ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ОТЧЕТ

О ВЫПОЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«АНИМАЦИЯ ТОЧКИ»

ПО ДИСЦИПЛИНЕ «ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ»

ВАРИАНТ ЗАДАНИЯ №20

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

Соловьева Надежда Сергеевна _____

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:

$r(t) = 1 + cos(t);\ \varphi(t) = 1.25\ t$

Текст программы:

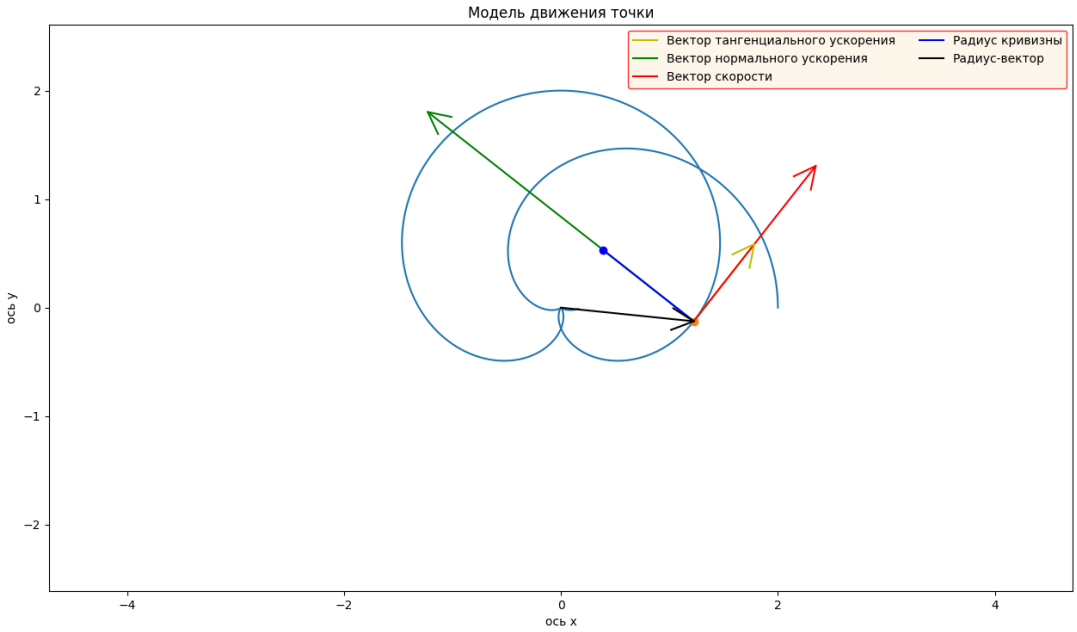
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import sympy as sp
import math

# Коэффициенты для размера стрелки
K_arrow_x = 0.1
K_arrow_y = 0.05

# Функция поворота на угол Alpha
def rot2d(X, Y, Alpha):
    RX = X * np.cos(Alpha) - Y * np.sin(Alpha)
    RY = X * np.sin(Alpha) + Y * np.cos(Alpha)
    return RX, RY


# Функция для анимации
def anima(i):
    P.set_data(X[i], Y[i])
    rhoDot.set_data(X[i] + RhoX[i], Y[i] + RhoY[i])
    VLine.set_data([X[i], X[i] + VX[i]], [Y[i], Y[i] + VY[i]])
    rhoLine.set_data([X[i], X[i] + RhoX[i]], [Y[i], Y[i] + RhoY[i]])
    WtauLine.set_data([X[i], X[i] + WtauX[i]], [Y[i], Y[i] + WtauY[i]])
    WnLine.set_data([X[i], X[i] + WnX[i]], [Y[i], Y[i] + WnY[i]])
    RLine.set_data([0, X[i]], [0, Y[i]])

