ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«АНИМАЦИЯ ТОЧКИ»

ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»

ВАРИАНТ ЗАДАНИЯ №16

Выполнил(а) студент группы М8О-208Б-23

Пинчук Михаил Сергеевич _____
подпись, дата

Проверил и принял

Ст. преп. каф. 802 Волков Е.В._____
подпись, дата

с оценкой _____

Москва, 2024

Задание: построить заданную траекторию, запустить анимацию движения точки, построить стрелки радиус-вектора, вектора скорости, вектора ускорения и радиуса кривизны.

Условия задачи для 16 варианта:

$r(t) = 1 + \sin(12t)$

$\varphi(t) = 1.8 * t + 0.2 * (\cos(12t))^2$

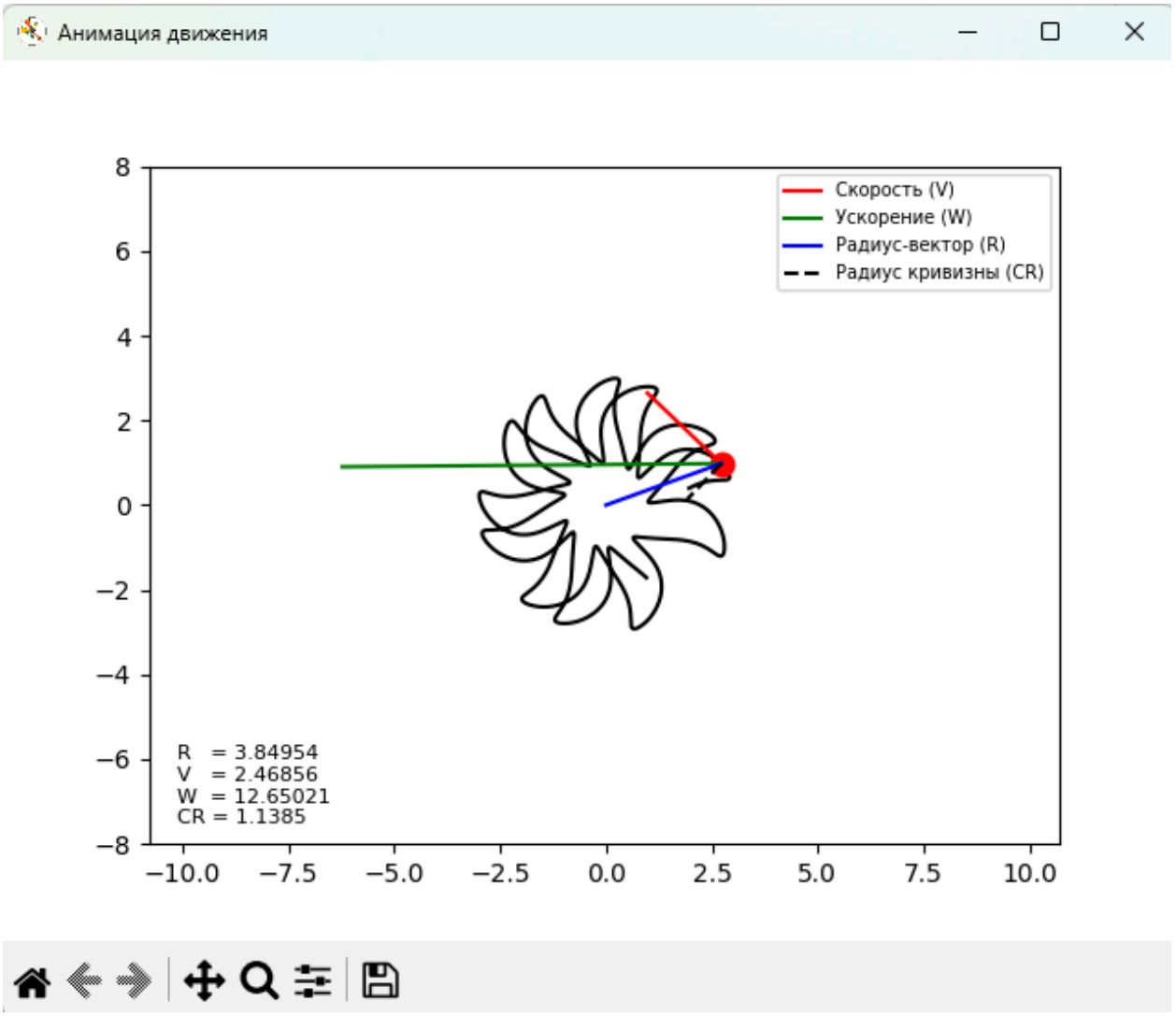
Код программы:

