ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ ТОЧКИ»**

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

**ВАРИАНТ ЗАДАНИЯ №32**

Выполнил(а) студент группы  М80-201Б-22

Парфенов Михаил Максимович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Зав. каф. 802, Бардин Б.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

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

Москва, 2023

*Задание:* построить анимацию движения системы с помощью Python.

Закон движения: r = cos(6t), phi = t + 0.2\*cos(3t).

Текст программы:

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

import sympy as sp

import math

#вариант 21

#r = cos(6t)

#phi = t + 0.2\*cos(3t)

#поворот на угол alpha

def rotate(X, Y, Alpha):

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

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

return RX, RY

def angle(a\_x, a\_y, b\_x, b\_y):

ab = a\_x \* b\_x + a\_y \* b\_y

mod\_a = math.sqrt(a\_x \* a\_x + a\_y \* a\_y)

mod\_b = math.sqrt(b\_x \* b\_x + b\_y \* b\_y)

return ab / (mod\_a \* mod\_b)

t = sp.Symbol('t')

#перевод полярных координат в Декартовы

x = (sp.cos(6\*t))\*sp.cos(t + 0.2\*sp.cos(3\*t))

y = (sp.cos(6\*t))\*sp.sin(t + 0.2\*sp.cos(3\*t))

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

#скорость

v\_x = sp.diff(x, t)

v\_y = sp.diff(y, t)

v\_mod = sp.sqrt(v\_x\*v\_x + v\_y\*v\_y)

#ускорение

w\_x = sp.diff(v\_x, t)

w\_y = sp.diff(v\_y, t)

w\_mod = sp.sqrt(w\_x\*w\_x + w\_y\*w\_y)

#эволюта (радиус кривизны)

evo\_x = -sp.diff(y, t)\*(sp.diff(x)\*\*2 + sp.diff(y)\*\*2)/(sp.diff(x, t)\*sp.diff(y, t, 2) - sp.diff(x, t, 2)\*sp.diff(y, t))

evo\_y = sp.diff(x, t)\*(sp.diff(x)\*\*2 + sp.diff(y)\*\*2)/(sp.diff(x, t)\*sp.diff(y, t, 2) - sp.diff(x, t, 2)\*sp.diff(y, t))

evo\_mod = sp.sqrt(evo\_x\*evo\_x + evo\_y\*evo\_y)

#тангенсальное ускорение

cos\_w\_v = (v\_x \* w\_x + v\_y \* w\_y) / (v\_mod \* w\_mod)

w\_tau\_x = (v\_x / v\_mod) \* w\_mod \* cos\_w\_v

w\_tau\_y = (v\_y / v\_mod) \* w\_mod \* cos\_w\_v

#нормальное ускорение

cos\_w\_evo = sp.sqrt(1 - cos\_w\_v\*cos\_w\_v)

w\_nor\_x = (evo\_x / evo\_mod) \* w\_mod \* cos\_w\_evo

w\_nor\_y = (evo\_y / evo\_mod) \* w\_mod \* cos\_w\_evo

#1000 чисел от 1 до 10

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

#куча нулевых массивов "как" массив Т

X = np.zeros\_like(T)

Y = np.zeros\_like(T)

V\_X = np.zeros\_like(T)

V\_Y = np.zeros\_like(T)

W\_X = np.zeros\_like(T)

W\_Y = np.zeros\_like(T)

W\_N\_X = np.zeros\_like(T)

W\_N\_Y = np.zeros\_like(T)

Phi = np.zeros\_like(T)

#вычисление значений функций в каждой точке Т

for i in np.arange(len(T)):

X[i] = sp.Subs(x, t, T[i])

Y[i] = sp.Subs(y, t, T[i])

V\_X[i] = sp.Subs(v\_x, t, T[i])

V\_Y[i] = sp.Subs(v\_y, t, T[i])

W\_X[i] = sp.Subs(w\_tau\_x, t, T[i])

W\_Y[i] = sp.Subs(w\_tau\_y, t, T[i])

W\_N\_X[i] = sp.Subs(w\_nor\_x, t, T[i])

W\_N\_Y[i] = sp.Subs(w\_nor\_y, t, T[i])

Phi[i] = sp.Subs(phi, t, T[i])

#создание окна, поля с графиком и его параметров

fig = plt.figure(figsize=(10, 5))

ax1 = fig.add\_subplot(1, 1, 1)

ax1.axis('equal')

ax1.set\_title("Модель движения точки")

ax1.set\_xlabel('X')

ax1.set\_ylabel('Y')

ax1.plot(X, Y)

P, = ax1.plot(X[0], Y[0], 'black', marker='o') #движущаяся точка

V\_Line, = ax1.plot([X[0], X[0] + V\_X[0]], [Y[0], X[0] + V\_X[0]], 'r', label = 'Скорость')

W\_Line, = ax1.plot([X[0], X[0] + W\_X[0]], [Y[0], Y[0] + W\_Y[0]], 'g', label = 'Тангенсальное ускорение')

