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

Лабораторная работа №3
по курсу «Теоретическая механика»
Составление и численное решения дифференциальных уравнений движения системы и ее анимация.

Выполнил студент группы М8О-206Б-20
Семин Александр Витальевич
Преподаватель: Сухов Е. А.
Дата: 16.12.21
Оценка:

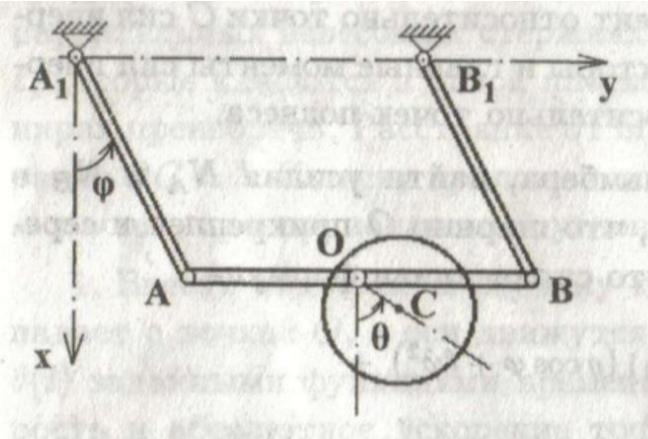
Москва, 2021

Вариант №21

Задание:

Необходимо составить и численно решить дифференциальные уравнения движения системы (уравнения Лагранжа второго рода), а затем реализовать анимацию движения механической системы используя язык программирования Python.

Механическая система:



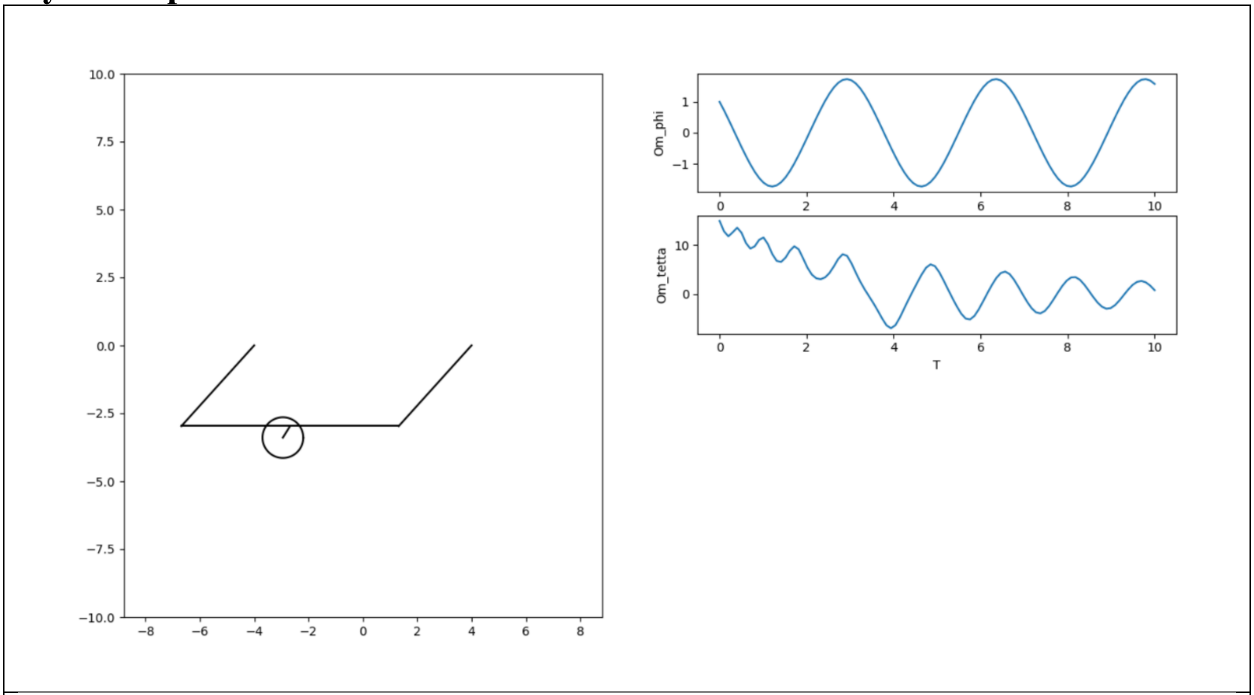
Текст программы:

