Московский авиационный институт
(национальный исследовательский университет)
Институт № 8 «Информационные технологии и прикладная математика»

Лабораторная работа №1

по курсу «Теоретическая механика и основы компьютерного моделирования»

Анимация точки

Выполнил студент группы М8О-210Б-21

Минеева Светлана Алексеевна

Преподаватель: Бардин Борис Сабирович

Оценка:

Дата: 16.09.2022

Москва, 2022

Вариант № 14

Задание: Построить заданную траекторию и анимацию движения точки, а также отобразить векторы скорости, ускорения, радиус-вектора и радиуса кривизны.

Закон движения точки:

r(t) = 2 + sin(12t); phi(t) = t + 0.2sin(12t)

Текст программы:

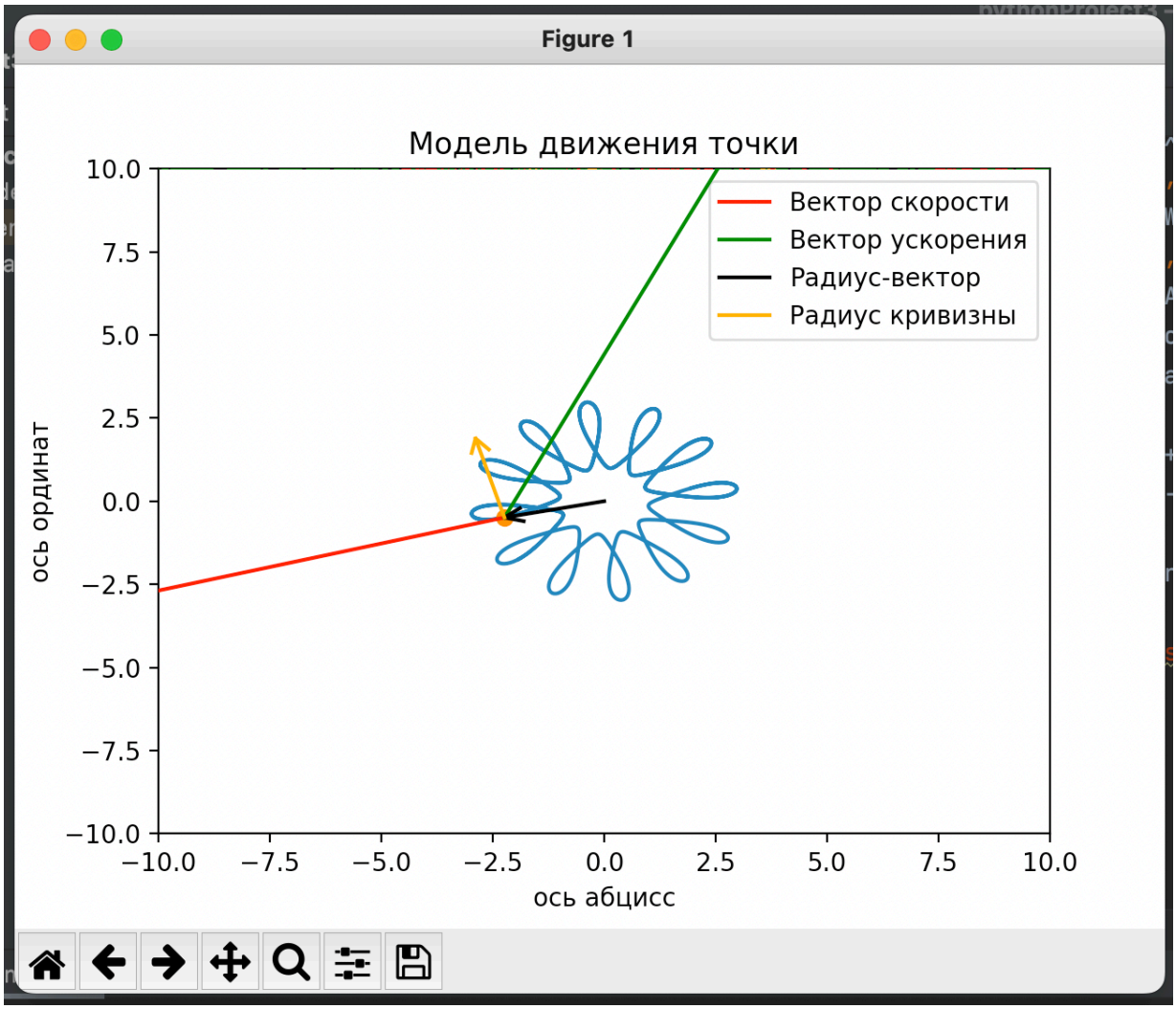
```
import numpy as np
import sympy as sp
import math
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

def Rot2D(X, Y, Alpha): #Функция поворота на угол альфа
    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')

#Переход из полярных координат в декартовы
x = (2 + sp.sin(12*t)) * sp.cos(t + 0.2 * sp.cos(12*t))
y = (2 + sp.sin(12*t)) * sp.sin(t + 0.2 * sp.cos(12*t))

Vx = sp.diff(x, t) #Вычисление скорости по x
Vy = sp.diff(y, t) #Вычисление скорости по y
V = sp.sqrt(Vx * Vx + Vy * Vy) #Вычисление общей скорости
Wx = sp.diff(Vx, t) #Вычисление ускорения по x
Wy = sp.diff(Vy, t) #Вычисление ускорения по y
W = sp.sqrt(Wx * Wx + Wy * Wy) #Вычисление общего ускорения
Wtau = sp.diff(V, t) #Вычисление тангенсального ускорения
Wn = sp.sqrt(W * W - Wtau * Wtau) #Вычисление нормального ускорения
Rk = (V * V) / Wn #Вычисление радиуса кривизны
XX = [0 for i in range(1000)]
YY = [0 for i in range(1000)]

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

#Заполнение массива нулями в соответствие с массивом T
X = np.zeros_like(T)
Y = np.zeros_like(T)
VX = np.zeros_like(T)
VY = np.zeros_like(T)
WX = np.zeros_like(T)
WY = np.zeros_like(T)
RK = np.zeros_like(T)
```

```
for i in np.arange(len(T)): #Равномерное распределение значений внутри
интервала
    X[i] = sp.Subs(x, t, T[i])
    Y[i] = sp.Subs(y, t, T[i])
    VX[i] = sp.Subs(Vx, t, T[i])
    VY[i] = sp.Subs(Vy, t, T[i])
    WX[i] = sp.Subs(Wx, t, T[i])
    WY[i] = sp.Subs(Wy, t, T[i])
    RK[i] = sp.Subs(Rk, t, T[i])

#Создание окна
fig = plt.figure()
ax1 = fig.add_subplot(1, 1, 1)
ax1.set_title("Модель движения точки")
