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 №5

Выполнил(а) студент группы М8О-207Б-22

Брюханов З. Д._____
подпись, дата

Проверил и принял

Зав. каф. 802, Бардин Б.С._____
подпись, дата

с оценкой _____

Москва, 2023

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

Закон движения

$R(t) = 2 + \sin(8t)$
$\varphi(t) = t + 0.5*\sin(4t)$

Код программы

```
import numpy as np
import sympy as sp
import math
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
# Функция для вращения точек
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
t = sp.Symbol('t')
T = np.linspace(0, 20, 1000)
r = 2 + sp.sin(8 * t)
phi = t + 0.5 * sp.sin(4 * t)
x = r * sp.cos(phi)
y = r * sp.sin(phi)
# Скорость
Vx = sp.diff(x, t)
Vy = sp.diff(y, t)
# Ускорение
Wx = sp.diff(Vx, t)
Wy = sp.diff(Vy, t)
# Тангециальное ускорение
t_x = Vx / (Vx ** 2 + Vy ** 2) ** (1 / 2)
t_y = Vy / (Vx ** 2 + Vy ** 2) ** (1 / 2)
wt_x = sp.diff((Vx ** 2 + Vy ** 2) ** (1 / 2), t) * t_x
wt_y = sp.diff((Vx ** 2 + Vy ** 2) ** (1 / 2), t) * t_y
# Центр кривизны
xc = x - Vy * (Vx ** 2 + Vy ** 2) / (Vx * Wy - Wx * Vy)
yc = y + Vx * (Vx ** 2 + Vy ** 2) / (Vx * Wy - Wx * Vy)
# Нормальное ускорение
wn_x = Wx - wt_x
wn_y = Wy - wt_y
X = np.zeros_like(T)
Y = np.zeros_like(T)
```

```
XC = np.zeros_like(T)
YC = np.zeros_like(T)
VX = np.zeros_like(T)
VY = np.zeros_like(T)
WY = np.zeros_like(T)
WX = np.zeros_like(T)
WT_X = np.zeros_like(T)
WT_Y = np.zeros_like(T)
WN_X = np.zeros_like(T)
WN_Y = np.zeros_like(T)
for i in np.arange(len(T)):
    X[i] = sp.Subs(x, t, T[i])
    Y[i] = sp.Subs(y, t, T[i])
    VY[i] = sp.Subs(Vy, t, T[i])
    VX[i] = sp.Subs(Vx, t, T[i])
    WX[i] = sp.Subs(Wx, t, T[i])
    WY[i] = sp.Subs(Wy, t, T[i])
    YC[i] = sp.Subs(yc, t, T[i])
    XC[i] = sp.Subs(xc, t, T[i])
    WT_X[i] = sp.Subs(wt_x, t, T[i])
    WT_Y[i] = sp.Subs(wt_y, t, T[i])
    WN_Y[i] = sp.Subs(wn_y, t, T[i])
    WN_X[i] = sp.Subs(wn_x, t, T[i])
fig = plt.figure()
ax1 = fig.add_subplot(1, 1, 1)
ax1.axis('equal')
ax1.set(xlim=[-6, 6], ylim=[-6, 6])
ax1.plot(X, Y, linestyle="--", color="gray")
point, = ax1.plot(X[0], Y[0], marker="o", color="#00B873")
# Кривизна
evoluta, = ax1.plot(XC[0], YC[0], marker=".", color="#210881")
curve_line, = ax1.plot([XC[0], X[0]], [YC[0], Y[0]], linestyle="--", color="#7253E2",
label="$радиус\\ кривизны$")
Rad = math.sqrt((XC[0] - X[0]) ** 2 + (YC[0] - Y[0]) ** 2)
curvature_circle = plt.Circle((XC[0], YC[0]), Rad, alpha=0.3, edgecolor="#210881",
facecolor="#7253E2")
ax1.add_patch(curvature_circle)
# Вектор скорости
VLine, = ax1.plot([X[0], X[0] + VX[0]], [Y[0], Y[0] + VY[0]], 'r', label="$v$")
ArrowX = np.array([-0.2, 0, -0.2])
ArrowY = np.array([0.1, 0, -0.1])
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')
# Радиус-вектор
```

```
R_Vline, = ax1.plot([0, X[0]], [0, Y[0]], color="orange", label="$\\rho\\ (радиус-вектор)
$")
R_RArrowX, R_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(Y[0], X[0]))
R_VArrow, = ax1.plot(R_RArrowX + X[0], R_RArrowY + Y[0], 'r', color="orange")