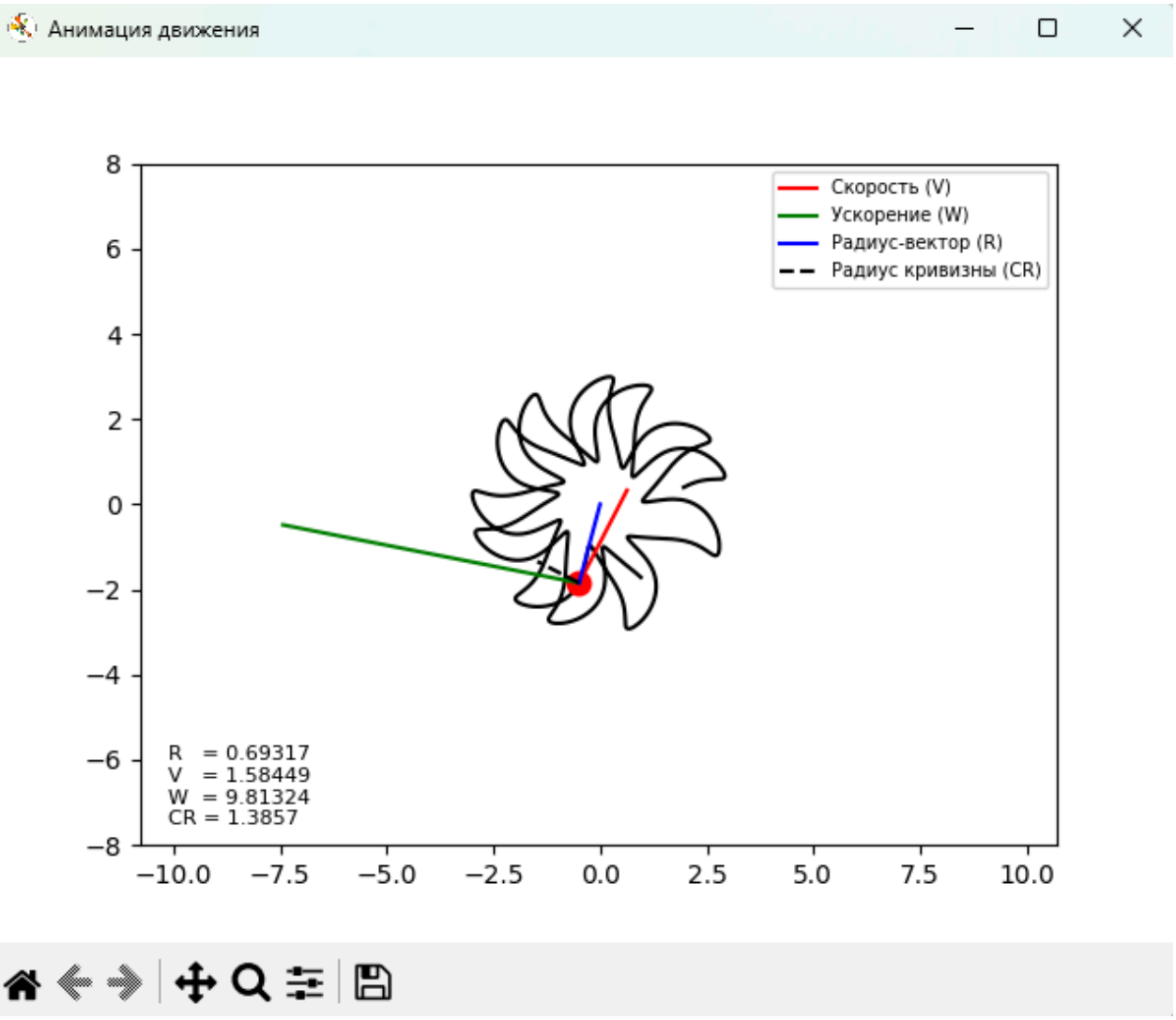
```
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import math

# Функция для вращения 2D координат на угол `angle`
def rotate_2d(x, y, angle):
    return (
        x * np.cos(angle) - y * np.sin(angle),
        x * np.sin(angle) + y * np.cos(angle)
    )

# Функция для расчета модуля вектора на основе его компонентов
def vector_magnitude(data, i):
    return round((data[i] ** 2 + data[i] ** 2) ** 0.5, 5)

# Объявляем символ для времени t
t = sp.Symbol('t')

# Параметрические уравнения: радиус r и угол phi
r = 2 + sp.sin(12 * t)  # Радиус
phi = 1.8 * t + 0.2 * (sp.cos(12 * t)) ** 2  # Угловая зависимость

# Переход к декартовым координатам
x = r * sp.cos(phi)
y = r * sp.sin(phi)

# Вычисляем скорость (V) и ускорение (W)
Vx = sp.diff(x, t)  # Проекция скорости по x
Vy = sp.diff(y, t)  # Проекция скорости по y
V = (Vx**2 + Vy**2)**0.5  # Модуль скорости
```

```
Wx = sp.diff(Vx, t)  # Проекция ускорения по x
Wy = sp.diff(Vy, t)  # Проекция ускорения по y
W = (Wx**2 + Wy**2)**0.5  # Модуль ускорения

# Нормальное и тангенциальное ускорение
Wt = sp.diff(V, t)  # Тангенциальное ускорение
Wn = (W**2 - Wt**2)**0.5  # Нормальное ускорение

# Координаты радиуса кривизны
curvature_module = ((V * V) / Wn)  # Радиус кривизны
curvature_x = curvature_module * (Wx - (Vx / V * Wt)) / Wn  # X-составляющая
curvature_y = curvature_module * (Wy - (Vy / V * Wt)) / Wn  # Y-составляющая

# Создание численных функций для t с помощью lambdify
Fx = sp.lambdify(t, x, 'numpy')
Fy = sp.lambdify(t, y, 'numpy')
FVx = sp.lambdify(t, Vx, 'numpy')
FVy = sp.lambdify(t, Vy, 'numpy')
FWx = sp.lambdify(t, Wx, 'numpy')
FWy = sp.lambdify(t, Wy, 'numpy')
F_curvature_x = sp.lambdify(t, curvature_x, 'numpy')
F_curvature_y = sp.lambdify(t, curvature_y, 'numpy')

# Временные точки для анимации
T = np.linspace(0, 6.28, 1000)

# Координаты точек траектории
X = Fx(T)
Y = Fy(T)

# Векторы скорости, ускорения и кривизны
VX = FVx(T) * 0.2
VY = FVy(T) * 0.2
WX = FWx(T) * 0.05
WY = FWy(T) * 0.05
CX = F_curvature_x(T)
CY = F_curvature_y(T)

# Создаем окно и оси для визуализации
fig, ax = plt.subplots()
fig.canvas.manager.set_window_title('Анимация движения')  # Название окна
ax.axis('equal')  # Одинаковый масштаб по осям
ax.set(ylim=[-8, 8])  # Диапазон для оси Y

# Отображаем траекторию
ax.plot(X, Y, 'black')

# Объекты для анимации: точка и векторы
point, = ax.plot(X[0], Y[0], marker='o', color='red', markersize=9)
