Московский авиационный институт

(национальный исследовательский университет)

Институт № 8 «Компьютерные науки и прикладная математика»

**Лабораторная работа №1**

**по курсу «Теоретическая механика»**

**Анимация точки**

Выполнил студент группы М8О-215Б-23

Кармишен Егор Сергеевич

Преподаватель: Волков Евгений Валерьевич

Оценка:

Дата:

Москва, 2024

**Вариант № «11»**

**Задание:**

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

**Закон движения точки:**

**r(t) = 2 + cos6t**

**ф(t) = 7t + 1.2cos6t**

**Текст программы**

import numpy as np

import sympy

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from sympy.utilities.lambdify import implemented\_function

*def* r(*t*):

*# Функция радиус-вектора материальной точки от времени*

return 2 + sympy.cos(6 \* t)

*def* phi(*t*):

*# Функция угла материальной точки от времени*

return 7 \* t + 1.2 \* sympy.cos(6 \* t)

*def* Rot2D(*X*, *Y*, *phi*):

*# Поворот двумерной ДСК с помощью матрицы поворота*

X\_r = X \* np.cos(phi) - Y \* np.sin(phi)

Y\_r = X \* np.sin(phi) + Y \* np.cos(phi)

return X\_r, Y\_r

*def* main():

*# Переменная времени t имеет символьный sympy тип*

t = sympy.Symbol("t")

*# Переход от ДСК к ПСК*

x = r(t) \* sympy.cos(phi(t))

y = r(t) \* sympy.sin(phi(t))

*# Определение скорости и ускорения как производных координаты*

Vx = sympy.diff(x, t)

Vy = sympy.diff(y, t)

Wx = sympy.diff(Vx, t)

Wy = sympy.diff(Vy, t)

*# Создание функций физических величин, зависящих от времени*

F\_x = sympy.lambdify(t, x, "numpy")

F\_y = sympy.lambdify(t, y, "numpy")

F\_Vx = sympy.lambdify(t, Vx, "numpy")

F\_Vy = sympy.lambdify(t, Vy, "numpy")

F\_Wx = sympy.lambdify(t, Wx, "numpy")

F\_Wy = sympy.lambdify(t, Wy, "numpy")

*# Указание временного начала и конца моделирования*

time\_steps\_amount = 1000

T = np.linspace(0, 4 \* np.pi, time\_steps\_amount)

*# Указание значения физических величин в каждый момент времени*

X = F\_x(T) *# Координата по оси OX*

Y = F\_y(T) *# Координата по оси OY*

VX = F\_Vx(T) *# Скорость по оси OX*

VY = F\_Vy(T) *# Скорость по оси OY*

V\_phi = np.arctan2(VY, VX) *# Направление (угол) вектора скорости*

WX = F\_Wx(T) *# Ускорение по оси OX*

WY = F\_Wy(T) *# Ускорение по оси OY*

W\_phi = np.arctan2(WY, WX) *# Направление (угол) вектора ускорения*

*# Вектор нормального ускорения перпендикулярен вектору линейной скорости и лежит на радиусе кривизны*

P\_phi = V\_phi + np.pi/2

*# Масштаб вектора V (если он нерепрезентативно мал или велик)*

V\_scale = 1/10

*# Масштаб вектора W (если он нерепрезентативно мал или велик)*

W\_scale = 1/10

*# Масштаб отрезка P (если он нерепрезентативно мал или велик)*

P\_scale = 1/10

*# Создание окна для графика*

fig = plt.figure()

*# Создание одной ячейки для отрисовки графика*

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

*# Указываем единый масштаб по осям и другие свойства*

ax1.axis("equal")

ax1.set(

*xlim*=[-8, 8],

*ylim*=[-8, 8],

*title*=*f*"\

Радиyс-вектор точки - серый вектор (масштаб 1.0)\n\

Скорость (V) - зеленый вектор (масштаб {V\_scale})\n\

Ускорение (W) - красный вектор (масштаб {W\_scale})",

)

ax1.plot(X, Y)

