ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ ТОЧКИ»**

**ПО ДИСЦИПЛИНЕ «ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ»**

**ВАРИАНТ ЗАДАНИЯ №21**

Выполнил(а) студент группы М8О-201Б-23

Терентьев М.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Ст. преп. каф. 802 Волков Е.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**Вариант №21 «Фантастическая кривая»**

**Задание:**

Построить заданную траекторию и анимацию движения точки, а также отобразить стрелки радиус-вектора, скорости и ускорения. Построить радиус кривизны траектории.

**Закон движения точки:**

**Текст программы**

import numpy as np

import sympy as sp

import matplotlib.pyplot as plt

import matplotlib.animation as animation

import math

# Создание массива времени

T = np.linspace(0, 10, 1000)

R = 1

t = sp.Symbol('t')

# Параметрические уравнения траектории

r = sp.cos(6 \* t)

fi = t + 0.2 \* sp.cos(3 \* t)

# Координаты точки на плоскости

x = R \* r \* sp.cos(fi)

y = R \* r \* sp.sin(fi)

# Первая производная по времени (скорость)

x\_diff = sp.diff(x, t)

y\_diff = sp.diff(y, t)

# Вторая производная по времени (ускорение)

x\_diff2 = sp.diff(x\_diff, t)

y\_diff2 = sp.diff(y\_diff, t)

# Инициализация массивов для хранения значений

X = np.zeros\_like(T)

Y = np.zeros\_like(T)

VX = np.zeros\_like(T)

VY = np.zeros\_like(T)

AX = np.zeros\_like(T)

AY = np.zeros\_like(T)

# Вычисление значений координат, скорости и ускорения в каждый момент времени

for i in range(len(T)):

X[i] = float(x.subs(t, T[i]))

Y[i] = float(y.subs(t, T[i]))

VX[i] = float(x\_diff.subs(t, T[i]))

VY[i] = float(y\_diff.subs(t, T[i]))

AX[i] = float(x\_diff2.subs(t, T[i]))

AY[i] = float(y\_diff2.subs(t, T[i]))

# Настройка графика

fig, ax = plt.subplots()

ax.axis('equal')

a\_lim = 0.8

ax.set\_xlim([min(X) - a\_lim, max(X) + a\_lim])

ax.set\_ylim([min(Y) - a\_lim, max(Y) + a\_lim])

# Инициализация графических элементов

point, = ax.plot([], [], 'go', markersize=10) # Точка на траектории

ax.plot(X, Y, 'r-', lw=1) # Траектория движения

velocity\_line, = ax.plot([], [], 'b-', lw=1) # Вектор скорости

velocity\_arrow\_head, = ax.plot([], [], 'b-') # Стрелка вектора скорости

acceleration\_line, = ax.plot([], [], 'g-', lw=1) # Вектор ускорения

acceleration\_arrow\_head, = ax.plot([], [], 'g-') # Стрелка вектора ускорения

radius\_vector\_line, = ax.plot([], [], 'y-', lw=1) # Радиус-вектор

radius\_vector\_arrow\_head, = ax.plot([], [], 'y-') # Стрелка радиус-вектора

curvature\_radius\_line, = ax.plot([], [], 'm--', lw=1) # Радиус кривизны

curvature\_radius\_arrow\_head, = ax.plot([], [], 'm--') # Стрелка радиуса кривизны

def rotate\_2d(x\_arr, y\_arr, angle):

# Функция для поворота точки на заданный угол

x\_new = x\_arr \* np.cos(angle) - y\_arr \* np.sin(angle)

y\_new = x\_arr \* np.sin(angle) + y\_arr \* np.cos(angle)

return x\_new, y\_new

def update(frame):

# Обновление данных для текущего кадра анимации

x0 = X[frame]

y0 = Y[frame]

vx = VX[frame]

vy = VY[frame]

ax0 = AX[frame]

ay0 = AY[frame]

# Обновление позиции точки

point.set\_data([x0], [y0])

# Обновление вектора скорости

velocity\_line.set\_data([x0, x0 + vx], [y0, y0 + vy])

angle\_v = math.atan2(vy, vx)

arrow\_x = np.array([-0.08, 0, -0.08]) # Координаты стрелки

arrow\_y = np.array([0.04, 0, -0.04])

VArrowX, VArrowY = rotate\_2d(arrow\_x, arrow\_y, angle\_v)

velocity\_arrow\_head.set\_data(VArrowX + x0 + vx, VArrowY + y0 + vy)

# Обновление вектора ускорения

acceleration\_line.set\_data([x0, x0 + ax0], [y0, y0 + ay0])

angle\_a = math.atan2(ay0, ax0)

AArrowX, AArrowY = rotate\_2d(arrow\_x, arrow\_y, angle\_a)

acceleration\_arrow\_head.set\_data(AArrowX + x0 + ax0, AArrowY + y0 + ay0)

# Обновление радиус-вектора

radius\_vector\_line.set\_data([0, x0], [0, y0])

angle\_r = math.atan2(y0, x0)

RArrowX, RArrowY = rotate\_2d(arrow\_x, arrow\_y, angle\_r)

radius\_vector\_arrow\_head.set\_data(RArrowX + x0, RArrowY + y0)

# Вычисление радиуса кривизны

numerator = (vx\*\*2 + vy\*\*2)\*\*1.5

denominator = abs(vx \* ay0 - vy \* ax0)

if denominator != 0:

R\_curv = numerator / denominator

else:

