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 №4**

Выполнил(а) студент группы М8О-212Б-22

Баталин Дмитрий Андреевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

подпись, дата

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

Москва, 2023

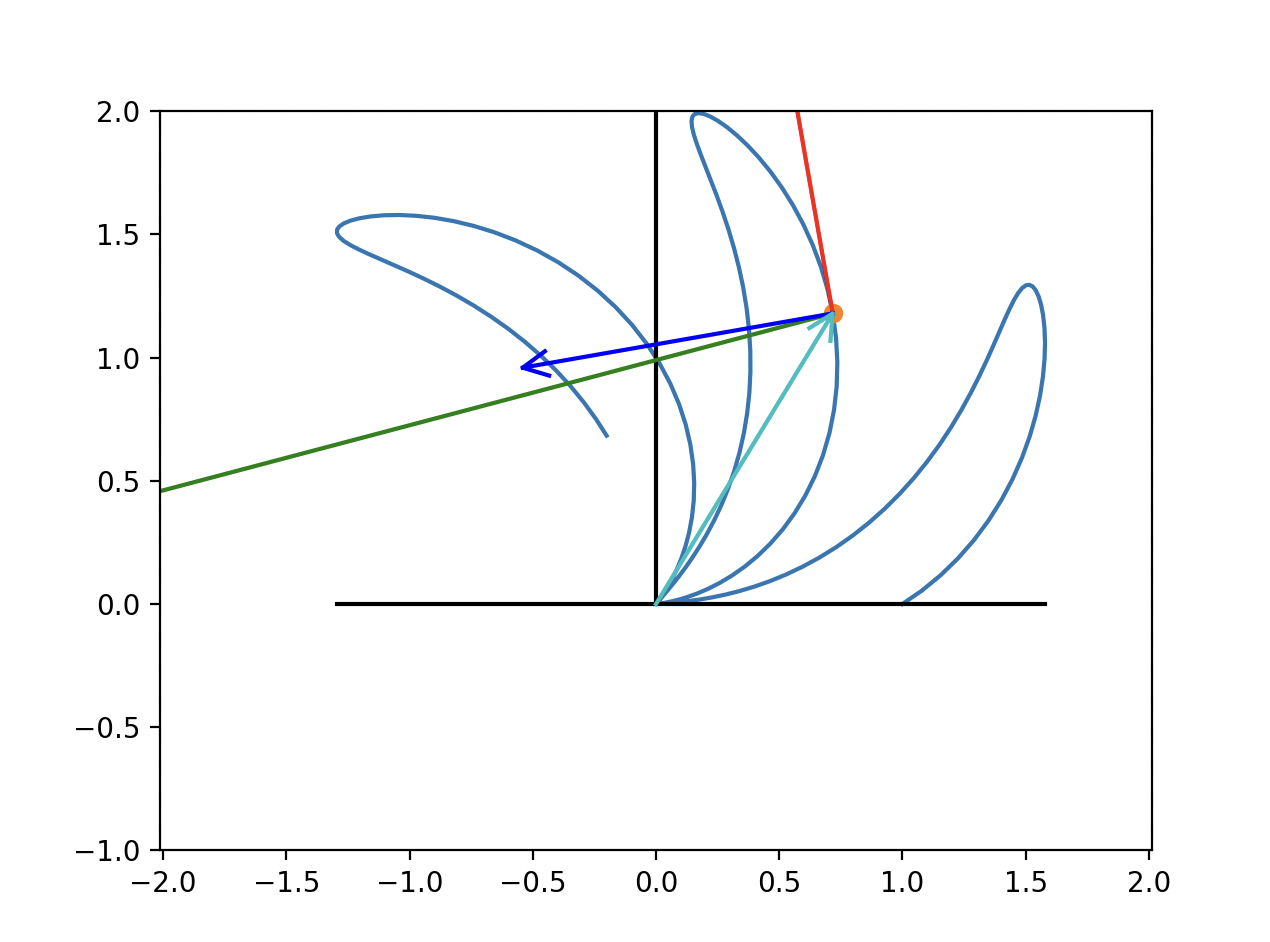
**Задание:** построить заданную траекторию, запустить анимацию движения точки, построить стрелки радиус-вектора, вектора скорости, вектора ускорения и радиуса кривизны.

**Условия задачи 4 варианта:**

r(t) = 1 + sin(8t)

φ(t) = t + 0.5sin(8t)

**Рисунок получившейся физической модели:**



Красным цветом изображен вектор скорости  
Синим цветом изображен радиус кривизны  
Зеленым цветом изображен вектор ускорения  
Голубым цветом изображен радиус-вектор

**Код программы**

**import** **numpy** **as** **np**

**import** **sympy** **as** **sp**

**import** **math**

**import** **matplotlib.pyplot** **as** **plt**

**from** **matplotlib.animation** **import** FuncAnimation

**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

frames = **200**

t = sp.Symbol('t')

r = **1** + sp.sin(**8**\*t)

phi = t + **0.5** \* sp.sin(**8**\*t)

x = r \* sp.cos(phi)

y = r \* sp.sin(phi)

Vx = sp.diff(x, t)

Vy = sp.diff(y, t)

v = (Vx \*\* **2** + Vy \*\* **2**) \*\* **0.5**

Wx = sp.diff(Vx, t)

Wy = sp.diff(Vy, t)

w = (Wx \*\* **2** + Wy \*\* **2**) \*\* **0.5**

Wtan = sp.diff(v, t) # получили модуль тангенциального ускорения в каждый момент времени

Wnor = (w \*\* **2** - Wtan \*\* **2**) \*\* **0.5** # нашли нормальное ускорение как разность полного и тангенциального в квадратах

# ищем модуль радиуса кривизны

curvatureRadius = v\*v/Wnor

# находим координаты вектора тангенциального ускорения:

# нормируем вектор скорости и умножаем на величину тангенциального

WTanx = Vx / v \* Wtan

WTany = Vy / v \* Wtan

# N - единичный вектор, сонаправленный с нормальным ускорением

# Вычитая из координат полного ускорения координаты тангенциального ускорения,

# получаем координаты нормального ускорения

WNorX = Wx - WTanx

WNorY = Wy - WTany

WNor = (WNorX \*\* **2** + WNorY \*\* **2**) \*\* **0.5**

Nx = WNorX / WNor

Ny = WNorY / WNor

curvatureRadiusx = Nx \* curvatureRadius

curvatureRadiusy = Ny \* curvatureRadius

Time = np.linspace(**0**, **2**, frames) # 10 секунд времени разделенные на frames частей

X\_dot = np.zeros\_like(Time)

Y\_dot = np.zeros\_like(Time)

VX = np.zeros\_like(Time)

VY = np.zeros\_like(Time)

AX = np.zeros\_like(Time)

AY = np.zeros\_like(Time)

RadiusVectorX = np.zeros\_like(Time)

RadiusVectorY = np.zeros\_like(Time)

CurvatureRadiusX = np.zeros\_like(Time)

CurvatureRadiusY = np.zeros\_like(Time)

# считаем все значения на нашем промежутке времени

**for** i **in** np.arange(len(Time)):

# точка

X\_dot[i] = sp.Subs(x, t, Time[i]) # в функицию х посдтавляет вместо t значение T[i]

Y\_dot[i] = sp.Subs(y, t, Time[i])

# скорость

VX[i] = sp.Subs(Vx, t, Time[i])

VY[i] = sp.Subs(Vy, t, Time[i])

# ускорение

AX[i] = sp.Subs(Wx, t, Time[i])

AY[i] = sp.Subs(Wy, t, Time[i])

# радиус вектор

RadiusVectorX[i] = sp.Subs(x, t, Time[i])

RadiusVectorY[i] = sp.Subs(y, t, Time[i])

# радиус кривизны

CurvatureRadiusX[i] = sp.Subs(curvatureRadiusx, t, Time[i])

CurvatureRadiusY[i] = sp.Subs(curvatureRadiusy, t, Time[i])

# задаем условия графика

fig = plt.figure()

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

ax1.axis('equal')

ax1.set(xlim=[int(X\_dot.min()) - **1**, int(X\_dot.max()) + **1**], ylim=[int(Y\_dot.min()) - **1**, int(Y\_dot.max()) + **1**])

# рисуем сразу всю траекторию движения точки

ax1.plot(X\_dot, Y\_dot)

# рисуем оси координат

ax1.plot([min(**0**, X\_dot.min()), max(**0**, X\_dot.max())], [**0**, **0**], 'black')

ax1.plot([**0**, **0**], [min(**0**, Y\_dot.min()), max(**0**, Y\_dot.max())], 'black')

# рисуем точку в начальный момент времени

P, = ax1.plot(X\_dot[**0**], Y\_dot[**0**], marker='o')

# вектор скорости

VLine, = ax1.plot([X\_dot[**0**], X\_dot[**0**]+VX[**0**]], [Y\_dot[**0**], Y\_dot[**0**]+VY[**0**]], 'r')

# вектор ускорения

ALine, = ax1.plot([X\_dot[**0**], X\_dot[**0**]+AX[**0**]], [Y\_dot[**0**], Y\_dot[**0**]+AY[**0**]], 'g')

# радиус вектор

RadiusVector, = ax1.plot([**0**, X\_dot[**0**]], [**0**, Y\_dot[**0**]], 'c')

# радиус кривизны

CurvatureRadiusVector, = ax1.plot([X\_dot[**0**], X\_dot[**0**] + CurvatureRadiusX[**0**]], [Y\_dot[**0**], Y\_dot[**0**] + CurvatureRadiusY[**0**]], 'b')

# задаем красивую стрелочку

arrowMult = **0.5**

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

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

# стрелочка для вектора скорости

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

VArrow, = ax1.plot(RArrowX+X\_dot[**0**]+VX[**0**], RArrowY+Y\_dot[**0**]+VY[**0**], 'r')