*# Добавление материальной точки*

(point,) = ax1.plot(X[0], Y[0], *marker*="o")

*# Добавление радиyс-вектора материальной точки*

R\_color = [0.5, 0.5, 0.5] *# Серый цвет*

(R\_line,) = ax1.plot(

[0, X[0]],

[0, Y[0]],

*color*=R\_color,

)

*# Добавление линии вектора скорости V*

V\_color = [0, 0.7, 0] *# Зеленый цвет*

(V\_line,) = ax1.plot(

[X[0], X[0] + VX[0] \* V\_scale],

[Y[0], Y[0] + VY[0] \* V\_scale],

*color*=V\_color,

)

*# Добавление линии вектора yскорения W*

W\_color = [0.7, 0, 0] *# Красный цвет*

(W\_line,) = ax1.plot(

[X[0], X[0] + WX[0] \* W\_scale],

[Y[0], Y[0] + WY[0] \* W\_scale],

*color*=W\_color,

)

*# Добавление стрелки вектора V*

X\_arr\_V = np.multiply(V\_scale, np.array([-0.7, 0, -0.7])) *# X координаты трех точек стрелки вектора V*

Y\_arr\_V = np.multiply(V\_scale, np.array([0.2, 0, -0.2])) *# Y координаты трех точек стрелки вектора V*

RX\_V, RY\_V = Rot2D(X\_arr\_V, Y\_arr\_V, V\_phi[0]) *# Задание начального направления вектора V*

(V\_arrow,) = ax1.plot(RX\_V, RY\_V, *color*=V\_color)

*# Добавление стрелки вектора W*

X\_arr\_W = np.multiply(W\_scale, np.array([-0.7, 0, -0.7])) *# X координаты трех точек стрелки вектора W*

Y\_arr\_W = np.multiply(W\_scale, np.array([0.2, 0, -0.2])) *# Y координаты трех точек стрелки вектора W*

RX\_W, RY\_W = Rot2D(X\_arr\_W, Y\_arr\_W, W\_phi[0]) *# Задание начального направления вектора W*

(W\_arrow,) = ax1.plot(RX\_W, RY\_W, *color*=W\_color)

*def* animate(*i*):

*# Анимация материальной точки*

point.set\_data(X[i], Y[i]) *# Смена положения материальной точки*

*# Анимация радиyс-вектора материальной точки*

R\_line.set\_data([0, X[i]], [0, Y[i]])

*# Анимация вектора V*

V\_line.set\_data([X[i], X[i] + VX[i] \* V\_scale], [Y[i], Y[i] + VY[i] \* V\_scale]) *# Смена положения линии вектора V*

RX\_V, RY\_V = Rot2D(X\_arr\_V, Y\_arr\_V, V\_phi[i]) *# Смена направления вектора V*

V\_arrow.set\_data(X[i] + VX[i] \* V\_scale + RX\_V, Y[i] + VY[i] \* V\_scale + RY\_V) *# Смена положения стрелки вектора V*

*# Анимация вектора W*

W\_line.set\_data([X[i], X[i] + WX[i] \* W\_scale], [Y[i], Y[i] + WY[i] \* W\_scale]) *# Смена положения линии вектора W*

RX\_W, RY\_W = Rot2D(X\_arr\_W, Y\_arr\_W, W\_phi[i]) *# Смена направления вектора W*

W\_arrow.set\_data(X[i] + WX[i] \* W\_scale + RX\_W, Y[i] + WY[i] \* W\_scale + RY\_W) *# Смена положения стрелки вектора W*

return point, R\_line, V\_line, V\_arrow, W\_line, W\_arrow

animation = FuncAnimation(fig, animate, *frames*=time\_steps\_amount, *interval*=100)

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Результат работы программы:**

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