

Московский авиационный институт
(национальный исследовательский университет)

Институт №7 «Робототехнические
и интеллектуальные системы»

**Лабораторная работа №1
по курсу теоретической механики**

Анимация точки

Выполнил студент группы М70-106С-22
Мастерских Егор Александрович

Преподаватель: Шамин Александр Юрьевич

Оценка:

Дата: 8 мая 2023

Москва 2023

Вариант №12

Задание

Построить траекторию точки, заданную в полярных координатах, запустить анимацию движения точки, построить стрелки векторов скорости и ускорения, графики $x(t)$, $v_y(t)$. 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 sympy as sp
2 import numpy as np
3 from math import sin, cos, tan, radians, sqrt
4 from matplotlib import rcParams
5 import matplotlib.pyplot as plt
6 import matplotlib.animation as animation
7 import argparse
8
9 # +++++ СИМВОЛЬНЫЕ ВЫЧИСЛЕНИЯ +++++
10 t = sp.Symbol('t')
11
12 r = 2 + sp.sin(6 * t)
13 phi = 6.5 * t + 1.2 * sp.cos(6 * t)
14
15 x, y = r * sp.cos(phi), r * sp.sin(phi)
16
17 v_x, v_y = sp.diff(x, t), sp.diff(y, t)
18 v = sp.sqrt(v_x ** 2 + v_y ** 2)
19
20 a_x, a_y = sp.diff(v_x, t), sp.diff(v_y, t)
21 a_n = sp.det(sp.Matrix(
22     [[v_x, v_y],
23     [a_x, a_y]]
24 )) / v
25
26 R = v ** 2 / a_n # радиус кривизны траектории
27 # ----- СИМВОЛЬНЫЕ ВЫЧИСЛЕНИЯ -----
28
29 # +++++ ПЕРЕВОД В ФУНКЦИИ +++++
30 x_f = sp.lambdify(
31     args=t, # аргументы, которые будет принимать функция
```

```

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

```

```

77 max_plot_x, max_plot_y = 1.75 * max_x, 1.75 * max_y
78
79 v_x, v_y, v = v_x_f(t), v_y_f(t), v_f(t)
80 v_phi = np.arctan2(v_y, v_x)
81
82 a_x, a_y, a_n = a_x_f(t), a_y_f(t), a_n_f(t)
83 a_phi = np.arctan2(a_y, a_x)
84
85 R = R_f(t)
86 # (R_x, R_y) --- вектор, проведённый из рассматриваемой точки
87 # в мгновенный центр кривизны траектории
88 R_x, R_y = R * rot2D(np.array([v_x, v_y]) / v, alpha=radians(90))
89 # ----- МАССИВЫ ДАННЫХ -----
90
91 # +++++ НАСТРОЙКА ОТОБРАЖЕНИЯ +++++
92 # глобальные настройки
93 rcParams['font.family'] = 'serif'
94 rcParams['font.serif'] = 'Times New Roman'
95 rcParams['mathtext.fontset'] = 'stix'
96 rcParams['font.size'] = 12
97 rcParams['axes.labelsize'] = 14
98 rcParams['axes.labelpad'] = -(rcParams['axes.labelsize']
99                               + rcParams['ytick.major.size'])
100 x_label_kw = dict(
101     horizontalalignment='left',
102     verticalalignment='center'
103 )
104
105 arrow_len = 0.3 # длина стрелки
106 arrow_angle = 30 # угол раствора стрелки, в градусах
107 arrow_width = 2 * arrow_len * tan(radians(arrow_angle / 2))
108 arrow_x = np.array((-arrow_len, 0, -arrow_len))
109 arrow_y = np.array((-arrow_width / 2, 0, arrow_width / 2))
110 arrow_xy = np.array((arrow_x, arrow_y))
111
112 fig, axd = plt.subplot_mosaic(
113     [['y(x)', 'y(x)'],
114     ['x(t)', 'v_y(t)']],
115     height_ratios=(2, 1), # первая строка в два раза выше второй
116     num='Анимация точки', # заголовок окна
117     # заведомо большой размер (затем окно примет размеры экрана)
118     figsize=[40] * 2
119 )
120
121 fig.suptitle(

```