    RArrowX, RArrowY = rot2d(ArrowX, ArrowY, math.atan2(VY[i], VX[i]))
    RWtauArrowX, RWtauArrowY = rot2d(WtauArrowX, WtauArrowY, math.atan2(WtauY[i],
WtauX[i]))
    RWnArrowX, RWnArrowY = rot2d(WnArrowX, WnArrowY, math.atan2(WnY[i], WnX[i]))
    RArrowRx, RArrowRy = rot2d(ArrowRx, ArrowRy, math.atan2(Y[i], X[i]))

    VArrow.set_data(RArrowX + X[i] + VX[i], RArrowY + Y[i] + VY[i])
```

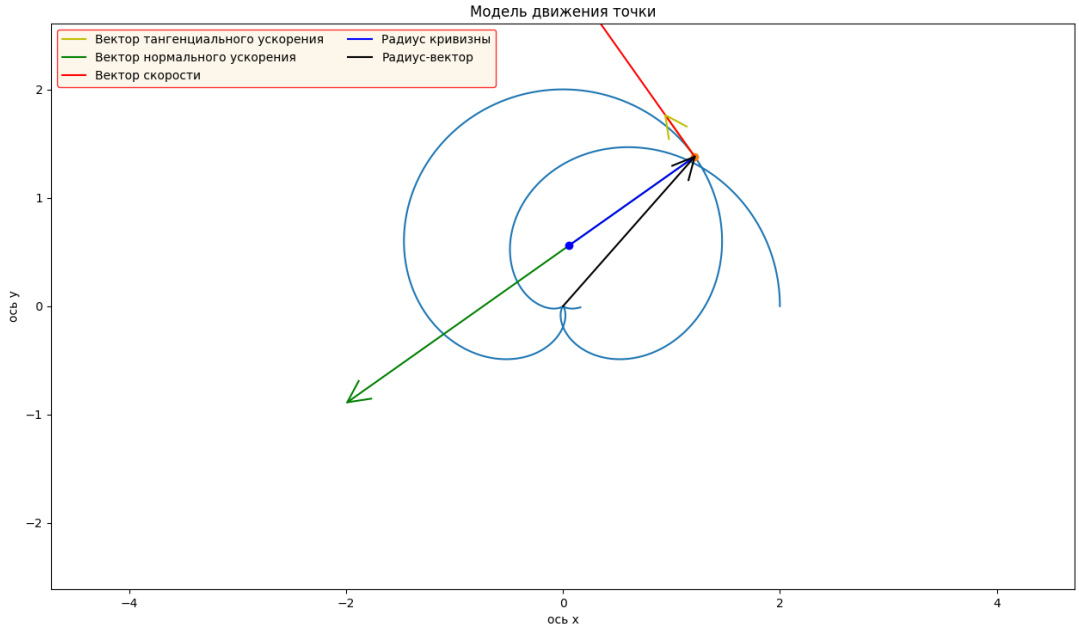
```
    WtauArrow.set_data(RWtauArrowX + X[i] + WtauX[i], RWtauArrowY + Y[i] + WtauY[i])
    WnArrow.set_data(RWnArrowX + X[i] + WnX[i], RWnArrowY + Y[i] + WnY[i])
    RArrow.set_data(RArrowRx + X[i], RArrowRy + Y[i])

    return P, VLine, VArrow, WtauLine, WnLine, rhoLine, WtauArrow, WnArrow, RLine, RArrow, rhoDot


t = sp.Symbol('t')
# переход в декартовы координаты
x = (1 + sp.cos(t)) * sp.cos(1.25 * t)
y = (1 + sp.cos(t)) * sp.sin(1.25 * t)

# Вычисление скорости
Vx = sp.diff(x, t)
Vy = sp.diff(y, t)
Vmod = sp.sqrt(Vx * Vx + Vy * Vy)

# Вычисление ускорения
Wx = sp.diff(Vx, t)
Wy = sp.diff(Vy, t)
Wmod = sp.sqrt(Wx * Wx + Wy * Wy)

# Вычисление тангенциального ускорения
cosa = (Vx*Wx + Vy*Wy)/(Vmod * Wmod)
Wtaux = Vx/Vmod*Wmod*cosa
Wtauy = Vy/Vmod*Wmod*cosa
Wtau = sp.diff(Vmod, t)

# Вычисление радиуса кривизны
rho = (Vmod * Vmod) / sp.sqrt(Wmod * Wmod - Wtau * Wtau)
rhox = -sp.diff(y, t)*(sp.diff(x, t)**2 + sp.diff(y, t)**2)/(sp.diff(x, t)*sp.diff(y, t, 2) - sp.diff(x, t, 2)*sp.diff(y, t))
rhoy = sp.diff(x, t)*(sp.diff(x, t)**2 + sp.diff(y, t)**2)/(sp.diff(x, t)*sp.diff(y, t, 2) - sp.diff(x, t, 2)*sp.diff(y, t))

# Вычисление нормального ускорения
sina = sp.sqrt(1-cosa**2)
Wnx = rhox/rho*Wmod*sina
Wny = rhoy/rho*Wmod*sina

T = np.linspace(0, 10, 1000)
x_def = sp.lambdify(t, x)
y_def = sp.lambdify(t, y)
vx_def = sp.lambdify(t, Vx)
vy_def = sp.lambdify(t, Vy)
```

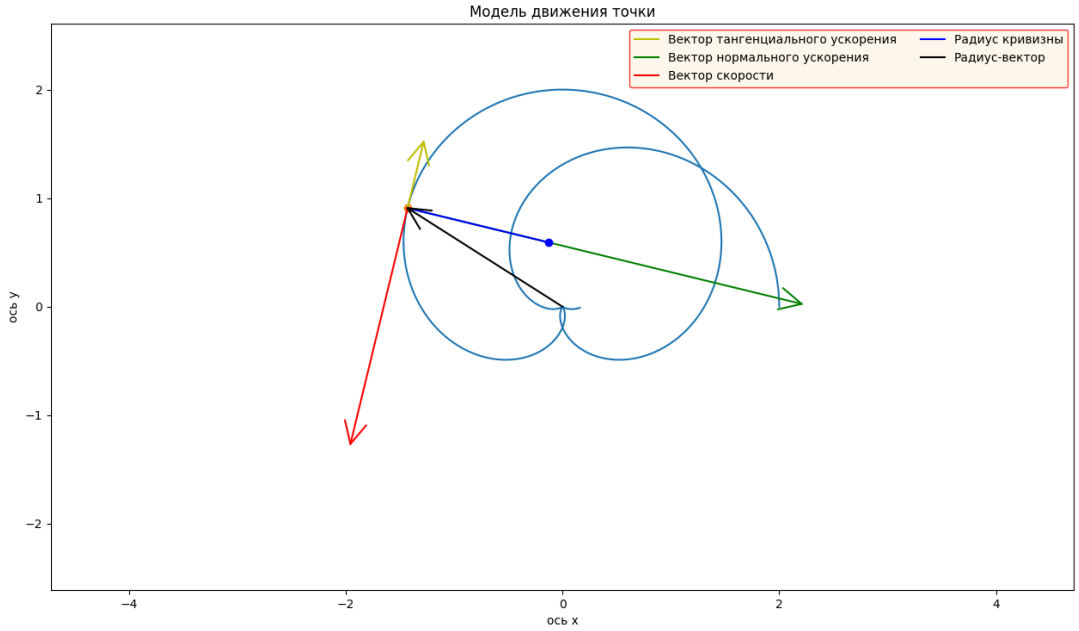
```
wtaux_def = sp.lambdify(t, Wtaux)
wtauy_def = sp.lambdify(t, Wtauy)
wnx_def = sp.lambdify(t, Wnx)
wny_def = sp.lambdify(t, Wny)
rhox_def = sp.lambdify(t, rhox)
rhoy_def = sp.lambdify(t, rhoy)
X = x_def(T)
Y = y_def(T)
VX = vx_def(T)
VY = vy_def(T)
WtauY = wtauy_def(T)
WtauX = wtaux_def(T)
WnY = wny_def(T)
WnX = wnx_def(T)
RhoY = rhoy_def(T)
RhoX = rhox_def(T)

