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

Лабораторная работа №1
по курсу «Теоретическая механика»
Анимация точки

Выполнил студент группы М8О-205Б-20
Манташев Асадулла Уллубиевич
Преподаватель: Беличенко Михаил Валериевич

Оценка:

Дата:

Москва, 2021

Вариант № 15

Задание:

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

Закон движения точки:

$$r = 2 + sin\ 12t, \qquad \varphi = 1.8t + 0.2cos\ 12t$$

Текст программы

Основная:

```
t = sp.Symbol('t')
r = 2 + sp.sin(12*t)
phi = 1.8*t + 0.2*sp.cos(12*t)

#переход к декартовой системе координат
x = r*sp.cos(phi)
y = r*sp.sin(phi)
# скорость
Vx = sp.diff(x, t)
Vy = sp.diff(y, t)
# ускорение
Wx = sp.diff(Vx, t)
Wy = sp.diff(Vy, t)
W = sp.sqrt(Wx ** 2 + Wy ** 2)
V = sp.sqrt(Vx ** 2 + Vy ** 2)
Wt = sp.diff(V, t)
Wn = sp.sqrt(W ** 2 - Wt ** 2)
#радиус кривизны
Ro = V * V / Wn

F_x = sp.lambdify(t, x)
F_y = sp.lambdify(t, y)
F_Vx = sp.lambdify(t, Vx)
F_Vy = sp.lambdify(t, Vy)
F_Wx = sp.lambdify(t, Wx)
F_Wy = sp.lambdify(t, Wy)
F_W = sp.lambdify(t, W)
F_V = sp.lambdify(t, V)
F_Wt = sp.lambdify(t, Wt)
F_Wn = sp.lambdify(t, Wn)
F_Ro = sp.lambdify(t, Ro)

T = np.linspace(0, 10, 1001)
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)
W = np.zeros_like(T)
V = np.zeros_like(T)
Wt = np.zeros_like(T)
Wn = np.zeros_like(T)
Ro = np.zeros_like(T)
```

```
for i in np.arange(len(T)):
    X[i]  = F_x(T[i])
    Y[i]  = F_y(T[i])
    VX[i] = F_Vx(T[i])
    VY[i] = F_Vy(T[i])
    WX[i] = F_Wx(T[i])
    WY[i] = F_Wy(T[i])
    W[i] = F_W(T[i])
    V[i] = F_V(T[i])
    Wt[i] = F_Wt(T[i])
    Wn[i] = F_Wn(T[i])
    Ro[i]  = F_Ro(T[i])

PhiV = np.arctan2(VY, VX)
PhiW = np.arctan2(WY, WX)
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.axis('equal')
ax.plot(X,Y)
ax.set(xlim = [-(max(X)-min(X))/2+min(X), (max(X)-min(X))/2+max(X)],
       ylim = [-(max(Y)-min(Y))/2+min(Y), (max(Y)-min(Y))/2+max(Y)])

V_Line = ax.plot([X[0], X[0] + VX[0]], [Y[0], Y[0] + VY[0]])[0]
W_Line = ax.plot([X[0], X[0] + WX[0]], [Y[0], Y[0] + WY[0]])[0]

XArrow = np.array([-0.15, 0, -0.15])
YArrow = np.array([0.1, 0, -0.1])

RArrowX, RArrowY = Rot2D(XArrow, YArrow, PhiV[0])
RArrowXx, RArrowYy = Rot2D(XArrow, YArrow, PhiW[0])

V_Arrow = ax.plot(X[0]+RArrowX, Y[0]+RArrowY)[0]
W_Arrow = ax.plot(X[0] + WX[0]+RArrowXx, Y[0] + WY[0]+RArrowYy)[0]

R_vector = ax.plot([0, X[0]],[0, Y[0]])[0]
Rc_vector = ax.plot([X[0], X[0] + Ro[0] * VY[0] / math.sqrt(VX[0] ** 2 +
VY[0] ** 2)],
                    [Y[0], Y[0] - Ro[0] * VX[0] / math.sqrt(VX[0] ** 2 +
VY[0] ** 2)])[0]
Point = ax.plot(X[0], Y[0], marker = 'o')[0]


def BeautifulFunc(i):
    Point.set_data(X[i], Y[i])
    V_Line.set_data([X[i], X[i]+VX[i]], [Y[i], Y[i]+VY[i]])
    W_Line.set_data([X[i], X[i]+WX[i]], [Y[i], Y[i]+WY[i]])
    RArrowX, RArrowY = Rot2D(XArrow, YArrow, PhiV[i])
    RArrowXx, RArrowYy = Rot2D(XArrow, YArrow, PhiW[i])
    V_Arrow.set_data(X[i]+ VX[i]+RArrowX, Y[i]+VY[i]+RArrowY)
    W_Arrow.set_data(X[i]+ WX[i]+RArrowXx, Y[i]+WY[i]+RArrowYy)
    R_vector.set_data([0, X[i]],[0, Y[i]])
    Rc_vector.set_data([X[i], -(X[i] + Ro[i] * VY[i] / math.sqrt(VX[i] ** 2 +
VY[i] ** 2))],
                       [Y[i], -(Y[i] - Ro[i] * VX[i] / math.sqrt(VX[i] ** 2 +
VY[i] ** 2))])
    return [Point, V_Line, W_Line, V_Arrow, W_Arrow, R_vector, Rc_vector]

nechto = FuncAnimation(fig,BeautifulFunc, interval = 20, frames = len(T))
```

Функция Rot2D:

```
def Rot2D(X, Y, Phi):
    RotX = X*np.cos(Phi) + Y*np.sin(Phi)
    RotY = X*np.sin(Phi) - Y*np.cos(Phi)
    return RotX, RotY
```

Результат работы программы: