

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

**Лабораторная работа № 1
по курсу "Теоретическая механика
и компьютерное моделирование"**

Кинематика точки

Выполнил студент группы М8О-206Б-21
Синюков А.С.
Преподаватель: Чекина Евгения Алексеевна

Оценка:
Дата: 14.11.2022

Вариант № 19

Задание:

Написать программу на языке Python, визуализирующую кинематику движения точки. Согласно варианту нарисовать траекторию точки, анимировать движение точки, в каждый момент времени отобразить векторы скорости, тангенциального и нормального ускорений, а также радиус кривизны.

Закон движения точки:

$$r(t) = 1 + 1.5\sin(12t), \phi(t) = 1.25t + 0.2\cos(12t)$$

Текст программы:

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

#вариант 19
#r = 1 + 1.5sin12t
#phi = 1.25t + 0.2cos12t

#поворот на угол alpha
def rotate(X, Y, Alpha):
    RX = X*np.cos(Alpha) - Y*np.sin(Alpha)
    RY = X*np.sin(Alpha) + Y*np.cos(Alpha)
    return RX, RY

def angle(a_x, a_y, b_x, b_y):
    ab = a_x * b_x + a_y * b_y
    mod_a = math.sqrt(a_x * a_x + a_y * a_y)
    mod_b = math.sqrt(b_x * b_x + b_y * b_y)
    return ab / (mod_a * mod_b)

t = sp.Symbol('t')
#перевод полярных координат в Декартовы
x = (1 + 1.5*sp.sin(12*t))*sp.cos(1.25*t + 0.2*sp.cos(12*t))
y = (1 + 1.5*sp.sin(12*t))*sp.sin(1.25*t + 0.2*sp.cos(12*t))
phi = 1.25*t + 0.2*sp.cos(12*t)

#скорость
v_x = sp.diff(x, t)
v_y = sp.diff(y, t)
v_mod = sp.sqrt(v_x*v_x + v_y*v_y)

#ускорение
w_x = sp.diff(v_x, t)
```

```

w_y = sp.diff(v_y, t)
w_mod = sp.sqrt(w_x*w_x + w_y*w_y)

#эволюта (радиус кривизны)
evo_x = -sp.diff(y, t)*(sp.diff(x)**2 + sp.diff(y)**2)/(sp.diff(x, t)*sp.diff(y, t, 2) - sp.diff(x, t, 2)*sp
evo_y = sp.diff(x, t)*(sp.diff(x)**2 + sp.diff(y)**2)/(sp.diff(x, t)*sp.diff(y, t, 2) - sp.diff(x, t, 2)*sp
evo_mod = sp.sqrt(evo_x*evo_x + evo_y*evo_y)

#тангенсальное ускорение
cos_w_v = (v_x * w_x + v_y * w_y) / (v_mod * w_mod)
w_tau_x = (v_x / v_mod) * w_mod * cos_w_v
w_tau_y = (v_y / v_mod) * w_mod * cos_w_v

#нормальное ускорение
cos_w_evo = sp.sqrt(1 - cos_w_v*cos_w_v)
w_nor_x = (evo_x / evo_mod) * w_mod * cos_w_evo
w_nor_y = (evo_y / evo_mod) * w_mod * cos_w_evo

#1000 чисел от 1 до 10
T = np.linspace(0, 10, 1000)

#куча нулевых массивов "как" массив T
X = np.zeros_like(T)
Y = np.zeros_like(T)
V_X = np.zeros_like(T)
V_Y = np.zeros_like(T)
W_X = np.zeros_like(T)
W_Y = np.zeros_like(T)
W_N_X = np.zeros_like(T)
W_N_Y = np.zeros_like(T)
EVO_X = np.zeros_like(T)
EVO_Y = np.zeros_like(T)
Phi = np.zeros_like(T)

#вычисление значений функций в каждой точке T
for i in np.arange(len(T)):
    X[i] = sp.Subs(x, t, T[i])
    Y[i] = sp.Subs(y, t, T[i])
    V_X[i] = sp.Subs(v_x, t, T[i])
    V_Y[i] = sp.Subs(v_y, t, T[i])
    W_X[i] = sp.Subs(w_tau_x, t, T[i])
    W_Y[i] = sp.Subs(w_tau_y, t, T[i])
    W_N_X[i] = sp.Subs(w_nor_x, t, T[i])
    W_N_Y[i] = sp.Subs(w_nor_y, t, T[i])
    EVO_X[i] = sp.Subs(evo_x, t, T[i])
    EVO_Y[i] = sp.Subs(evo_y, t, T[i])
    Phi[i] = sp.Subs(phi, t, T[i])

```