ax1.set_xlabel('ось абцисс')
ax1.set_ylabel('ось ординат')
ax1.set_xlim([-10, 10]) #пределы оси абцисс
ax1.set_ylim([-10, 10]) #пределы оси ординат

ax1.plot(X, Y) #Построение траектории

R = math.sqrt(X[0] * X[0] + Y[0] * Y[0])

#Построение точки, векторов скорости, ускорения, радиус-вектора и радиуса
кривизны соотвественно
P, = ax1.plot(X[0], Y[0], marker = 'o')
VLine, = ax1.plot([X[0], X[0] + VX[0]], [Y[0], Y[0] + VY[0]], 'red', label =
'Вектор скорости')
WLine, = ax1.plot([X[0], X[0] + WX[0]], [Y[0], Y[0] + WY[0]], 'green', label
= 'Вектор ускорения')
RLine, = ax1.plot([0, X[0]], [0, Y[0]], 'black', label = 'Радиус-вектор')
RKLine, = ax1.plot([X[0], X[0] + (Y[0] + VY[0]) * RK[0]/sp.sqrt((Y[0] +
VY[0])**2 + (X[0] + VX[0])**2)],
                   [Y[0], Y[0] - (X[0] + VX[0]) * RK[0]/sp.sqrt((Y[0] +
VY[0])**2 + (X[0]+VX[0])**2)],
                   'orange', label = 'Радиус кривизны')

ax1.legend() #Вывод легенды на окно

#Делаем созданные точку и векторы невидимыми в окне
P.set_visible(False)
VLine.set_visible(False)
WLine.set_visible(False)
RLine.set_visible(False)
RKLine.set_visible(False)

#Создание стрелки на конце вектора скорости
ArrowX = np.array([-0.2*R, 0, -0.2*R])
ArrowY = np.array([0.1*R, 0, -0.1*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], 'red')
VArrow.set_visible(False) #Делаем стрелку невидимой в окне
```

```
#Создание стрелки на конце вектора ускорения
WArrowX = np.array([-0.2*R, 0, -0.2*R])
WArrowY = np.array([0.1*R, 0, -0.1*R])
RWArrowX, RWArrowY = Rot2D(WArrowX, WArrowY, math.atan2(WY[0], WX[0]))
WArrow, = ax1.plot(RWArrowX + X[0] + WX[0], RWArrowY + Y[0] + WY[0], 'green')
WArrow.set_visible(False) #Делаем стрелку невидимой в окне

#Создание стрелки на конце радиус-вектора
ArrowRX = np.array([-0.2*R, 0, -0.2*R])
ArrowRY = np.array([0.1*R, 0, -0.1*R])
RArrowRX, RArrowRY = Rot2D(ArrowRX, ArrowRY, math.atan2(Y[0], X[0]))
ArrowR, = ax1.plot(RArrowRX + X[0], RArrowRY + Y[0], 'black')
ArrowR.set_visible(False) #Делаем стрелку невидимой в окне

#Создание стрелки на конце радиуса кривизны
RKArrowX = np.array([-0.2*R, 0, -0.2*R])
RKArrowY = np.array([0.1*R, 0, -0.1*R])
RRKArrowX, RRKArrowY = Rot2D(RKArrowX, RKArrowY,
                             math.atan2(-(RK[0] * (X[0] + VX[0]) /
sp.sqrt((X[0] + VX[0])**2 + (Y[0] + VY[0])**2)),
