Московский авиационный институт (национальный исследовательский университет)

Институт №7 «Робототехнические и интеллектуальные системы»

Лабораторная работа №1 по курсу теоретической механики Анимация точки

Выполнил студент группы M70-106C-22 Мастерских Егор Александрович

Преподаватель: Шамин Александр Юрьевич

Оценка:

Дата: 2 мая 2023

Вариант №12

Задание

Построить траекторию точки, заданную в полярных координатах, запустить анимацию движения точки, построить стрелки векторов скорости и ускорения, графики x(t), $v_v(t)$. Построить также центр кривизны траектории.

Закон движения точки

```
r(t) = 2 + \sin 6t\varphi(t) = 6.5t + 1.2\cos 6t
```

Текст программы

```
1 import importlib.util
 2 import argparse
3 import sympy as sp
 4 import numpy as np
 5 from math import sin, cos, tan, radians
 6 from matplotlib import rcParams
 7 import matplotlib.pyplot as plt
 8 import matplotlib.animation as animation
10 # используются для указания типа значений,
11 # возвращаемых методами matplotlib'a
12 import matplotlib.axes as axes
13
14 # +++++ СИМВОЛЬНЫЕ ВЫЧИСЛЕНИЯ +++++
15 t = sp.Symbol('t')
16
17 r = 2 + sp.sin(6 * t)
18 phi = 6.5 * t + 1.2 * sp.cos(6 * t)
20 x, y = r * sp.cos(phi), r * sp.sin(phi)
21
22 v_x, v_y = sp.diff(x, t), sp.diff(y, t)
23 v = sp.sqrt(v_x ** 2 + v_y ** 2)
24
25 a_x, a_y = sp.diff(v_x, t), sp.diff(v_y, t)
26 a_n = sp.det(sp.Matrix(
       [[v_x, v_y],
27
       [a_x, a_y]]
28
29 )) / V
30
31 R = v ** 2 / a_n # радиус кривизны траектории
```

```
32 # ---- СИМВОЛЬНЫЕ ВЫЧИСЛЕНИЯ ----
33
34 # +++++ ПЕРЕВОД В ФУНКЦИИ +++++
35 x_f = sp.lambdify(
       args=t, # аргументы, которые будет принимать функция
36
37
       expr=x, # собственно логическая часть функции
       modules='numpy'
38
39 )
40 y_f = sp.lambdify(t, y, 'numpy')
41
42 v_x = sp.lambdify(t, v_x, 'numpy')
43 v_y = sp. lambdify(t, v_y, 'numpy')
44 v_f = sp.lambdify(t, v, 'numpy')
45
46 a_x = sp.lambdify(t, a_x, 'numpy')
47 a_y_f = sp.lambdify(t, a_y, 'numpy')
48 a_n = sp. lambdify(t, a_n, 'numpy')
49
50 R_f = sp.lambdify(t, R, 'numpy')
51 # ---- ПЕРЕВОД В ФУНКЦИИ -----
52
53 # +++++ МАССИВЫ ДАННЫХ +++++
54 T_{END} = 20
55
56 t = np.linspace(0, T_END, 1000) # массив моментов времени
57
58
59 def rot_matrix(alpha):
60
61
       Возвращает матрицу поворота для заданного угла alpha (в радианах)
62
       return [[cos(alpha), -sin(alpha)],
63
               [sin(alpha), cos(alpha)]]
64
65
66
67 def rot2D(*args, alpha):
68
       Возвращает повёрнутый на заданный угол объект (массив точек)
69
70
71
       if len(args) = 1:
72
           (x, y), = args
       else:
73
74
           x, y, = args
       return np.matmul(rot_matrix(alpha), [x, y])
75
76
```

```
77
78 x = x_f(t)
79 y = y_f(t)
81 \max_x, \max_y = np.\max(x), np.\max(y)
82 \max_{plot_x} \max_{plot_y} = 1.75 * \max_x, 1.75 * \max_y
83
84 v_x, v_y, v = v_x_f(t), v_y_f(t), v_f(t)
85 v_{phi} = np.arctan2(v_y, v_x)
86
87 a_x, a_y, a_n = a_x_f(t), a_y_f(t), a_n_f(t)
88 a_phi = np.arctan2(a_y, a_x)
89
90 R = R_f(t)
91 # (R_x, R_y) --- вектор, проведённый из рассматриваемой точки
92 # в мгновенный центр кривизны траектории
93 R_x, R_y = R * rot2D(np.array([v_x, v_y]) / v, alpha=radians(90))