```

#создание окна, поля с графиком и его параметров
fig = plt.figure(figsize=(10, 5))

ax1 = fig.add_subplot(1, 1, 1)
ax1.axis('equal')
ax1.set_title("Модель движения точки")
ax1.set_xlabel('X')
ax1.set_ylabel('Y')

ax1.plot(X, Y)

P, = ax1.plot(X[0], Y[0], 'black', marker='o') #движущаяся точка
Rho_Dot, = ax1.plot(X[0], Y[0], 'b', marker='o')

V_Line, = ax1.plot([X[0], X[0] + V_X[0]], [Y[0], X[0] + V_X[0]], 'r', label = 'Скорость')
W_Line, = ax1.plot([X[0], X[0] + W_X[0]], [Y[0], Y[0] + W_Y[0]], 'g', label = 'Тангенсальное ускорение')
Rho_Line, = ax1.plot([X[0], X[0] + EVO_X[0]], [Y[0], Y[0] + EVO_Y[0]], 'b', label = 'Радиус кривизны')
R_Line, = ax1.plot([0, X[0]], [0, Y[0]], 'black', label = 'Радиус-вектор')
W_N_Line, = ax1.plot([X[0], X[0] + W_N_X[0]], [Y[0], Y[0] + W_N_Y[0]], 'y', label = 'Нормальное ускорение')

R = math.sqrt(math.pow(X[0], 2) + math.pow(Y[0], 2))

#построение стрелочек
Arrow_X = np.array([-0.2*R, 0, -0.2*R])
Arrow_Y = np.array([0.1*R, 0, -0.1*R])
R_Arrow_X, R_Arrow_Y = rotate(Arrow_X, Arrow_Y, math.atan2(V_Y[0], V_X[0]))
V_Arrow, = ax1.plot(R_Arrow_X + X[0] + V_X[0], R_Arrow_Y + Y[0] + V_Y[0], 'r')

W_Arrow_X = np.array([-0.2*R, 0, -0.2*R])
W_Arrow_Y = np.array([0.1*R, 0, -0.1*R])
R_W_Arrow_X, R_W_Arrow_Y = rotate(W_Arrow_X, W_Arrow_Y, math.atan2(W_Y[0], W_X[0]))
W_Arrow, = ax1.plot(R_W_Arrow_X + X[0] + W_X[0], R_W_Arrow_Y + Y[0] + W_Y[0], 'g')

Arrow_R_X = np.array([-0.2*R, 0, -0.2*R])
Arrow_R_Y = np.array([0.1*R, 0, -0.1*R])
R_Arrow_RX, R_Arrow_RY = rotate(Arrow_R_X, Arrow_R_Y, math.atan2(Y[0], X[0]))
R_Arrow, = ax1.plot(R_Arrow_RX + X[0], R_Arrow_RY + Y[0], 'black')

W_N_Arrow_X = np.array([-0.2*R, 0, -0.2*R])
W_N_Arrow_Y = np.array([0.1*R, 0, -0.1*R])
R_W_N_Arrow_X, R_W_N_Arrow_Y = rotate(W_N_Arrow_X, W_N_Arrow_Y, math.atan2(W_N_Y[0], W_N_X[0]))
W_N_Arrow, = ax1.plot(R_W_N_Arrow_X + X[0] + W_N_X[0], R_W_N_Arrow_Y + Y[0] + W_N_Y[0], 'y')
"""
#Если нужно, чтобы у радиуса кривизны была стрелочка (она криво висит)
#Если убрать комменты, то нужно закомментировать точку на эволюте
Arrow_Rho_X = np.array([-0.2*R, 0, -0.2*R])
Arrow_Rho_Y = np.array([0.1*R, 0, -0.1*R])
R_Arrow_Rho_X, R_Arrow_Rho_Y = rotate(Arrow_Rho_X, Arrow_Rho_Y, math.atan2(EVO_Y[0], EVO_X[0]))
Arrow_Rho = ax1.plot(R_Arrow_Rho_X + EVO_X[0], R_Arrow_Rho_Y + EVO_Y[0], 'b')

```

```

"""

ax1.legend(loc='lower center', ncol = 5, edgecolor = 'b')
ax1.set(xlim=[-4, 4], ylim=[-4, 4])

#(math.sqrt(W_X[i]*W_X[i0] + W_Y[i]*W_Y[i]))
#функция анимации
def anima(i):
    P.set_data(X[i], Y[i])
    Rho_Dot.set_data(X[i] + EVO_X[i], Y[i] + EVO_Y[i])