```

122     r'$r(t) = 2 + \sin 6t$;' + ' ' * 5 + r'$\varphi(t) = 6.5t + 1.2\cos 6t$'
123 )
124
125 ax_y_x = axd['y(x)']
126 ax_x_t = axd['x(t)']
127 ax_v_y_t = axd['v_y(t)']
128
129 # сохранение пропорций графика посредством
130 # изменения размеров ограничивающего блока
131 ax_y_x.axis('scaled')
132
133 # определение области графика, в которой будет производиться отрисовка
134 ax_y_x.set_xlim(-max_plot_x, max_plot_x)
135 ax_y_x.set_ylim(-max_plot_y, max_plot_y)
136 ax_x_t.set_xlim(0, T_END)
137 ax_v_y_t.set_xlim(0, T_END)
138
139 ax_y_x.set_xlabel(
140     '$x$',
141     x=1 + rcParams['axes.labelsize'] / ax_y_x.bbox.width,
142     **x_label_kw
143 )
144 ax_y_x.set_title('$y$', loc='left')
145
146 ax_x_t.set_xlabel(
147     '$t$',
148     x=1 + rcParams['axes.labelsize'] / ax_x_t.bbox.width,
149     **x_label_kw
150 )
151 ax_x_t.set_title('$x$', loc='left')
152
153 ax_v_y_t.set_xlabel(
154     '$t$',
155     x=1 + rcParams['axes.labelsize'] / ax_v_y_t.bbox.width,
156     **x_label_kw
157 )
158 ax_v_y_t.set_title('$v_y$', loc='left')
159 # ----- НАСТРОЙКА ОТОБРАЖЕНИЯ -----
160
161 # +++++ ПОСТРОЕНИЯ +++++
162 ax_y_x.plot(x, y, color='tab:blue')
163
164 point = ax_y_x.plot(0, 0, marker='o', color='tab:orange')[0]
165
166 v_line = ax_y_x.plot(0, 0, color='tab:green')[0]

```

```

167 v_arrow = ax_y_x.plot(0, 0, color='tab:green')[0]
168
169 max_v_x, max_v_y = np.max(v_x), np.max(v_y)
170 # коэффициент уменьшения длины вектора скорости
171 v_k = max(max_v_x / max_plot_x, max_v_y / max_plot_y)
172
173 a_line = ax_y_x.plot(0, 0, color='tab:red')[0]
174 a_arrow = ax_y_x.plot(0, 0, color='tab:red')[0]
175
176 max_a_x, max_a_y = np.max(a_x), np.max(a_y)
177 # коэффициент уменьшения длины вектора ускорения
178 a_k = max(max_a_x / max_plot_x, max_a_y / max_plot_y)
179
180 curvature_center = ax_y_x.plot(0, 0, marker='o', color='tab:olive')[0]
181
182 ax_x_t.plot(t, x, color='tab:blue')
183 ax_v_y_t.plot(t, v_y, color='tab:blue')
184
185
186 def update(frame):
187     point.set_data([x[frame]], [y[frame]])
188
189     v_line.set_data(
190         [x[frame], x[frame] + v_x[frame] / v_k],
191         [y[frame], y[frame] + v_y[frame] / v_k]
192     )
193     v_arrow.set_data(
194         np.array([
195             [x[frame] + v_x[frame] / v_k],
196             [y[frame] + v_y[frame] / v_k]
197         ]) + rot2D(arrow_xy, alpha=v_phi[frame])
198     )
199
200     a_line.set_data(
201         [x[frame], x[frame] + a_x[frame] / a_k],
202         [y[frame], y[frame] + a_y[frame] / a_k]
203     )
204     a_arrow.set_data(
205         np.array([
206             [x[frame] + a_x[frame] / a_k],
207             [y[frame] + a_y[frame] / a_k]
208         ]) + rot2D(arrow_xy, alpha=a_phi[frame])
209     )
210
211     curvature_center.set_data(

```

```

212         [x[frame] + R_x[frame]],
213         [y[frame] + R_y[frame]]
214     )
215
216
217 anim = animation.FuncAnimation(
218     fig=fig,
219     func=update, # функция, запускаемая для каждого кадра
220     frames=len(t), # количество кадров
221     interval=50, # задержка в миллисекундах между кадрами
222     # задержка в миллисекундах между последовательными запусками анимации
223     repeat_delay=3000
224 )
225
226 # уменьшаем отступы блока графиков от границ окна
227 plt.subplots_adjust(left=0.03, bottom=0.03, right=0.97, top=0.9)
228 # ----- ПОСТРОЕНИЯ -----
229
230 # +++++ СОХРАНЕНИЕ ИЛИ ОТОБРАЖЕНИЕ +++++
231 parser = argparse.ArgumentParser()
232 parser.add_argument('-s', '--save', action='store_true')
233 terminal_args = parser.parse_args()
234
235 # в зависимости от аргумента командной строки анимация
236 # либо сохраняется,
237 # либо отображается непосредственно в окне
238
239 if terminal_args.save:
240     from pathlib import Path
241
242     A4_inches = (8.25, 8.25 * sqrt(2))
243     fig.set_size_inches(*reversed(A4_inches))
244     anim_filepath = Path('анимация_точки.gif')
245     anim.save(filename=str(anim_filepath), writer='pillow', fps=30)
246
247     import logging
248     import img2pdf
249
250     # устанавливаем уровень логирования не ниже уровня ошибок,
251     # чтобы не получать предупреждения при работе функции img2pdf.convert
252     logging.basicConfig(level=logging.ERROR)
253
254     pdf_frames_filepath = f'{anim_filepath.parent / anim_filepath.stem}.pdf'
255     with open(pdf_frames_filepath, 'wb') as pdf_frames_file:
256         pdf_frames_file.write(

```

```
257         img2pdf.convert(str(anim_filepath))
258     )
259 else:
260     plt.show()
261 # ----- СОХРАНЕНИЕ ИЛИ ОТОБРАЖЕНИЕ -----
```


$$r(t) = 2 + \sin 6t; \quad \varphi(t) = 6.5t + 1.2\cos 6t$$