# Вектор ускорения
W_Vline, = ax1.plot([X[0], X[0] + WX[0]], [Y[0], Y[0] + WY[0]], color="#00D939",
label="$w$")
W_RArrowX, W_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WY[0], WX[0]))
W_VArrow, = ax1.plot(W_RArrowX + X[0] + WX[0], W_RArrowY + Y[0] + WY[0],
color="#00D939")
# Вектор тангенциального ускорения
WT_Vline, = ax1.plot([X[0], X[0] + WT_X[0]], [Y[0], Y[0] + WT_Y[0]],
color="purple",
                     label="$тангенциальное \\ уравнение$")
WT_RArrowX, WT_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WT_Y[0],
WT_X[0]))
WT_VArrow, = ax1.plot(WT_RArrowX + X[0] + WT_X[0], WT_RArrowY + Y[0] +
WT_Y[0], color="purple")
# Вектор нормального ускорения
WN_Vline, = ax1.plot([X[0], X[0] + WN_X[0]], [Y[0], Y[0] + WN_Y[0]],
color="green", label="$нормальное \\ ускорение$",
                     alpha=0.3)
WN_RArrowX, WN_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WN_Y[0],
WN_X[0]))
WN_VArrow, = ax1.plot(WN_RArrowX + X[0] + WN_X[0], WN_RArrowY + Y[0] +
WN_Y[0], color="green", alpha=0.3)
def anima(i):
    # Точка
    point.set_data(X[i], Y[i])
    # Кривизна
    curve_line.set_data([XC[i], X[i]], [YC[i], Y[i]])
    evoluta.set_data(XC[i], YC[i])
    Radius = ax1.plot((VX[i] ** 2 + VY[i] ** 2) / (
        math.sqrt(WX[i] ** 2 + WY[i] ** 2 - sp.diff(math.sqrt(VX[i] ** 2 + VY[i] ** 2), t)
** 2)))
    global curvature_circle
    curvature_circle.remove()
    Rad = math.sqrt((XC[i] - X[i]) ** 2 + (YC[i] - Y[i]) ** 2)
    curvature_circle = plt.Circle((XC[i], YC[i]), Rad, alpha=0.3, edgecolor="#210881",
facecolor='#7253E2')
    ax1.add_patch(curvature_circle)
    # Вектор скорости
    VLine.set_data([X[i], X[i] + VX[i]], [Y[i], Y[i] + VY[i]])
    RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[i], VX[i]))
```

```
    VArrow.set_data(RArrowX + X[i] + VX[i], RArrowY + Y[i] + VY[i])
    # Радиус-вектор
    R_Vline.set_data([0, X[i]], [0, Y[i]])
    R_RArrowX, R_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(Y[i], X[i]))
    R_VArrow.set_data(R_RArrowX + X[i], R_RArrowY + Y[i])
    # Вектор ускорения
    W_Vline.set_data([X[i], X[i] + WX[i]], [Y[i], Y[i] + WY[i]])
    W_RArrowX, W_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WY[i], WX[i]))
    W_VArrow.set_data(R_RArrowX + X[i] + WX[i], R_RArrowY + Y[i] + WY[i])
    # Вектор тангециального ускорения
    WT_Vline.set_data([X[i], X[i] + WT_X[i]], [Y[i], Y[i] + WT_Y[i]])
    WT_RArrowX, WT_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WT_Y[i],
WT_X[i]))
    WT_VArrow.set_data(WT_RArrowX + X[i] + WT_X[i], WT_RArrowY + Y[i] +
WT_Y[i])
    # Вектор нормального ускорения
    WN_Vline.set_data([X[i], X[i] + WN_X[i]], [Y[i], Y[i] + WN_Y[i]])
    WN_RArrowX, WN_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WN_Y[i],
WN_X[i]))
    WN_VArrow.set_data(WN_RArrowX + X[i] + WN_X[i], WN_RArrowY + Y[i] +
WN_Y[i])
    return point, evoluta, curve_line, curvature_circle, VLine, VArrow, R_VArrow,
R_Vline, W_Vline, W_VArrow, WT_Vline, WT_VArrow, WN_Vline, WN_VArrow
anim = FuncAnimation(fig, anima, frames=1000, interval=50, blit=True)
plt.legend()
plt.show()
```