                                        RK[0] * (Y[0] + VY[0]) /
sp.sqrt((X[0] + VX[0])**2 + (Y[0] + VY[0])**2)))
RKArrow, = ax1.plot(RRKArrowX + X[0] + RK[0] * (Y[0] + VY[0]) / sp.sqrt((X[0]
+ VX[0])**2 + (Y[0] + VY[0])**2),
                    RRKArrowY + Y[0] - (RK[0] * (X[0] + VX[0]) /
sp.sqrt((X[0] + VX[0])**2 + (Y[0]+VY[0])**2)), 'orange')
RKArrow.set_visible(False) #Делаем стрелку невидимой в окне

def anima(i): #Функция анимации
    P.set_visible(True) #Делаем объекты видимыми в окне
    VLine.set_visible(True)
    RLine.set_visible(True)
    RKLine.set_visible(True)
    VArrow.set_visible(True)
    WArrow.set_visible(True)
    WLine.set_visible(True)
    ArrowR.set_visible(True)
    RKArrow.set_visible(True)

    P.set_data(X[i], Y[i])
    VLine.set_data([X[i], X[i] + VX[i]], [Y[i], Y[i] + VY[i]])
    WLine.set_data([X[i], X[i] + WX[i]], [Y[i], Y[i] + WY[i]])
    RLine.set_data([XX[i], X[i]], [YY[i], Y[i]])
    RKLine.set_data([X[i], X[i] + (Y[i] + VY[i]) * RK[i] / sp.sqrt((Y[i] +
VY[i])**2 + (X[i] + VX[i])**2)],
                    [Y[i], Y[i] - (X[i] + VX[i]) * RK[i] / sp.sqrt((Y[i] +
VY[i])**2 + (X[i] + VX[i])**2)])

    RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[i], VX[i]))
    VArrow.set_data(RArrowX + X[i] + VX[i], RArrowY + Y[i] + VY[i])
    RWArrowX, RWArrowY = Rot2D(WArrowX, WArrowY, math.atan2(WY[i], WX[i]))
    WArrow.set_data(RWArrowX + X[i] + WX[i], RWArrowY + Y[i] + WY[i])
    RArrowRX, RArrowRY = Rot2D(ArrowRX, ArrowRY, math.atan2(Y[i], X[i]))
```

```
    ArrowR.set_data(RArrowRX + X[i], RArrowRY + Y[i])
    RRKArrowX, RRKArrowY = Rot2D(RKArrowX, RKArrowY,
                                 math.atan2(-(RK[i] * (X[i] + VX[i]) /
sp.sqrt((X[i] + VX[i])**2 + (Y[i] + VY[i])**2)),
                                            RK[i] * (Y[i] + VY[i]) /
sp.sqrt((X[i] + VX[i])**2 + (Y[i] + VY[i])**2)))
    RKArrow.set_data(RRKArrowX + X[i] + RK[i] * (Y[i] + VY[i]) /
sp.sqrt((X[i] + VX[i]) ** 2 + (Y[i] + VY[i]) ** 2),
                     RRKArrowY + Y[i] - (RK[i] * (X[i] + VX[i]) /
sp.sqrt((X[i] + VX[i]) ** 2 + (Y[i] + VY[i]) ** 2)))

    return P, VLine, VArrow, WLine, WArrow, RLine, ArrowR, RKLine, RKArrow

anim = FuncAnimation(fig, anima, frames = 1000, interval = 20, blit = True)
plt.show()
```

