Московский авиационный институт
(национальный исследовательский университет)

Институт №7 «Робототехнические
и интеллектуальные системы»

Лабораторная работа №1
по курсу теоретической механики

Анимация точки

Выполнил студент группы М7О-106С-22
Мастерских Егор Александрович

Преподаватель: Шамин Александр Юрьевич

Оценка:

Дата: 1 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

```
import argparse
from pathlib import Path
import os
import sympy as sp
import numpy as np
from math import sin, cos, tan, radians
from screeninfo import get_monitors
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.animation as animation

# используются для указания типа значений,
# возвращаемых методами matplotlib'a
import matplotlib.figure as figure
import matplotlib.axes as axes
import matplotlib.lines as lines

# +++++ СИМВОЛЬНЫЕ ВЫЧИСЛЕНИЯ +++++
t = sp.Symbol('t')

r = 2 + sp.sin(6 * t)
phi = 6.5 * t + 1.2 * sp.cos(6 * t)

x, y = r * sp.cos(phi), r * sp.sin(phi)

v_x, v_y = sp.diff(x, t), sp.diff(y, t)
v = sp.sqrt(v_x ** 2 + v_y ** 2)

a_x, a_y = sp.diff(v_x, t), sp.diff(v_y, t)
a_n = sp.det(sp.Matrix(
    [[v_x, v_y],
```

```
     [a_x, a_y]]
)) / v

R = v ** 2 / a_n  # радиус кривизны траектории
# ----- СИМВОЛЬНЫЕ ВЫЧИСЛЕНИЯ -----

# +++++ ПЕРЕВОД В ФУНКЦИИ +++++
x_f = sp.lambdify(
    args=t,  # аргументы, которые будет принимать функция
    expr=x,  # собственно логическая часть функции
    modules='numpy'
)
y_f = sp.lambdify(t, y, 'numpy')

v_x_f = sp.lambdify(t, v_x, 'numpy')
v_y_f = sp.lambdify(t, v_y, 'numpy')
v_f = sp.lambdify(t, v, 'numpy')

a_x_f = sp.lambdify(t, a_x, 'numpy')
a_y_f = sp.lambdify(t, a_y, 'numpy')
a_n_f = sp.lambdify(t, a_n, 'numpy')

R_f = sp.lambdify(t, R, 'numpy')
# ----- ПЕРЕВОД В ФУНКЦИИ -----

# +++++ МАССИВЫ ДАННЫХ +++++
T_END = 20

t = np.linspace(0, T_END, 1000)  # массив моментов времени


def rot_matrix(alpha):
    """
    Возвращает матрицу поворота для заданного угла alpha (в радианах)
    """
    return [[cos(alpha), -sin(alpha)],
            [sin(alpha), cos(alpha)]]


def rot2D(*args, alpha):
    """
    Возвращает повёрнутый на заданный угол объект (массив точек)
    """
    if len(args) == 1:
        (x, y), = args
```

```
    else:
        x, y, = args
    return np.matmul(rot_matrix(alpha), [x, y])


x = x_f(t)
y = y_f(t)

max_x, max_y = np.max(x), np.max(y)
max_plot_x, max_plot_y = 1.75 * max_x, 1.75 * max_y

v_x, v_y, v = v_x_f(t), v_y_f(t), v_f(t)
v_phi = np.arctan2(v_y, v_x)

max_v_x, max_v_y = np.max(v_x), np.max(v_y)
if max_v_x > max_plot_x or max_v_y > max_plot_y:
    # коэффициент уменьшения длины вектора скорости
    k = max(max_v_x / max_plot_x, max_v_y / max_plot_y)
    v_x /= k
    v_y /= k
    v /= k

a_x, a_y, a_n = a_x_f(t), a_y_f(t), a_n_f(t)
a_phi = np.arctan2(a_y, a_x)

max_a_x, max_a_y = np.max(a_x), np.max(a_y)
if max_a_x > max_plot_x or max_a_y > max_plot_y:
    # коэффициент уменьшения длины вектора ускорения
    k = max(max_a_x / max_plot_x, max_a_y / max_plot_y)
    a_x /= k
    a_y /= k
    a_n /= k

R = R_f(t)
# (R_x, R_y) --- вектор, проведённый из рассматриваемой точки
# в мгновенный центр кривизны траектории
R_x, R_y = R * rot2D(np.array([v_x, v_y]) / v, alpha=radians(90))
# ----- МАССИВЫ ДАННЫХ -----

# +++++ НАСТРОЙКА ОТОБРАЖЕНИЯ +++++
monitor = get_monitors()[0]
monitor_width_in = monitor.width_mm / 25.4
monitor_height_in = monitor.height_mm / 25.4
fig_width = fig_height = 0.75 * min(monitor_width_in, monitor_height_in)