    W_Line.set_data([X[i], X[i] + W_X[i]], [Y[i], Y[i] + W_Y[i]])
    V_Line.set_data([X[i], X[i] + V_X[i]], [Y[i], Y[i] + V_Y[i]])
    Rho_Line.set_data([X[i], X[i] + EVO_X[i]], [Y[i], Y[i] + EVO_Y[i]])
    R_Line.set_data([0, X[i]], [0, Y[i]])
    W_N_Line.set_data([X[i], X[i] + W_N_X[i]], [Y[i], Y[i] + W_N_Y[i]])

    R_Arrow_X, R_Arrow_Y = rotate(Arrow_X, Arrow_Y, math.atan2(V_Y[i], V_X[i]))
    R_W_Arrow_X, R_W_Arrow_Y = rotate(W_Arrow_X, W_Arrow_Y, math.atan2(W_Y[i], W_X[i]))
    R_Arrow_RX, R_Arrow_RY = rotate(Arrow_RX, Arrow_RY, math.atan2(Y[i], X[i]))
    R_W_N_Arrow_X, R_W_N_Arrow_Y = rotate(W_N_Arrow_X, W_N_Arrow_Y, math.atan2(W_N_Y[i], W_N_X[i]))

    #R_Arrow_Rho_X, R_Arrow_Rho_Y = rotate(Arrow_Rho_X, Arrow_Rho_Y, math.atan2(EVO_Y[i], EVO_X[i]))