Результат работы программы:



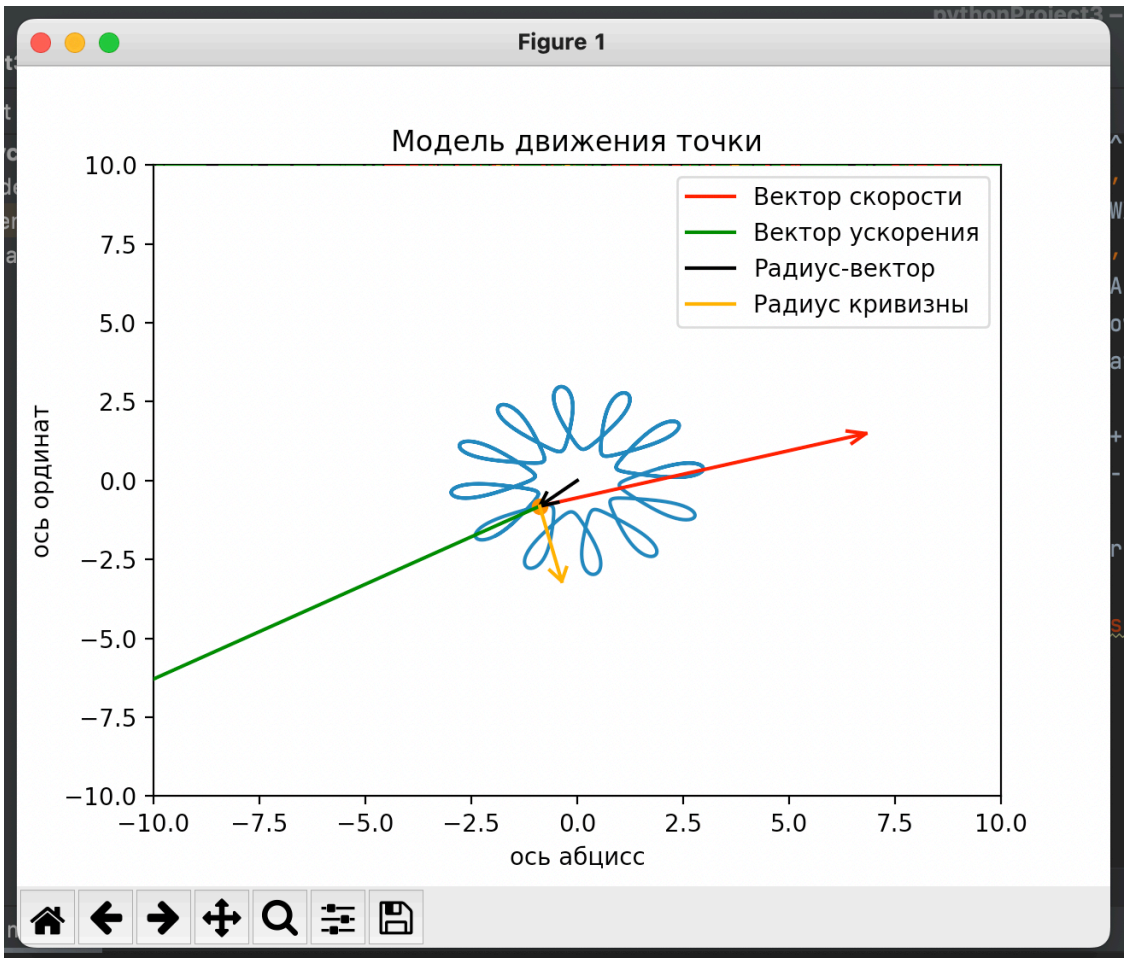


Figure 1
Модель движения точки
Вектор скорости
Вектор ускорения
Радиус-вектор
Радиус кривизны
ось ординат
ось абцисс
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

