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: 2 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 importlib.util
import argparse
import sympy as sp
import numpy as np
from math import sin, cos, tan, radians
from matplotlib import rcParams
import matplotlib.pyplot as plt
import matplotlib.animation as animation

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

# +++++ 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])

```

```
77
78 x = x_f(t)
79 y = y_f(t)
80
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']
104                               + rcParams['ytick.major.size'])
105 x_label_kw = dict(
106     horizontalalignment='left',
107     verticalalignment='center'
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'):
120     from screeninfo import get_monitors
121
```

```
    monitor = get_monitors()[0]
    monitor_width_in = monitor.width_mm / 25.4
    monitor_height_in = monitor.height_mm / 25.4
    fig_width = monitor_width_in
    fig_height = monitor_height_in
else:
    fig_width = fig_height = 8

fig, axd = plt.subplot_mosaic(
    [['y(x)', 'y(x)'],
     ['x(t)', 'v_y(t)']],
    height_ratios=(2, 1),  # первая строка в два раза выше второй
    num='Анимация точки',  # заголовок окна
    figsize=(fig_width, fig_height)
)

fig.suptitle(
    r'$r(t) = 2 + \sin 6t$;' + ' ' * 5 + r'$\varphi(t) = 6.5t + 1.2\cos 6t$'
)

ax_y_x: axes._axes.Axes = axd['y(x)']
ax_x_t: axes._axes.Axes = axd['x(t)']
ax_v_y_t: axes._axes.Axes = axd['v_y(t)']

# сохранение пропорций графика посредством
# изменения размеров ограничивающего блока
ax_y_x.axis('scaled')

# определение области графика, в которой будет производиться отрисовка
ax_y_x.set_xlim(-max_plot_x, max_plot_x)
ax_y_x.set_ylim(-max_plot_y, max_plot_y)
ax_x_t.set_xlim(0, T_END)
ax_v_y_t.set_xlim(0, T_END)

ax_y_x.set_xlabel(
    '$x$',
    x=1 + rcParams['axes.labelsize'] / ax_y_x.bbox.width,
    **x_label_kw
)
ax_y_x.set_title('$y$', loc='left')

ax_x_t.set_xlabel(
    '$t$',
    x=1 + rcParams['axes.labelsize'] / ax_x_t.bbox.width,
    **x_label_kw
```

```
)
ax_x_t.set_title('$x$', loc='left')

ax_v_y_t.set_xlabel(
    '$t$',
    x=1 + rcParams['axes.labelsize'] / ax_v_y_t.bbox.width,
    **x_label_kw
)
ax_v_y_t.set_title('$v_y$', loc='left')
# ----- НАСТРОЙКА ОТОБРАЖЕНИЯ -----

# +++++ ПОСТРОЕНИЯ +++++
ax_y_x.plot(x, y, color='tab:blue')

point = ax_y_x.plot(0, 0, marker='o', color='tab:orange')[0]

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

max_v_x, max_v_y = np.max(v_x), np.max(v_y)
# коэффициент уменьшения длины вектора скорости
v_k = max(max_v_x / max_plot_x, max_v_y / max_plot_y)

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

max_a_x, max_a_y = np.max(a_x), np.max(a_y)
# коэффициент уменьшения длины вектора ускорения
a_k = max(max_a_x / max_plot_x, max_a_y / max_plot_y)

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

ax_x_t.plot(t, x, color='tab:blue')
ax_v_y_t.plot(t, v_y, color='tab:blue')


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

    v_line.set_data(
        [x[frame], x[frame] + v_x[frame] / v_k],
        [y[frame], y[frame] + v_y[frame] / v_k]
    )
    v_arrow.set_data(
        np.array([
```

```
            [x[frame] + v_x[frame] / v_k],
            [y[frame] + v_y[frame] / v_k]
        ]) + rot2D(arrow_xy, alpha=v_phi[frame])
    )

    a_line.set_data(
        [x[frame], x[frame] + a_x[frame] / a_k],
        [y[frame], y[frame] + a_y[frame] / a_k]
    )
    a_arrow.set_data(
        np.array([
            [x[frame] + a_x[frame] / a_k],
            [y[frame] + a_y[frame] / a_k]
        ]) + 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
    # задержка в миллисекундах между последовательными запусками анимации
    repeat_delay=3000
)

# уменьшаем отступы блока графиков от границ окна
plt.subplots_adjust(left=0.03, bottom=0.03, right=0.97, top=0.9)
# ----- ПОСТРОЕНИЯ -----

# +++++ СОХРАНЕНИЕ ИЛИ ОТОБРАЖЕНИЕ +++++
parser = argparse.ArgumentParser()
parser.add_argument('-s', '--save', action='store_true')
terminal_args = parser.parse_args()

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

if terminal_args.save:
```

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

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