94 # ---- МАССИВЫ ДАННЫХ ----
95
96 # ++++ НАСТРОЙКА ОТОБРАЖЕНИЯ +++++
97 # глобальные настройки
98 rcParams['font.family'] = 'serif'
99 rcParams['font.serif'] = 'Times New Roman'
100 rcParams['mathtext.fontset'] = 'stix'
101 rcParams['font.size'] = 12
102 rcParams['axes.labelsize'] = 14
103 rcParams['axes.labelpad'] = -(rcParams['axes.labelsize']
                                  + rcParams['ytick.major.size'])
104
105 x_{abel_kw} = dict(
       horizontalalignment='left',
106
       verticalalignment='center'
107
108 )
109
110 arrow_len = 0.3 # длина стрелки
111 arrow_angle = 30 # угол раствора стрелки, в градусах
112 arrow_width = 2 * arrow_len * tan(radians(arrow_angle / 2))
113 arrow_x = np.array((-arrow_len, 0, -arrow_len))
114 arrow_y = np.array((-arrow_width / 2, 0, arrow_width / 2))
115 arrow_xy = np.array((arrow_x, arrow_y))
116
117 # если установлен пакет screeninfo,
118 # то подбираем размер окна, исходя из размеров экрана монитора
119 if importlib.util.find_spec('screeninfo'):
       from screeninfo import get_monitors
120
121
```

```
monitor = get_monitors()[0]
122
       monitor_width_in = monitor.width_mm / 25.4
123
       monitor_height_in = monitor.height_mm / 25.4
124
125
       fig_width = monitor_width_in
126
       fig_height = monitor_height_in
127 else:
128
       fig_width = fig_height = 8
129
130 fig, axd = plt.subplot_mosaic(
       [['y(x)', 'y(x)'],
131
        ['x(t)', 'v_y(t)']],
132
       height_ratios=(2, 1), # первая строка в два раза выше второй
133
134
       num='Анимация точки', # заголовок окна
       figsize=(fig_width, fig_height)
135
136 )
137
138 fig.suptitle(
       r' r'(t) = 2 + \sin 6t' + ' + ' + 5 + r' varphi(t) = 6.5t + 1.2 \cos 6t'
140 )
141
142 ax_yx: axes_axes. Axes = axd['y(x)']
143 ax_x_t: axes_axes_Axes = axd['x(t)']
144 ax_v_y_t: axes_axes_Axes = axd['v_y(t)']
145
146 # сохранение пропорций графика посредством
147 # изменения размеров ограничивающего блока
148 ax_y_x.axis('scaled')
149
150 # определение области графика, в которой будет производиться отрисовка
151 ax_y_x.set_xlim(-max_plot_x, max_plot_x)
152 ax_y_x.set_ylim(-max_plot_y, max_plot_y)
153 ax_x_t.set_xlim(0, T_END)
154 ax_v_y_t.set_xlim(0, T_END)
155
156 ax_y_x.set_xlabel(
       '$x$',
157
       x=1 + rcParams['axes.labelsize'] / ax_y_x.bbox.width,
158
       **x_label_kw
159
160 )
161 ax_y_x.set_title('$y$', loc='left')
162
163 ax_x_t.set_xlabel(
       '$t$',
164
       x=1 + rcParams['axes.labelsize'] / ax_x_t.bbox.width,
165
166
       **x_label_kw
```

```
167 )
168 ax_x_t.set_title('$x$', loc='left')
169
170 ax_v_y_t.set_xlabel(
171
        '$t$',
172
       x=1 + rcParams['axes.labelsize'] / ax_v_y_t.bbox.width,
       **x_label_kw
173
174 )
175 ax_v_y_t.set_title('$v_y$', loc='left')
176 # ---- НАСТРОЙКА ОТОБРАЖЕНИЯ ----
177
178 # +++++ ПОСТРОЕНИЯ +++++
179 ax_y_x.plot(x, y, color='tab:blue')
180
181 point = ax_y_x.plot(0, 0, marker='o', color='tab:orange')[0]
183 v_line = ax_y_x.plot(0, 0, color='tab:green')[0]
184 v_arrow = ax_y_x.plot(0, 0, color='tab:green')[0]
185
186 \max_{v_x}, \max_{v_y} = np.\max(v_x), np.\max(v_y)
