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-207Б-20

Алапанова Эльза Халилевна

Преподаватель: Чекина Евгения Александровна

Оценка:

Дата: 05.10.2021

Москва, 2021

**Вариант № 4**

**Задание:**

Построить заданную траекторию и анимацию движения точки, а также отобразить стрелки скорости и ускорения.

**Закон движения точки:**

r(t) = 1 + sin(8t); phi(t) = t + 0.5sin(8t)

**Текст программы**

Основная:

import numpy as np  
import matplotlib.pyplot as plt  
from matplotlib.animation import FuncAnimation  
import sympy as sp  
import math  
  
# var 4: r(t) = 1 + sin8t; phi(t) = t + 0.5sin8t;  
  
# Функция поворота на угол альфа  
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'*)*phi = *(*t + 0.5\*sp.sin*(*8 \* t*))*# Переход из полярных координат в Декартовы координаты  
x = *(*1 + sp.sin*(*8 \* t*))* \* sp.cos*(*phi*)*y = *(*1 + sp.sin*(*8 \* t*))* \* sp.sin*(*phi*)*# Вычисление скорости по x  
Vx = sp.diff*(*x, t*)*# Вычисление скорости по y  
Vy = sp.diff*(*y, t*)*# Вычисление общей скорости  
Vmod = sp.sqrt*(*Vx\*Vx+Vy\*Vy*)*# Вычисление ускорения по x  
Wx = sp.diff*(*Vx, t*)*# Вычисление ускорения по y  
Wy = sp.diff*(*Vy, t*)*# Вычисление общего ускорения  
Wmod = sp.sqrt*(*Wx\*Wx+Wy\*Wy*)*# Вычисление тангенсального ускорения  
Wtau = sp.diff*(*Vmod, t*)*# Вычесление радиуса кривизны  
rho = *(*Vmod\*Vmod*)*/sp.sqrt*(*Wmod\*Wmod-Wtau\*Wtau*)*# Равномерное распределение по массиву тысячи чисел от 0 до 10.  
T = np.linspace*(*0, 10, 1000*)* # возвращает одномерный массив из указанного количества элементов, равномерно распределенных внутри заданного интервала  
# Заполнение массивов нулями, в соответствии с массивом T  
X = np.zeros\_like*(*T*)* # zeros\_like возвращает новый массив из нулей с формой и типом данных указанного массива а   
Y = np.zeros\_like*(*T*)*VX = np.zeros\_like*(*T*)*VY = np.zeros\_like*(*T*)*WY = np.zeros\_like*(*T*)*WX = np.zeros\_like*(*T*)*Rho = np.zeros\_like*(*T*)*Phi = np.zeros\_like*(*T*)*# Меняем переменную x на t, а t меняем на значение из T[i]. По сути, подстановка значения в символьную переменную  
for i in np.arange*(*len*(*T*))*: # возвращает одномерный массив с равномерно разнесенными значениями внутри заданного 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*])* WY*[*i*]* = sp.Subs*(*Wy, t, T*[*i*])* WX*[*i*]* = sp.Subs*(*Wx, t, T*[*i*])* Rho*[*i*]* = sp.Subs*(*rho, t, T*[*i*])* Phi*[*i*]* = sp.Subs*(*phi, t, T*[*i*])*# 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 абцисс'*)*ax1.set\_ylabel*(*'ось ординат'*)*# Построение траектории  
ax1.plot*(*X, Y*)*# Построение точки  
P, = ax1.plot*(*X*[*0*]*, Y*[*0*]*, marker='o'*)*# Построение векторов ускорения, скорости, радиуса-кривизны, радиуса-вектора  
WLine, = ax1.plot*([*X*[*0*]*, X*[*0*]*+WX*[*0*]]*, *[*Y*[*0*]*, Y*[*0*]*+WY*[*0*]]*, 'g', label = 'Вектор 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*]* + *(*Y*[*0*]* + VY*[*0*])* \* Rho*[*0*]* / sp.sqrt*((*Y*[*0*]* + VY*[*0*])*\*\*2 +  
