ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«Фантастическая кривая»**

**ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»**

**ВАРИАНТ ЗАДАНИЯ №11**

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

Ермеков Георгий Александрович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

подпись, дата

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

**Москва, 2024**

**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 sympy as sp

import matplotlib.animation as animation

import matplotlib.pyplot as plt

import numpy as np

import math

def Rot2D(X, Y, Alpha):

"""Функция для поворота"""

RX = X \* np.cos(Alpha) - Y \* np.sin(Alpha)

RY = X \* np.sin(Alpha) + Y \* np.cos(Alpha)

return RX, RY

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

t = sp.Symbol("t")

# условние

R = 1

r = 2 + sp.cos(6 \* t)

fi = 7 \* t + 1.2 \* sp.cos(6 \* t)

# перевод из полярных в декартовы координаты

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

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

# вычисление скорости и ускорения

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

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

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

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

# массивы с значениями в каждый момент времени t

X = np.zeros\_like(T)

Y = np.zeros\_like(T)

VX = np.zeros\_like(T)

VY = np.zeros\_like(T)

VVX = np.zeros\_like(T)

VVY = np.zeros\_like(T)

# расчет значений

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]))

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

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

fig, ax = plt.subplots()

ax.axis("equal")

ax.set\_xlim([min(X) - 0.5, max(X) + 0.5])

ax.set\_ylim([min(Y) - 0.5, max(Y) + 0.5])

SPD\_V\_REDUCE\_KOEF = 3

ACC\_V\_REDUCE\_KOEF = 30

ArrowX = np.array([-0.2 \* R, 0, -0.2 \* R])

ArrowY = np.array([0.1 \* R, 0, -0.1 \* R])

# стрелки

RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[0], VX[0]))

# стрелка вектора скорости

(VArrow,) = ax.plot(RArrowX + X[0] + VX[0] / 3, RArrowY + Y[0] + VY[0] / 3, "b")

# стрелка вектора ускорения

(VVArrow,) = ax.plot(RArrowX + X[0] + VVX[0] / 3, RArrowY + Y[0] + VVY[0] / 3, "g")

# стрелка радиус-вектора

(RadArrow,) = ax.plot(RArrowX + X[0], RArrowY + Y[0], "r")

(point,) = ax.plot([], [], "go", markersize=10)

ax.plot(X, Y, "r-", lw=1)

# линия вектора скорости

(VLine,) = ax.plot([], [], "b-", lw=1)

# линия вектора ускорения

(VVLine,) = ax.plot([], [], "g-", lw=1)

# линия радиус-вектора

(RadLine,) = ax.plot([], [], "r-", lw=1)

def init():

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

return (point,)

def update(frame):

"""Функция для отрисовки нового кадра"""

point.set\_data([X[frame]], [Y[frame]])

RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[frame], VX[frame]))

VArrow.set\_data(

RArrowX + X[frame] + VX[frame] / SPD\_V\_REDUCE\_KOEF, RArrowY + Y[frame] + VY[frame] / SPD\_V\_REDUCE\_KOEF

)

VLine.set\_data(

[X[frame], X[frame] + VX[frame] / SPD\_V\_REDUCE\_KOEF], [Y[frame], Y[frame] + VY[frame] / SPD\_V\_REDUCE\_KOEF]

)

VRArrowX, VRArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VVY[frame], VVX[frame]))

VVArrow.set\_data(

VRArrowX + X[frame] + VVX[frame] / ACC\_V\_REDUCE\_KOEF, VRArrowY + Y[frame] + VVY[frame] / ACC\_V\_REDUCE\_KOEF

)

VVLine.set\_data(

[X[frame], X[frame] + VVX[frame] / ACC\_V\_REDUCE\_KOEF], [Y[frame], Y[frame] + VVY[frame] / ACC\_V\_REDUCE\_KOEF]

)

RadArrowX, RadArrowY = Rot2D(ArrowX, ArrowY, math.atan2(Y[frame], X[frame]))

RadArrow.set\_data(RadArrowX + X[frame], RadArrowY + Y[frame])

RadLine.set\_data([0, X[frame]], [0, Y[frame]])

return point, VLine, VArrow, VVLine, VVArrow, RadArrow, RadLine

# запуск анимации

ani = animation.FuncAnimation(fig, update, frames=len(T), init\_func=init, interval=40, blit=True)

plt.show()

**Результат работы программы:**

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

**Вывод:** в ходе лабораторной работы мы освоили работу с библиотеками python для расчетов и визуализации. В результате была создана программа, которая отображает траекторию движения точки по закону, заданному формулой согласно варианту, её скорость, ускорение и радиус-вектор к этой точки.