R\_Line, = ax1.plot([0, X[0]], [0, Y[0]], 'black', label = 'Радиус-вектор')

W\_N\_Line, = ax1.plot([X[0], X[0] + W\_N\_X[0]], [Y[0], Y[0] + W\_N\_Y[0]], 'y', label = 'Нормальное ускорение')

R = math.sqrt(math.pow(X[0], 2) + math.pow(Y[0], 2))

#построение стрелочек

#для скорости

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

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

R\_Arrow\_X, R\_Arrow\_Y = rotate(Arrow\_X, Arrow\_Y, math.atan2(V\_Y[0], V\_X[0]))

V\_Arrow, = ax1.plot(R\_Arrow\_X + X[0] + V\_X[0], R\_Arrow\_Y + Y[0] + V\_Y[0], 'r')

#для тангенсального ускорения

W\_Arrow\_X = np.array([-0.2\*R, 0, -0.2\*R])

W\_Arrow\_Y = np.array([0.1\*R, 0, -0.1\*R])

R\_W\_Arrow\_X, R\_W\_Arrow\_Y = rotate(W\_Arrow\_X, W\_Arrow\_Y, math.atan2(W\_Y[0], W\_X[0]))

W\_Arrow, = ax1.plot(R\_W\_Arrow\_X + X[0] + W\_X[0], R\_W\_Arrow\_Y + Y[0] + W\_Y[0], 'g')

#для радиус вектора

Arrow\_R\_X = np.array([-0.2\*R, 0, -0.2\*R])

Arrow\_R\_Y = np.array([0.1\*R, 0, -0.1\*R])

R\_Arrow\_RX, R\_Arrow\_RY = rotate(Arrow\_R\_X, Arrow\_R\_Y, math.atan2(Y[0], X[0]))

R\_Arrow, = ax1.plot(R\_Arrow\_RX + X[0], R\_Arrow\_RY + Y[0], 'black')

#для нормального ускорения

W\_N\_Arrow\_X = np.array([-0.2\*R, 0, -0.2\*R])

W\_N\_Arrow\_Y = np.array([0.1\*R, 0, -0.1\*R])

R\_W\_N\_Arrow\_X, R\_W\_N\_Arrow\_Y = rotate(W\_N\_Arrow\_X, W\_N\_Arrow\_Y, math.atan2(W\_N\_Y[0], W\_N\_X[0]))

W\_N\_Arrow, = ax1.plot(R\_W\_N\_Arrow\_X + X[0] + W\_N\_X[0], R\_W\_N\_Arrow\_Y + Y[0] + W\_N\_Y[0], 'y')

ax1.legend(loc='lower center', ncol = 5, edgecolor = 'b')

ax1.set(xlim=[-4, 4], ylim=[-4, 4])

#функция анимацииУ

def anima(i):

P.set\_data(X[i], Y[i])

W\_Line.set\_data([X[i], X[i] + W\_X[i]], [Y[i],Y[i] + W\_Y[i]])

V\_Line.set\_data([X[i], X[i] + V\_X[i]], [Y[i], Y[i] + V\_Y[i]])

R\_Line.set\_data([0, X[i]], [0, Y[i]])

W\_N\_Line.set\_data([X[i], X[i] + W\_N\_X[i]], [Y[i], Y[i] + W\_N\_Y[i]])

R\_Arrow\_X, R\_Arrow\_Y = rotate(Arrow\_X, Arrow\_Y, math.atan2(V\_Y[i], V\_X[i]))

R\_W\_Arrow\_X, R\_W\_Arrow\_Y = rotate(W\_Arrow\_X, W\_Arrow\_Y, math.atan2(W\_Y[i], W\_X[i]))

R\_Arrow\_RX, R\_Arrow\_RY = rotate(Arrow\_R\_X, Arrow\_R\_Y, math.atan2(Y[i], X[i]))

R\_W\_N\_Arrow\_X, R\_W\_N\_Arrow\_Y = rotate(W\_N\_Arrow\_X, W\_N\_Arrow\_Y, math.atan2(W\_N\_Y[i], W\_N\_X[i]))

V\_Arrow.set\_data(R\_Arrow\_X + X[i] + V\_X[i], R\_Arrow\_Y + Y[i] + V\_Y[i])

W\_Arrow.set\_data(R\_W\_Arrow\_X + X[i] + W\_X[i], R\_W\_Arrow\_Y + Y[i] + W\_Y[i])

R\_Arrow.set\_data(R\_Arrow\_RX + X[i], R\_Arrow\_RY + Y[i])

W\_N\_Arrow.set\_data(R\_W\_N\_Arrow\_X + W\_N\_X[i], R\_W\_N\_Arrow\_Y + Y[i] + W\_N\_Y[i])

return P, V\_Line, W\_Line, R\_Line, W\_N\_Line, V\_Arrow, W\_Arrow, R\_Arrow, W\_N\_Arrow

#вывести график

anim = FuncAnimation(fig, anima, frames=1000, interval=100, blit=True)

plt.show()

Результат работы программы:

  
  
  
