1 = v0
53
54         x_half_1 = x0 + v1 * dt/2.0
55         v_half_1 = v0 + a1 * dt/2.0
56
57         a2 = self.diff_eq(x_half_1, v_half_1, t0+dt/2.0, mi)
58         v2 = v_half_1
59
60         x_half_2 = x0 + v2 * dt/2.0
61         v_half_2 = v0 + a2 * dt/2.0
62
63         a3 = self.diff_eq(x_half_2, v_half_2, t0+dt/2.0, mi)
64         v3 = v_half_2
65
66         x_end = x0 + v3 * dt
67         v_end = v0 + a3 * dt
68
69         a4 = self.diff_eq(x_end, v_end, t0 + dt, mi)
70         v4 = v_end
71
72         a_middle = 1.0/6.0 * (a1 + 2*a2 + 2*a3 + a4)
73         v_middle = 1.0/6.0 * (v1 + 2*v2 + 2*v3 + v4)
74
75         x_end = x0 + v_middle * dt
76         v_end = v0 + a_middle * dt
77
78         return x_end, v_end
79
80
81     def solve(self):
82         x = self.x; v = self.v; t = self.t; N = self.N; dt = self.
83         dt; m = self.m; dm = self.dm
84         for i in range(N-1):
85             [x[i+1], v[i+1]] = self.rk4_step(x[i], v[i], t[i], m[i]
86         )
87             t[i+1] = t[i] + dt
88             m[i+1] = m[i] + dm
89
90         return x, v, t, m
91
92 class Plotter():
93     def __init__(self):
94         a = 0

```

```

95
96     def set_parameters(self, x_values, y_values, title, xlabel,
97         ylabel, color):
98         self.x_values = x_values
99         self.y_values = y_values
100         self.title = title
101         self.xlabel = xlabel
102         self.ylabel = ylabel
103         self.color = color
104
105     def show(self):
106         plt.title(self.title)
107         plt.xlabel(self.xlabel)
108         plt.ylabel(self.ylabel)
109         plt.plot(self.x_values, self.y_values, self.color)
110         plt.show()
111
112 #
113 def diff_eq_1(x, v, t, m):
114     """
115     Differential equation for problem 1
116     """
117     m = 0.5
118     k = 1.0
119
120     a = -(1./0.5)*x
121
122     return a
123
124 #
125 def diff_eq_2(x, v, t, m):
126     """
127     Differential equation for problem 2
128     """
129     m = 0.5      # []
130     k = 1.0      # []
131     b = 0.1      # []
132
133     a = -(k/m)*x - (b/m)*v
134
135     return a
136
137 #
138 def diff_eq_4(x, v, t, m):
139     """
140     Differential equation for problem 4
141     """
142     m = 0.5      # []
143     k = 1.0      # []
144     F_D = 0.7    # N
145     w_0 = np.sqrt(k/m)
146     omega_D = 13.0/(8.0*w_0)
147
148     a = (F_D*np.cos(omega_D*t)-k*x)/m
149
150

```

```

151     return a
152
153
154
155 #
156 def diff_eq_5(x, v, t, m):
157     """
158     Differential equation for problem 5
159     """
160     m = 0.5      # []
161     k = 1.0      # []
162     b = 0.1      # []
163     F_D = 0.7    # N
164     w_0 = np.sqrt(k/m)
165     omega_D = 13.0/(8.0*w_0)
166
167     a = (F_D*np.cos(omega_D*t)-k*x-b*v)/m
168
169     return a
170
171
172 #
173 def diff_eq_6(x, v, t, m):
174     """
175     Differential equation for problem 5
176     """
177     k = 0.475    # []
178     b = 0.001    # []
179     g = 9.81     #
180
181     a = -(b*v + k*x + g)/m
182
183     print m
184
185     return a
186
187
188 solver = Solver(20.0, 1e-2)
189 plotter = Plotter()
190
191 problem_to_show = 6
192 if( problem_to_show == 1 ):
193     solver.set_diff_eq(diff_eq_1)
194     solver.set_initial_conditions(1.0, 0.0, 0.5)
195     [x, v, t, m] = solver.solve();
196     plotter.set_parameters(x, v, 'Testplot', 'x', 'y', 'r')
197     plotter.show()
198 elif( problem_to_show == 2 ):
199     solver.set_diff_eq(diff_eq_2)
200     solver.set_initial_conditions(1.0, 0.0, 0.5)
201     [x, v, t, m] = solver.solve();
202     plotter.set_parameters(x, v, 'Testplot', 'x', 'y', 'r')
203     plotter.show()
204 elif( problem_to_show == 4 ):
205     solver.set_diff_eq(diff_eq_4)
206     solver.set_time(200.0)
207     solver.set_initial_conditions(2.0, 0.0, 0.5)

```

```

208     [x, v, t, m] = solver.solve();
209     plotter.set_parameters(x, v, 'Testplot', 'x', 'y', 'r')
210     plotter.show()
211 elif( problem_to_show == 5):
212     solver.set_diff_eq(diff_eq_5)
213     solver.set_time(100.0)
214     solver.set_initial_conditions(2.0, 0.0, 0.5)
215     [x, v, t, m] = solver.solve();
216     plotter.set_parameters(x, v, 'Testplot', 'x', 'y', 'r')
217     plotter.show()
218 elif( problem_to_show == 6):
219     solver.set_diff_eq(diff_eq_6)
220     solver.set_time(3.0)
221     solver.set_dt(1e-4)
222     solver.set_dm(0.00055)
223     solver.set_initial_conditions(0.001, 0.001, 0.00001)
224     [x, v, t, m] = solver.solve();
225     plotter.set_parameters(x, v, 'Testplot', 'x', 'y', 'r')
226     plotter.show()
227 else:
228     print "Please input a valid problem numer: [1,2,4]"

```