R\_curv = np.inf

# Нормальный вектор для определения центра кривизны

norm\_vx = -vy

norm\_vy = vx

norm = np.hypot(norm\_vx, norm\_vy)

if norm != 0:

norm\_vx /= norm

norm\_vy /= norm

# Координаты центра кривизны

center\_x = x0 + R\_curv \* norm\_vx

center\_y = y0 + R\_curv \* norm\_vy

# Обновление линии радиуса кривизны

curvature\_radius\_line.set\_data([x0, center\_x], [y0, center\_y])

angle\_c = math.atan2(center\_y - y0, center\_x - x0)

CArrowX, CArrowY = rotate\_2d(arrow\_x, arrow\_y, angle\_c)

curvature\_radius\_arrow\_head.set\_data(

CArrowX + center\_x, CArrowY + center\_y)

return (point,

velocity\_line, velocity\_arrow\_head,

acceleration\_line, acceleration\_arrow\_head,

radius\_vector\_line, radius\_vector\_arrow\_head,

curvature\_radius\_line, curvature\_radius\_arrow\_head)

# Создание анимации

ani = animation.FuncAnimation(

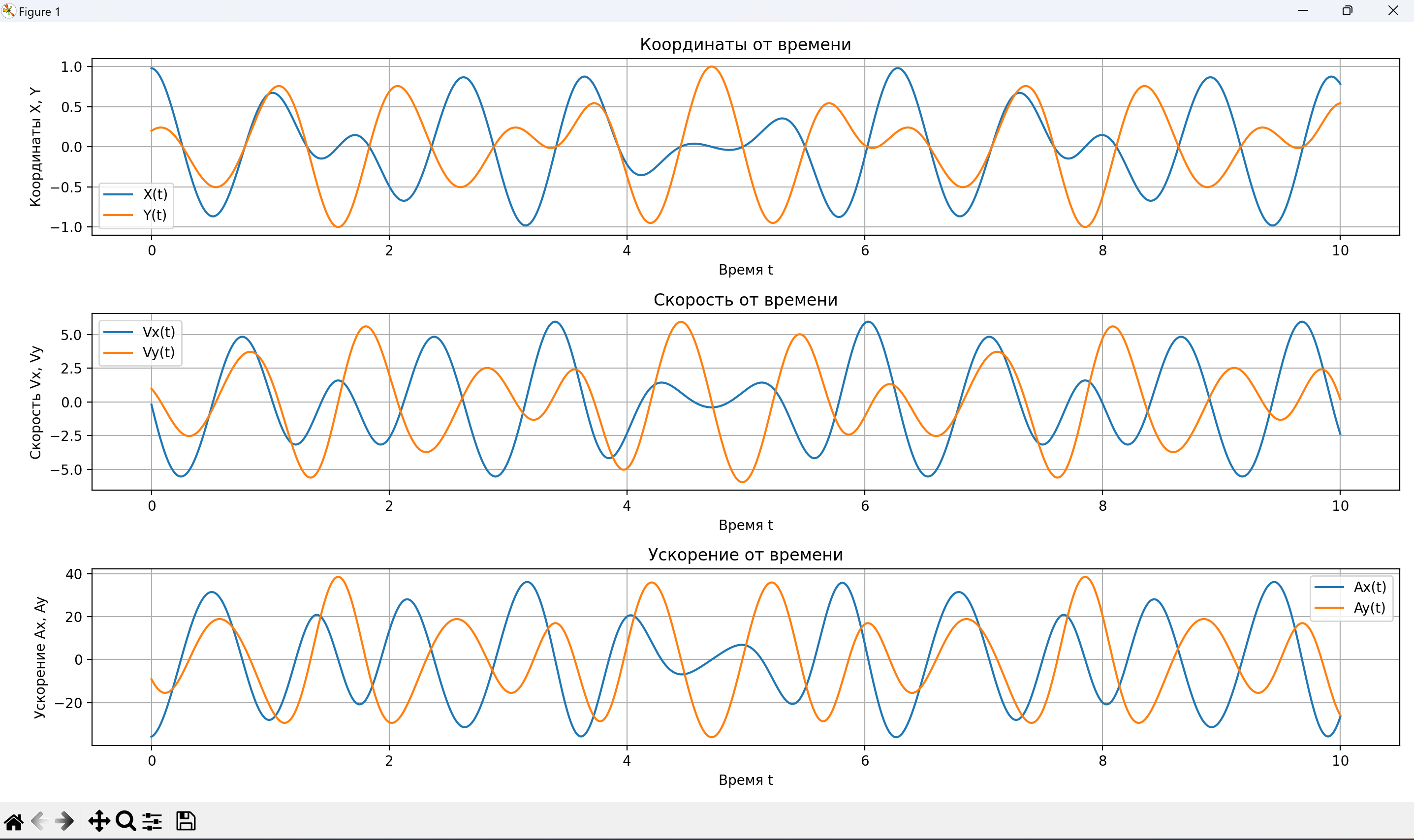
fig, update, frames=len(T), interval=20, blit=True)

plt.show()

**Результат работы программы:**

|  |  |
| --- | --- |
|  |  |

**Результаты работы программы:**



**Вывод:** В данной работе была разработана программа на Python для моделирования движения точки по заданной траектории. С помощью библиотеки SymPy получены аналитические выражения для координат, скорости и ускорения точки. Используя NumPy, вычислены численные значения этих величин на заданном интервале времени.

Анимация, созданная с помощью Matplotlib, наглядно продемонстрировала движение точки и изменение её кинематических параметров. Одновременно были построены графики зависимостей координат, компонент скорости и ускорения от времени.

Работа показала практическую ценность комбинирования символических и численных методов вычислений, а также важность визуализации при изучении кинематики точки.