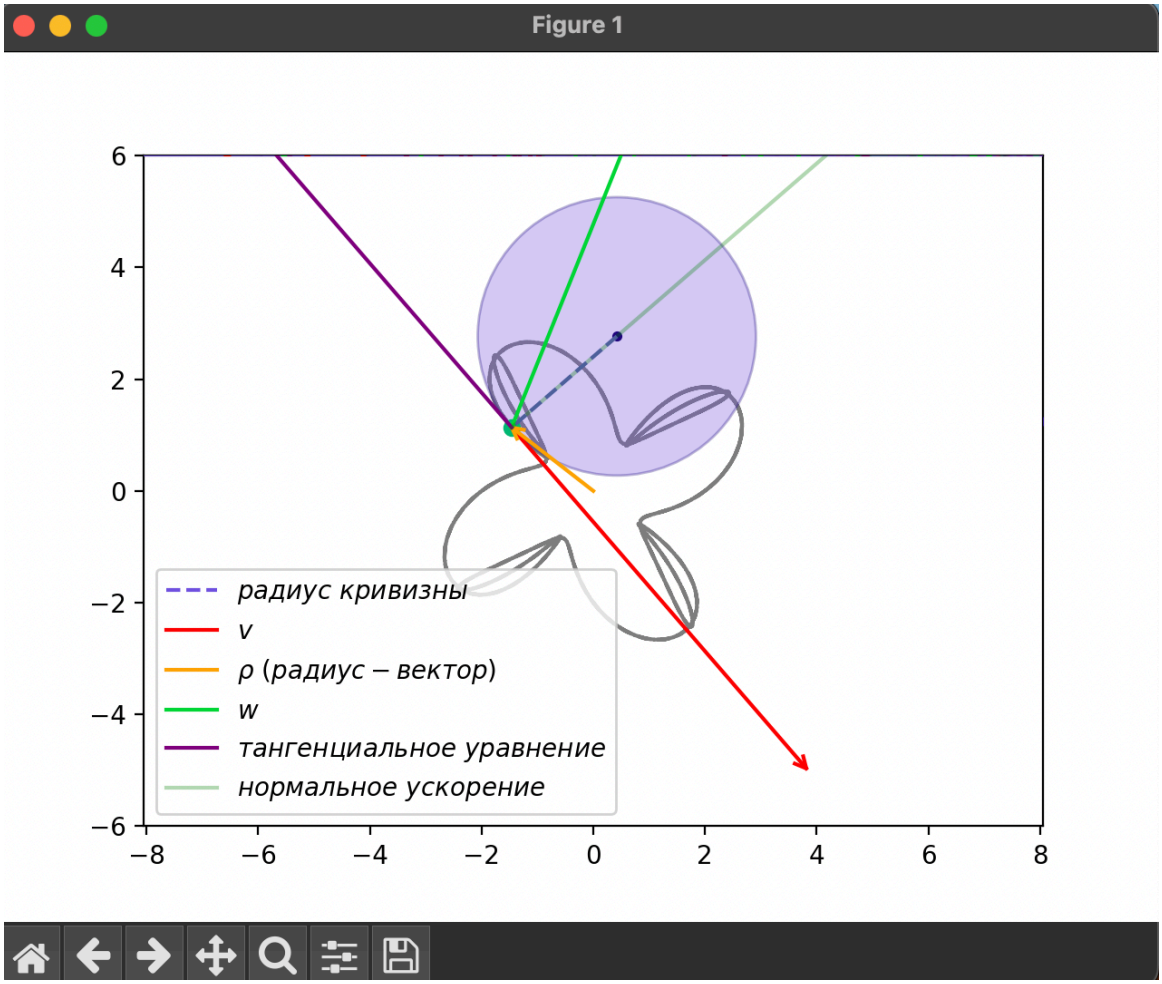
Описание

Эта программа создает анимацию движения точки на плоскости, описанного параметрическим уравнением. Параметры движения, такие как координаты точки, скорость, ускорение и другие, вычисляются с использованием библиотеки SymPy для символьных вычислений и библиотеку Matplotlib для построения графиков и анимации.

Программа моделирует движение точки, заданное уравнением в полярных координатах (r(t), phi(t)). Затем вычисляются скорость, ускорение и другие характеристики движения. Анимация показывает, как эти характеристики меняются во времени.

Результат работы программы

Вывод



В ходе выполнения лабораторной работы мною были освоены навыки символьных вычислений с использованием библиотеки SymPy, численного интегрирования для моделирования движения объекта, создания анимации с использованием Matplotlib, а также понимание кинематических концепций, таких как траектория, скорость, ускорение, тангенциальное и нормальное ускорения, и кривизна траектории.