*(*X*[*0*]* + VX*[*0*])*\*\*2*)]*, *[*Y*[*0*]*, Y*[*0*]* - *(*X*[*0*]* + VX*[*0*])* \* Rho*[*0*]* /  
sp.sqrt*((*Y*[*0*]* + VY*[*0*])*\*\*2 + *(*X*[*0*]* + VX*[*0*])*\*\*2*)]*, 'b', label = 'Вектор кривизны'*)*RLine, = ax1.plot*([*0, X*[*0*]]*, *[*0, Y*[*0*]]*, 'black', label = 'Радиус-вектор'*)*R = math.sqrt*(*math.pow*(*X*[*0*]*, 2*)* + math.pow*(*Y*[*0*]*, 2*))*ax1.set*(*xlim=*[*-4, 4*]*, ylim=*[*-4, 4*])*;  
# Построение стрелочек на концах векторах  
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*]*, 'r'*)*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*]*, 'g'*)*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*]))*RArrow, = ax1.plot*(*RArrowRx + X*[*0*]*, RArrowRy + Y*[*0*]*, 'black'*)*ArrowRhoX = np.array*([*-0.2\*R, 0, -0.2\*R*])*ArrowRhoY = np.array*([*0.1\*R, 0, -0.1\*R*])*ux = Rho*[*0*]*\**(*Y*[*0*]* + VY*[*0*])*/math.sqrt*(*math.pow*(*X*[*0*]* + VX*[*0*]*, 2*)*+math.pow*(*Y*[*0*]* + VY*[*0*]*, 2*))*uy = Rho*[*0*]*\**(*X*[*0*]* + VX*[*0*])*/math.sqrt*(*np.power*(*X*[*0*]* + VX*[*0*]*, 2*)*+math.pow*(*Y*[*0*]* + VY*[*0*]*, 2*))*RArrowRhox, RArrowRhoy = Rot2D*(*ArrowRhoX, ArrowRhoY, math.atan2*(*-uy, ux*))*ArrowRho, = ax1.plot*(*RArrowRhox + X*[*0*]* + ux, RArrowRhoy + Y*[*0*]* - uy, 'b'*)*# Вывод легенды на график  
ax1.legend*(* ncol = 2, # количество столбцов  
 facecolor = 'oldlace', # цвет области  
 edgecolor = 'r', # цвет крайней линии  
 *)*# Функция для анимации  
def anima*(*i*)*:  
 ux = Rho*[*i*]* \* *(*Y*[*i*]* + VY*[*i*])* / math.sqrt*(*math.pow*(*X*[*i*]* + VX*[*i*]*, 2*)* + math.pow*(*Y*[*i*]* + VY*[*i*]*, 2*))* uy = Rho*[*i*]* \* *(*X*[*i*]* + VX*[*i*])* / math.sqrt*(*np.power*(*X*[*i*]* + VX*[*i*]*, 2*)* + math.pow*(*Y*[*i*]* + VY*[*i*]*, 2*))* P.set\_data*(*X*[*i*]*, Y*[*i*])* VLine.set\_data*([*X*[*i*]*, X*[*i*]*+VX*[*i*]]*, *[*Y*[*i*]*, Y*[*i*]*+VY*[*i*]])* Rholine.set\_data*([*X*[*i*]*, X*[*i*]* + ux*]*, *[*Y*[*i*]*, Y*[*i*]* - uy*])* WLine.set\_data*([*X*[*i*]*,X*[*i*]*+WX*[*i*]]*,*[*Y*[*i*]*,Y*[*i*]*+WY*[*i*]])* RLine.set\_data*([*0, X*[*i*]]*, *[*0, Y*[*i*]])* RArrowX, RArrowY = Rot2D*(*ArrowX, ArrowY, math.atan2*(*VY*[*i*]*, VX*[*i*]))* RWArrowX, RWArrowY = Rot2D*(*WArrowX, WArrowY, math.atan2*(*WY*[*i*]*, WX*[*i*]))* RArrowRx, RArrowRy = Rot2D*(*ArrowRx, ArrowRy, math.atan2*(*Y*[*i*]*, X*[*i*]))* RArrowRhox, RArrowRhoy = Rot2D*(*ArrowRhoX, ArrowRhoY, math.atan2*(*-uy, ux*))* ArrowRho.set\_data*(*RArrowRhox + X*[*i*]* + ux, RArrowRhoy + Y*[*i*]* - uy*)* VArrow.set\_data*(*RArrowX + X*[*i*]*+VX*[*i*]*, RArrowY + Y*[*i*]*+VY*[*i*])* WArrow.set\_data*(*RWArrowX+X*[*i*]*+WX*[*i*]*, RWArrowY+Y*[*i*]*+WY*[*i*])* RArrow.set\_data*(*RArrowRx + X*[*i*]*, RArrowRy + Y*[*i*])* return P, VLine, Rholine, VArrow, WLine, WArrow, RLine, RArrow, ArrowRho,  
# animation function  
anim = FuncAnimation*(*fig, anima, frames=1000, interval=20, blit=True*)*plt.show*()*

**Функция Rot2D:**

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

**Результат работы программы:**

|  |  |
| --- | --- |
|  |  |
|  |  |