v_line, = ax.plot([], [], 'r', label='Скорость (V)')
```

```
w_line, = ax.plot([], [], 'g', label='Ускорение (W)')
r_line, = ax.plot([], [], 'b', label='Радиус-вектор (R)')
curvature_radius, = ax.plot([], [], 'black', linestyle='--', label='Радиус
кривизны (CR)')

# Текстовая информация для анимации
text_template = 'R   = {r}\nV   = {v}\nW  = {w}\nCR = {cr}'
text = ax.text(0.03, 0.03, '', transform=ax.transAxes, fontsize=8)

# Функция анимации для обновления кадра
def animate(i):
    if i >= len(X): i = len(X) - 1

    # Обновление позиции точки
    point.set_data([X[i]], [Y[i]])

    # Вектор скорости
    v_line.set_data([X[i], X[i] + VX[i]], [Y[i], Y[i] + VY[i]])

    # Вектор ускорения
    w_line.set_data([X[i], X[i] + WX[i]], [Y[i], Y[i] + WY[i]])

    # Радиус-вектор
    r_line.set_data([0, X[i]], [0, Y[i]])

    # Радиус кривизны
    curvature_radius.set_data([X[i], X[i] + CX[i]], [Y[i], Y[i] + CY[i]])

    # Обновление текста
    text.set_text(text_template.format(
        r=vector_magnitude(X, i),
        v=vector_magnitude(VX, i),
        w=vector_magnitude(WX, i),
        cr=vector_magnitude(CX, i)
    ))

    return point, v_line, w_line, r_line, curvature_radius, text

# Создание анимации
animation = FuncAnimation(fig, animate, frames=1000, interval=50, blit=True)

# Легенда и отображение
ax.legend(fontsize=7)
plt.show()
```

Рисунки получившейся физической модели:


Анимация движения
Скорость (V)
Ускорение (W)
Радиус-вектор (R)
Радиус кривизны (CR)
R = 3.84954
V = 2.46856
W = 12.65021
CR = 1.1385
8
6
4
2
0
−2
−4
−6
−8
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0



Вектор скорости — красным цветом

Вектор ускорения — зеленым цветом

Радиус-вектор — синим цветом

Радиус кривизны — черным пунктиром

Вывод:

В ходе выполнения этой лабораторной работы я написал Python код, который строит заданную траекторию и запускает анимацию движения точки. Также были построены стрелки радиус вектора, вектора скорости и вектора ускорения.

Эта работа помогла мне лучше понять основы теоретической механики и применение ее в программировании. Я узнал, как использовать

математические формулы для решения задач движения тела и как визуализировать результаты с помощью анимации.

Кроме того, я научился работать с графическими библиотеками Python, такими как Matplotlib и NumPy. Эти инструменты позволили мне создать графики и анимации.