187 # коэффициент уменьшения длины вектора скорости
188 v_k = max(max_v_x / max_plot_x, max_v_y / max_plot_y)
190 a_line = ax_y_x.plot(0, 0, color='tab:red')[0]
191 a_arrow = ax_y_x.plot(0, 0, color='tab:red')[0]
192
193 \max_a x, \max_a y = np.\max(a_x), np.\max(a_y)
194 # коэффициент уменьшения длины вектора ускорения
195 a_k = max(max_a_x / max_plot_x, max_a_y / max_plot_y)
197 curvature_center = ax_y_x.plot(0, 0, marker='o', color='tab:olive')[0]
198
199 ax_x_t.plot(t, x, color='tab:blue')
200 ax_v_y_t.plot(t, v_y, color='tab:blue')
201
202
203 def update(frame):
        point.set_data([x[frame]], [y[frame]])
204
205
       v_line.set_data(
206
207
            [x[frame], x[frame] + v_x[frame] / v_k],
            [y[frame], y[frame] + v_y[frame] / v_k]
208
209
210
       v_arrow.set_data(
211
            np.array([
```

```
[x[frame] + v_x[frame] / v_k],
212
                [y[frame] + v_y[frame] / v_k]
213
            ]) + rot2D(arrow_xy, alpha=v_phi[frame])
214
       )
215
216
217
       a_line.set_data(
            [x[frame], x[frame] + a_x[frame] / a_k],
218
            [y[frame], y[frame] + a_y[frame] / a_k]
219
       )
220
       a_arrow.set_data(
221
            np.array([
222
223
                [x[frame] + a_x[frame] / a_k],
224
                [y[frame] + a_y[frame] / a_k]
            ]) + rot2D(arrow_xy, alpha=a_phi[frame])
225
       )
226
227
228
       curvature_center.set_data(
229
            [x[frame] + R_x[frame]],
            [y[frame] + R_y[frame]]
230
       )
231
232
233
234 anim = animation.FuncAnimation(
235
       fig=fig,
236
       func=update, # функция, запускаемая для каждого кадра
       frames=len(t), # количество кадров
237
       interval=50, # задержка в миллисекундах между кадрами
238
239
       # задержка в миллисекундах между последовательными запусками анимации
240
       repeat_delay=3000
241 )
242
243 # уменьшаем отступы блока графиков от границ окна
244 plt.subplots_adjust(left=0.03, bottom=0.03, right=0.97, top=0.9)
245 # ---- ПОСТРОЕНИЯ ----
246
247 # +++++ СОХРАНЕНИЕ ИЛИ ОТОБРАЖЕНИЕ +++++
248 parser = argparse.ArgumentParser()
249 parser.add_argument('-s', '--save', action='store_true')
250 terminal_args = parser.parse_args()
251
252 # в зависимости от аргумента командной строки анимация
253 # либо сохраняется,
254 # либо отображается непосредственно в окне
256 if terminal_args.save:
```

```
from pathlib import Path
257
258
       anim_filepath = Path('report/animation.gif')
259
       anim.save(filename=str(anim_filepath), writer='pillow', fps=30)
260
261
262
       import logging
263
       import img2pdf
264
265
       # устанавливаем уровень логирования не ниже уровня ошибок,
       # чтобы не получать предупреждения при работе функции img2pdf.convert
266
       logging.basicConfig(level=logging.ERROR)
267
268
       pdf_frames_filepath = f'{anim_filepath.parent / anim_filepath.stem}.pdf'
269
       with open(pdf_frames_filepath, 'wb') as pdf_frames_file:
270
           pdf_frames_file.write(
271
                img2pdf.convert(str(anim_filepath))
272
           )
273
274 else:
       plt.show()
275
276 # ---- СОХРАНЕНИЕ ИЛИ ОТОБРАЖЕНИЕ ----
```