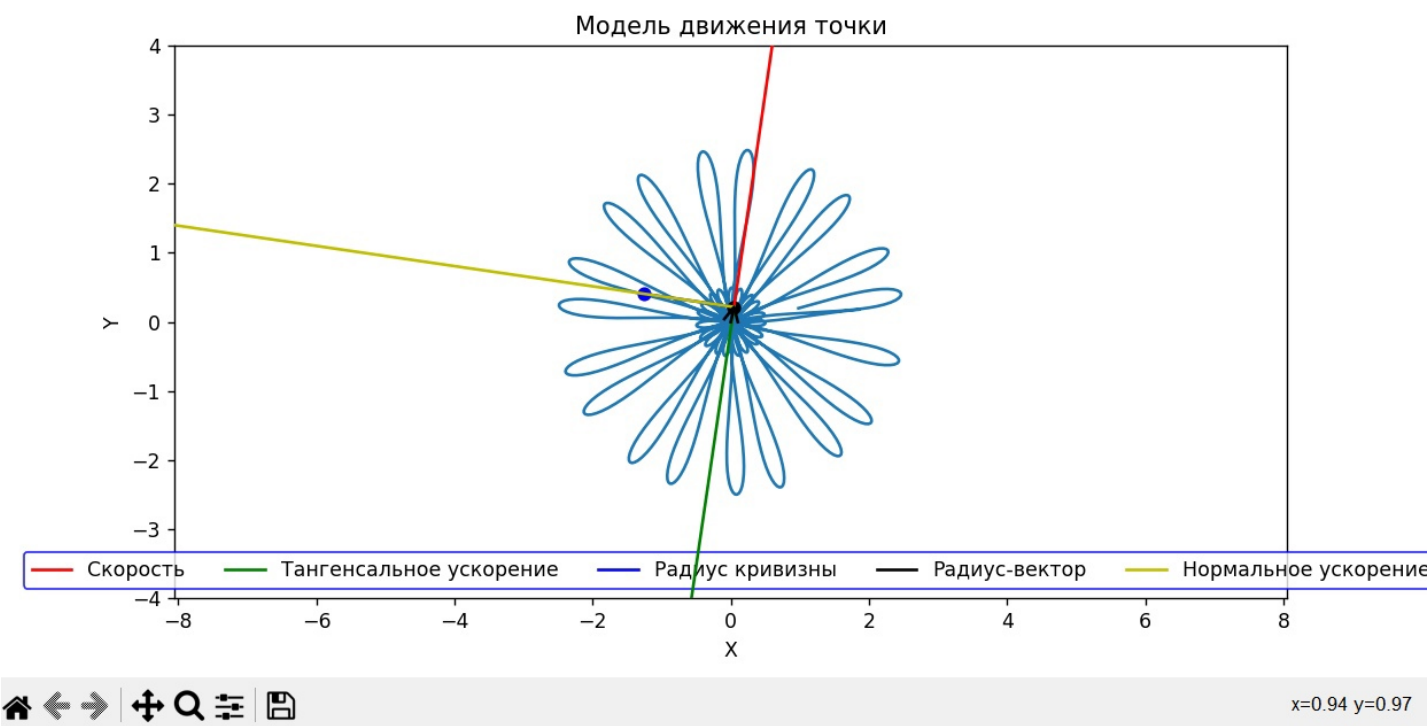
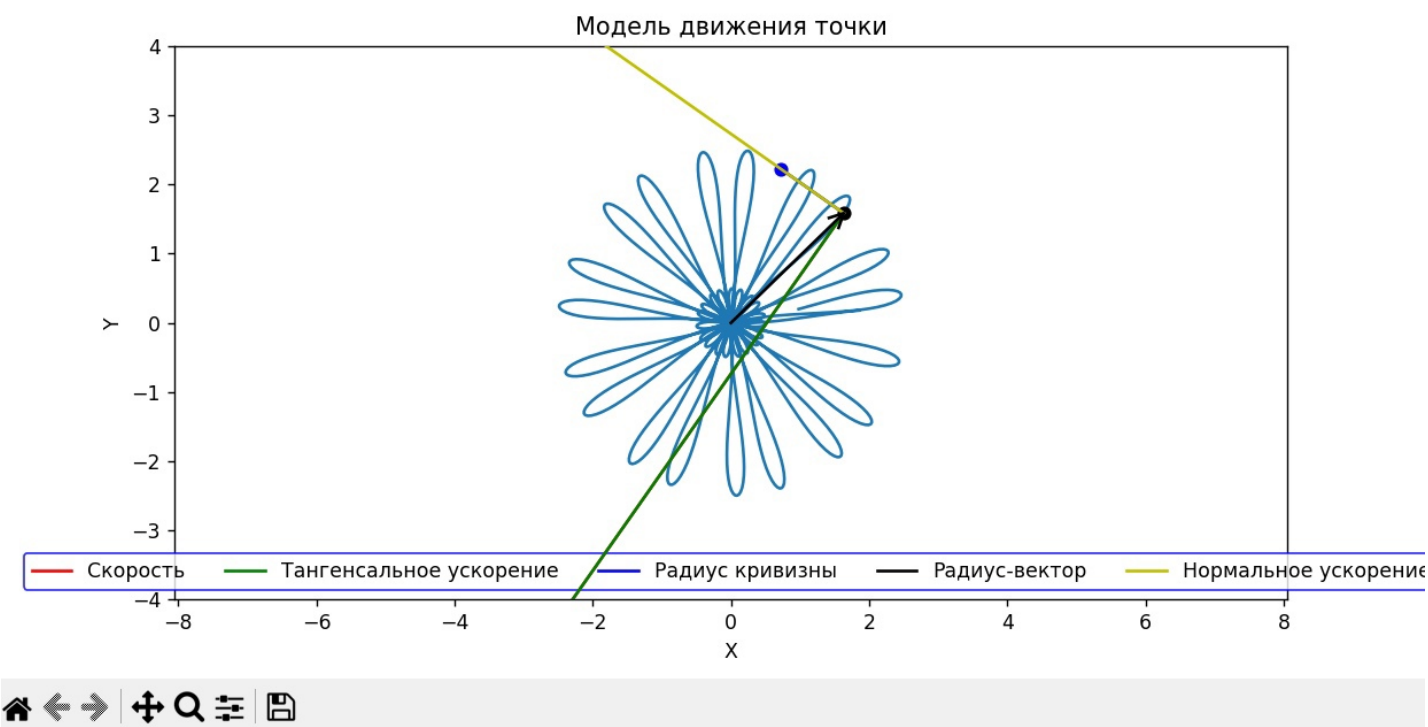
    #Arrow_Rho.set_data(R_Arrow_Rho_X + X[i] + EVO_X[i], R_Arrow_Rho_Y + Y[i] + EVO_Y[i])
    V_Arrow.set_data(R_Arrow_X + X[i] + V_X[i], R_Arrow_Y + Y[i] + V_Y[i])
    W_Arrow.set_data(R_W_Arrow_X + X[i] + W_X[i], R_W_Arrow_Y + Y[i] + W_Y[i])
    R_Arrow.set_data(R_Arrow_RX + X[i], R_Arrow_RY + Y[i])
    W_N_Arrow.set_data(R_W_N_Arrow_X + W_N_X[i], R_W_N_Arrow_Y + Y[i] + W_N_Y[i])

    return P, Rho_Dot, V_Line, W_Line, Rho_Line, R_Line, W_N_Line, V_Arrow, W_Arrow, R_Arrow, W_N_Arrow

#вывести график
anim = FuncAnimation(fig, anima, frames=1000, interval=35, blit=True)
plt.show()

```

Результат работы программы:



Модель движения точки