# Создать окно
fig = plt.figure()
# ax1 - окно с графиком
ax1 = fig.add_subplot(1, 1, 1)
ax1.axis('equal')
ax1.set_title("Модель движения точки")
ax1.set_xlabel('ось x')
ax1.set_ylabel('ось y')

# Построение траектории
ax1.plot(X, Y)

# Построение точки
P, = ax1.plot(X[0], Y[0], marker='o')
# Построение векторов
WtauLine, = ax1.plot([X[0], X[0] + WtauX[0]], [Y[0], Y[0] + WtauY[0]], 'y', label='Вектор тангенциального ускорения')
WnLine, = ax1.plot([X[0], X[0] + WnX[0]], [Y[0], Y[0] + WnY[0]], 'g', label='Вектор нормального ускорения')
VLine, = ax1.plot([X[0], X[0] + VX[0]], [Y[0], Y[0] + VY[0]], 'r', label='Вектор скорости')
rhoLine, = ax1.plot([X[0], X[0] + RhoX[0]], [Y[0], Y[0] + RhoY[0]], 'b', label='Радиус кривизны')
RLine, = ax1.plot([0, X[0]], [0, Y[0]], 'black', label='Радиус-вектор')
# Построение точки для радиуса кривизны
rhoDot, = ax1.plot(X[0] + RhoX[0], Y[0] + RhoY[0], 'b', marker='o')

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

```
ArrowX = np.array([-K_arrow_x * R, 0, -K_arrow_x * R])
ArrowY = np.array([K_arrow_y * R, 0, -K_arrow_y * R])
RArrowX, RArrowY = rot2d(ArrowX, ArrowY, math.atan2(VY[0], VX[0]))
VArrow, = ax1.plot(RArrowX + X[0] + VX[0], RArrowY + Y[0] + VY[0], 'r')

WtauArrowX = np.array([-K_arrow_x * R, 0, -K_arrow_x * R])
WtauArrowY = np.array([K_arrow_y * R, 0, -K_arrow_y * R])
RWtauArrowX, RWtauArrowY = rot2d(WtauArrowX, WtauArrowY, math.atan2(WtauY[0], WtauX[0]))
WtauArrow, = ax1.plot(RWtauArrowX + X[0] + WtauX[0], RWtauArrowY + Y[0] + WtauY[0], 'y')

WnArrowX = np.array([-K_arrow_x * R, 0, -K_arrow_x * R])
WnArrowY = np.array([K_arrow_y * R, 0, -K_arrow_y * R])
RWnArrowX, RWnArrowY = rot2d(WnArrowX, WnArrowY, math.atan2(WnY[0], WnX[0]))
WnArrow, = ax1.plot(RWnArrowX + X[0] + WnX[0], RWnArrowY + Y[0] + WnY[0], 'g')

ArrowRx = np.array([-K_arrow_x * R, 0, -K_arrow_x * R])
ArrowRy = np.array([K_arrow_y * R, 0, -K_arrow_y * R])
RArrowRx, RArrowRy = rot2d(ArrowRx, ArrowRy, math.atan2(Y[0], X[0]))
RArrow, = ax1.plot(RArrowRx + X[0], RArrowRy + Y[0], 'black')

# Вывод легенды
ax1.legend(ncol=2, facecolor='oldlace', edgecolor='r')
ax1.set(xlim=[-5, 5], ylim=[-5, 5])

anim = FuncAnimation(fig, anima, frames=1000, interval=100, blit=True)
plt.show()
```

Результат работы программы:







Вывод:

В процессе выполнения работы я смогла совместить навык программирования с знаниями в области теоретической механики, а именно движения точки. Также ознакомилась с функционалом языка Python для данных задач в виде библиотек Matplotlib, Sympy, Numpy.