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: 30 апреля 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
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

t_sp = sp.Symbol('t')

r_sp = 2 + sp.sin(6 * t_sp)
phi_sp = 6.5 * t_sp + 1.2 * sp.cos(6 * t_sp)

x_sp = r_sp * sp.cos(phi_sp)
y_sp = r_sp * sp.sin(phi_sp)
v_x_sp = sp.diff(x_sp, t_sp)
v_y_sp = sp.diff(y_sp, t_sp)
a_x_sp = sp.diff(v_x_sp, t_sp)
a_y_sp = sp.diff(v_y_sp, t_sp)

x_f = sp.lambdify(
    args=t_sp,  # аргументы, которые будет принимать функция
    expr=x_sp,  # собственно логическая часть функции
    modules='numpy'
```

```
)
y_f = sp.lambdify(args=t_sp, expr=y_sp, modules='numpy')
v_x_f = sp.lambdify(args=t_sp, expr=v_x_sp, modules='numpy')
v_y_f = sp.lambdify(args=t_sp, expr=v_y_sp, modules='numpy')
a_x_f = sp.lambdify(args=t_sp, expr=a_x_sp, modules='numpy')
a_y_f = sp.lambdify(args=t_sp, expr=a_y_sp, modules='numpy')


T_END = 20

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


def rot_matrix(phi):
    """
    :param phi: угол поворота в радианах
    :return: матрица поворота
    """
    return [[cos(phi), -sin(phi)],
            [sin(phi), cos(phi)]]


def rot2D(x: np.ndarray, y: np.ndarray, phi):
    """
    Поворачивает объект (массив точек) на заданный угол
    """
    return np.matmul(rot_matrix(phi), [x, y])


x = x_f(t)
y = y_f(t)
# TODO: коэффициент масштабирования должен подбираться автоматически
v_x = v_x_f(t) / 3
v_y = v_y_f(t) / 3
v_phi = np.arctan2(v_y, v_x)
a_x = a_x_f(t) / 20
a_y = a_y_f(t) / 20
a_phi = np.arctan2(a_y, a_x)

fig: figure.Figure = plt.figure(figsize=(10, 10))

ax: axes._axes.Axes = fig.add_subplot()

# сохранение пропорций графика вне зависимости от конфигурации окна
ax.axis('equal')
```

```
77
78 # определение области графика, в которой будет производиться отрисовка
79 # TODO: границы должны определяться из
80 #  максимального и минимального значений координат и скоростей
81 ax.set_xlim(-7, 7)
82 ax.set_ylim(-7, 7)
83
84 ax.plot(x, y, color='tab:blue')
85
86 arrow_len = 0.3  # длина стрелки
87 arrow_angle = 30  # угол раствора стрелки, в градусах
88 arrow_width = 2 * arrow_len * tan(radians(arrow_angle / 2))
89 arrow_x = np.array((-arrow_len, 0, -arrow_len))
90 arrow_y = np.array((-arrow_width / 2, 0, arrow_width / 2))
91
92 point: lines.Line2D = ax.plot(x[0], y[0], marker='o', color='tab:orange')[0]
93
94 v_line = ax.plot(
95     (x[0], x[0] + v_x[0]),
96     (y[0], y[0] + v_y[0]),
97     color='tab:green'
98 )[0]
99
100 v_arrow_x, v_arrow_y = rot2D(arrow_x, arrow_y, v_phi[0])
101 v_arrow = ax.plot(
102     x[0] + v_x[0] + v_arrow_x,
103     y[0] + v_y[0] + v_arrow_y,
104     color='tab:green'
105 )[0]
106
107 a_line = ax.plot(
108     (x[0], x[0] + a_x[0]),
109     (y[0], y[0] + a_y[0]),
110     color='tab:red'
111 )[0]
112
113 a_arrow_x, a_arrow_y = rot2D(arrow_x, arrow_y, a_phi[0])
114 a_arrow = ax.plot(
115     x[0] + a_x[0] + a_arrow_x,
116     y[0] + a_y[0] + a_arrow_y,
117     color='tab:red'
118 )[0]
119
120
121 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_x, v_arrow_y = rot2D(arrow_x, arrow_y, v_phi[frame])
    v_arrow.set_data(
        (x[frame] + v_x[frame] + v_arrow_x,),
        (y[frame] + v_y[frame] + v_arrow_y,)
    )

    a_line.set_data(
        (x[frame], x[frame] + a_x[frame]),
        (y[frame], y[frame] + a_y[frame])
    )
    a_arrow_x, a_arrow_y = rot2D(arrow_x, arrow_y, a_phi[frame])
    a_arrow.set_data(
        (x[frame] + a_x[frame] + a_arrow_x,),
        (y[frame] + a_y[frame] + a_arrow_y,)
    )


anim = animation.FuncAnimation(
    fig=fig,
    func=update,    # функция, запускаемая для каждого кадра
    frames=len(t),  # количество кадров
    interval=50,    # задержка в миллисекундах между кадрами
    # задержка в миллисекундах между последовательными запусками анимации
    repeat_delay=3000
)


parser = argparse.ArgumentParser()
parser.add_argument('-s', '--save', action='store_true')
args = parser.parse_args()

# в зависимости от аргумента командной строки анимация
# либо сохраняется,
# либо отображается непосредственно в окне

if args.save:
    anim_filepath = Path('report/animation.gif')
    anim.save(filename=str(anim_filepath), writer='pillow', fps=30)
    os.system(
```

```
167         f'img2pdf {anim_filepath} -o '
168         f'{anim_filepath.parent / anim_filepath.stem}.pdf'
169     )
170 
171 else:
172     plt.show()
```

Результат работы программы