```

```
122 fig: figure.Figure = plt.figure(
123     num='Анимация точки',  # заголовок окна
124     figsize=(fig_width, fig_height)
125 )
126 
127 ax: axes._axes.Axes = fig.add_subplot()
128 
129 # сохранение пропорций графика вне зависимости от конфигурации окна
130 ax.axis('equal')
131 
132 # определение области графика, в которой будет производиться отрисовка
133 ax.set_xlim(-max_plot_x, max_plot_x)
134 ax.set_ylim(-max_plot_y, max_plot_y)
135 
136 font_size = matplotlib.rcParams['font.size']
137 ax.set_xlabel(
138     'x',
139     x=1 - font_size / (72 * fig_width),
140     labelpad=-3.2 * font_size,
141     fontfamily='serif',
142     fontsize='x-large',
143     fontstyle='italic'
144 )
145 ax.set_ylabel(
146     'y',
147     y=1 - 2.2 * font_size / (72 * fig_width),
148     labelpad=-3.2 * font_size,
149     rotation='horizontal',
150     fontfamily='serif',
151     fontsize='x-large',
152     fontstyle='italic'
153 )
154 
155 arrow_len = 0.3  # длина стрелки
156 arrow_angle = 30  # угол раствора стрелки, в градусах
157 arrow_width = 2 * arrow_len * tan(radians(arrow_angle / 2))
158 arrow_x = np.array((-arrow_len, 0, -arrow_len))
159 arrow_y = np.array((-arrow_width / 2, 0, arrow_width / 2))
160 arrow_xy = np.array((arrow_x, arrow_y))
161 # ----- НАСТРОЙКА ОТОБРАЖЕНИЯ -----
162 
163 ax.plot(x, y, color='tab:blue')
164 
165 point: lines.Line2D = ax.plot(0, 0, marker='o', color='tab:orange')[0]
166 
```

```
v_line = ax.plot(0, 0, color='tab:green')[0]
v_arrow = ax.plot(0, 0, color='tab:green')[0]

a_line = ax.plot(0, 0, color='tab:red')[0]
a_arrow = ax.plot(0, 0, color='tab:red')[0]

curvature_center = ax.plot(0, 0, marker='o', color='tab:olive')[0]


def update(frame):
    point.set_data([x[frame]], [y[frame]])

    v_line.set_data(
        [x[frame], x[frame] + v_x[frame]],
        [y[frame], y[frame] + v_y[frame]]
    )
    v_arrow.set_data(
        np.array([
            [x[frame] + v_x[frame]],
            [y[frame] + v_y[frame]]
        ]) + rot2D(arrow_xy, alpha=v_phi[frame])
    )

    a_line.set_data(
        [x[frame], x[frame] + a_x[frame]],
        [y[frame], y[frame] + a_y[frame]]
    )
    a_arrow.set_data(
        np.array([
            [x[frame] + a_x[frame]],
            [y[frame] + a_y[frame]]
        ]) + rot2D(arrow_xy, alpha=a_phi[frame])
    )

    curvature_center.set_data(
        [x[frame] + R_x[frame]],
        [y[frame] + R_y[frame]]
    )


anim = animation.FuncAnimation(
    fig=fig,
    func=update,  # функция, запускаемая для каждого кадра
    frames=len(t),  # количество кадров
    interval=50,  # задержка в миллисекундах между кадрами
```

```
212     # задержка в миллисекундах между последовательными запусками анимации
213     repeat_delay=3000
214 )
215 
216 parser = argparse.ArgumentParser()
217 parser.add_argument('-s', '--save', action='store_true')
218 terminal_args = parser.parse_args()
219 
220 # в зависимости от аргумента командной строки анимация
221 # либо сохраняется,
222 # либо отображается непосредственно в окне
223 
224 if terminal_args.save:
225     anim_filepath = Path('report/animation.gif')
226     anim.save(filename=str(anim_filepath), writer='pillow', fps=30)
227     os.system(
228         f'img2pdf {anim_filepath} -o '
229         f'{anim_filepath.parent / anim_filepath.stem}.pdf'
230     )
231 else:
232     plt.show()
```

Результат работы программы