# стрелочка для вектора ускорения

AArrowX, AArrowY = Rot2D(ArrowX, ArrowY, math.atan2(AY[**0**], AX[**0**]))

AArrow, = ax1.plot(AArrowX+X\_dot[**0**]+AX[**0**], AArrowY+Y\_dot[**0**]+AY[**0**], 'g')

# стрелочка для радиус вектора

RadiusVectorArrowX, RadiusVectorArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[**0**], VX[**0**]))

RadiusVectorArrow, = ax1.plot(RadiusVectorArrowX+X\_dot[**0**], RadiusVectorArrowY+Y\_dot[**0**], 'c')

# стрелочка для радиуса кривизны

CurvatureRadiusVectorArrowX, CurvatureRadiusVectorArrowY = Rot2D(

ArrowX, ArrowY, math.atan2(CurvatureRadiusY[**0**], CurvatureRadiusX[**0**])

)

CurvatureRadiusVectorArrow, = ax1.plot(

CurvatureRadiusVectorArrowX+CurvatureRadiusX[**0**], CurvatureRadiusVectorArrowY+CurvatureRadiusY[**0**], 'b'

)

**def** **anima**(i):

P.set\_data(X\_dot[i], Y\_dot[i]) # изменяем положение точки, меняем координаты на соответствующие времени

# вектор скорости

VLine.set\_data([X\_dot[i], X\_dot[i]+VX[i]], [Y\_dot[i], Y\_dot[i]+VY[i]])

# вектор ускорения

ALine.set\_data([X\_dot[i], X\_dot[i]+AX[i]], [Y\_dot[i], Y\_dot[i]+AY[i]])

# радиус вектор

RadiusVector.set\_data([**0**, X\_dot[i]], [**0**, Y\_dot[i]])

# радиус кривизны

CurvatureRadiusVector.set\_data([X\_dot[i], X\_dot[i] + CurvatureRadiusX[i]], [Y\_dot[i], Y\_dot[i] + CurvatureRadiusY[i]])

# стрелочка для вектора скорости

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

VArrow.set\_data(RArrowX+X\_dot[i]+VX[i], RArrowY+Y\_dot[i]+VY[i])

# стрелочка для вектора ускорения

AArrowX, AArrowY = Rot2D(ArrowX, ArrowY, math.atan2(AY[i], AX[i]))

AArrow.set\_data(AArrowX+X\_dot[i]+AX[i], AArrowY+Y\_dot[i]+AY[i])

# стрелочка для радиус вектора

RadiusVectorArrowX, RadiusVectorArrowY = Rot2D(ArrowX, ArrowY, math.atan2(Y\_dot[i], X\_dot[i]))

RadiusVectorArrow.set\_data(RadiusVectorArrowX+X\_dot[i], RadiusVectorArrowY+Y\_dot[i])

# стрелочка для радиуса кривизны

CurvatureRadiusVectorArrowX, CurvatureRadiusVectorArrowY = Rot2D(

ArrowX, ArrowY, math.atan2(CurvatureRadiusY[i], CurvatureRadiusX[i])

)

CurvatureRadiusVectorArrow.set\_data(

CurvatureRadiusVectorArrowX+X\_dot[i] + CurvatureRadiusX[i], CurvatureRadiusVectorArrowY+Y\_dot[i] + CurvatureRadiusY[i]

)

anim = FuncAnimation(fig, anima, frames=frames, interval=**50**, repeat=False)

plt.show()

**Пояснения**

В процессе выполнения работы мне нужно было изобразить несколько векторов. Вектор скорости достаточно просто рисовался, благодаря имеющейся функции координат, требовалось всего лишь взять производную по каждой из координат. Таким же образом рисовался вектор ускорения – вторая производная по времени от координат. Радиус-вектор – отрезок, соединяющий начало координат и движущуюся точку. А вот построить радиус кривизны уже не было такой простой задачей.

Чтобы построить радиус кривизны, нам требовалось узнать направление и модуль. Модуль можно узнать по формуле:

Где – нормальное ускорение. можно узнать из полного и тангенциального. Тангенциальное ускорение получается как производная модуля скорости по времени. Таким образом, получаем Следующим шагом требуется найти направление. Радиус кривизны сонаправлен с нормальным ускорением, следовательно, требуется лишь найти вектор нормального ускорения и нормировать его. Получить координаты нормального ускорения можно через полное и тангенциальное, и поделив координаты на длины получим единичный вектор, сонаправленный с радиусом кривизны. Умножая длину радиуса кривизны на единичный вектор получаем вектор радиуса кривизны.

**Вывод**

Я успешно выполнил лабораторную работу по теоретической механике. С помощью языка программирования Python и библиотек matplotlib, numpy и sympy я построил заданную траекторию, а также запустил анимацию движения точки по этой траектории. Для каждого момента времени я изобразил векторы скорости, ускорения, радиус-вектора, вектора радиуса кривизны.

Эта лабораторная работа позволила мне лучше разобраться в теме движения точки, понять как связаны между собой разные характеристики движения точки – скорость и ускорения.