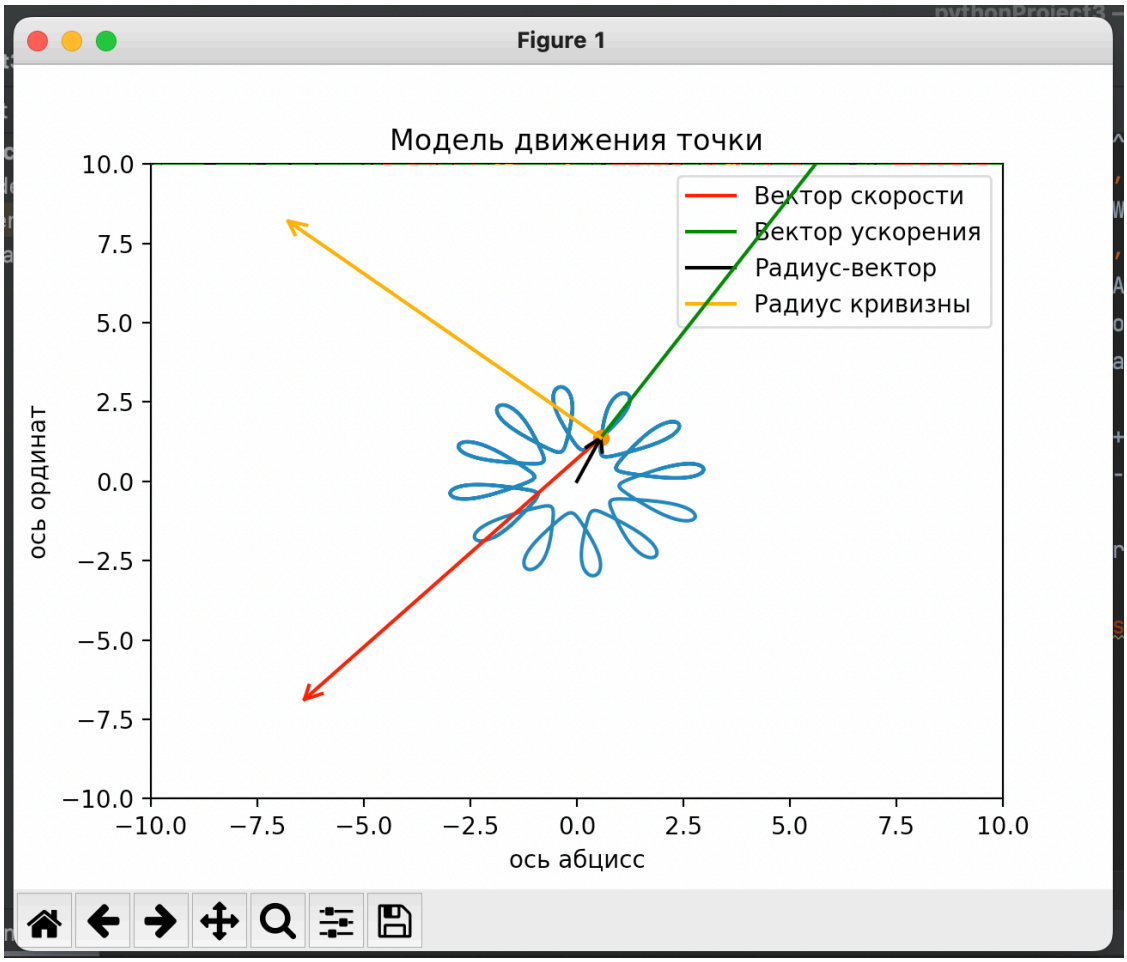
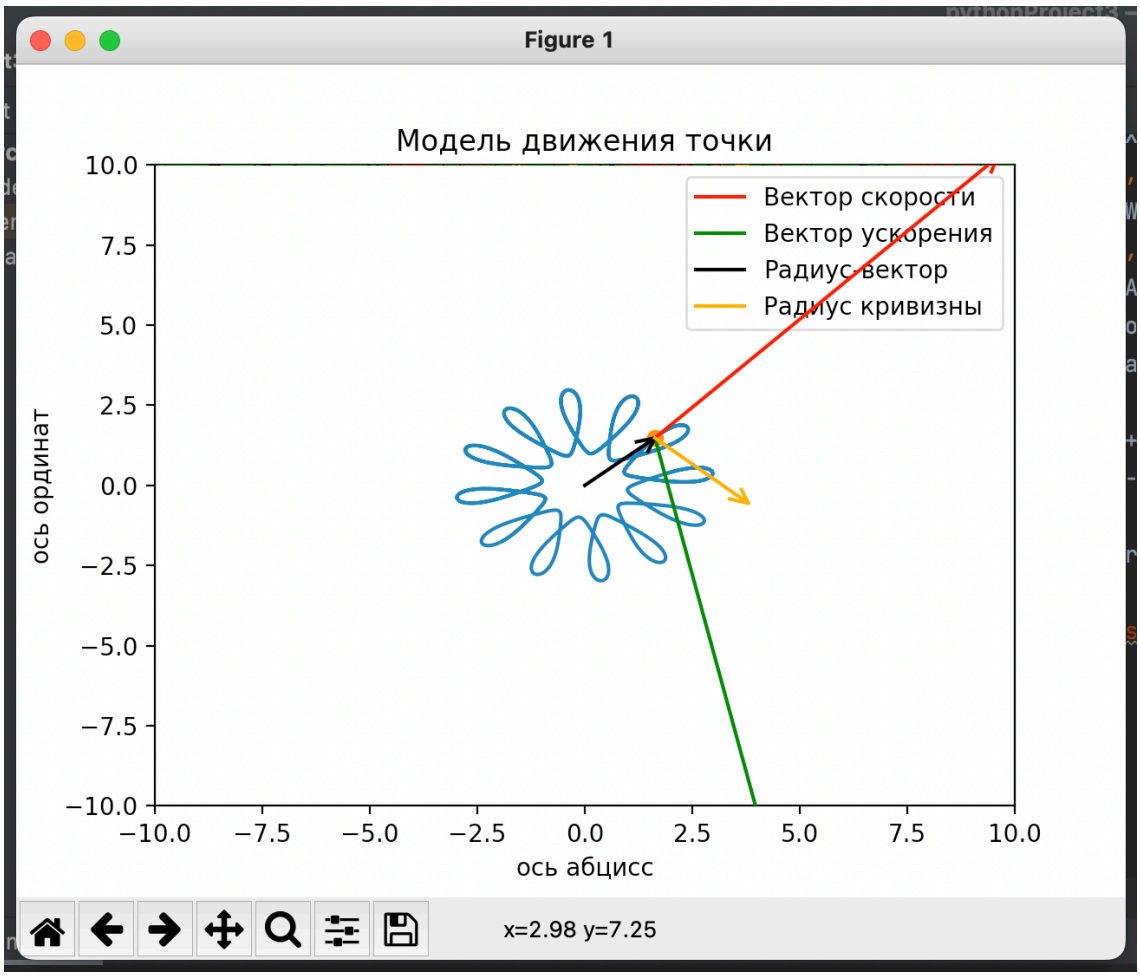


Figure 1
Модель движения точки
Вектор скорости
Вектор ускорения
Радиус-вектор
Радиус кривизны
ось ординат
ось абцисс
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0



Вывод: Я построила заданную траекторию и анимацию движения точки, отобразила векторы скорости, ускорения, радиус-вектора и радиуса кривизны. В ходе данной работы, я приобрела практические знания в работе с языком программирования Python и его библиотекой Matplotlib.