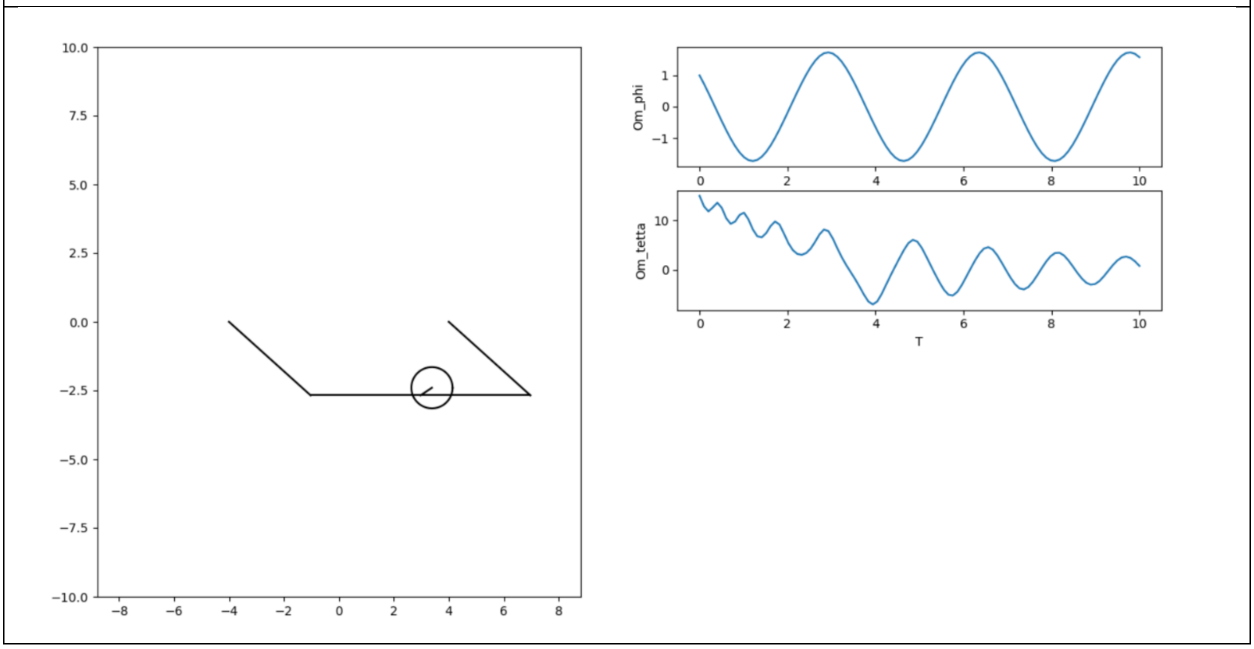
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from scipy.integrate import odeint
import sympy as sp
import math
def Circle(X, Y):
    CX = [X + 0.75 * math.cos(i/100) for i in range(0, 628)]
    CY = [Y + 0.75 * math.sin(i/100) for i in range(0, 628)]
    return CX, CY

def anima(i):
    Beam_1.set_data([- 4, - 4 + X_ab[i]], [0, Y_ab[i]])
    Beam_2.set_data([4, 4 + X_ab[i]], [0, Y_ab[i]])
    Beam_3.set_data([- 4 + X_ab[i], 4 + X_ab[i]], [Y_ab[i], Y_ab[i]])
    Beam_4.set_data([X_ab[i], X_ab[i] + X_o[i]], [Y_ab[i], Y_ab[i] +
Y_o[i]])
    circle.set_data(*Circle(X_ab[i] + X_o[i], Y_ab[i] + Y_o[i]))
    return Beam_1, Beam_2, Beam_3, Beam_4, circle,

def formY(y, t, f_om_tetta, f_om_phi):
    y1, y2, y3, y4 = y
    dydt = [y4, y3, f_om_tetta(y1, y2, y3, y4), f_om_phi(y1, y2, y3,
y4)]
    return dydt

beam_length = 4
beam_length_2 = 1.5
m_2 = 4
m_1 = 2
g = 10
l = 4
```

```
b = 0.5
k = 0.2
R = 0.75
t = sp.Symbol('t')
phi = sp.Function('phi')(t)
tetta = sp.Function('tetta')(t)
om = sp.Function('om')(t)
om_2 = sp.Function('om_2')(t)
E_ab = (m_2 * om**2 * l**2) / 2
Vx_c = om * l * sp.sin(phi)
Vy_c = om * l * sp.cos(phi)
Vx_o = om_2 * sp.sin(tetta) * b
Vy_o = om_2 * sp.cos(tetta) * b
J_cir = (m_1 * R**2) / 2
E_cir = m_1 * ((Vx_o + Vx_c)**2 + (Vy_o + Vy_c)**2) / 2 + (J_cir *
om_2**2)/2
Ekin = E_ab + E_cir
Pi1 = m_2 * g * (l * (1 - sp.cos(phi)) + b)
Pi2 = m_1 * g * (l * (1 - sp.cos(phi)) + b * (1 - sp.cos(tetta)))
Epot = Pi1 + Pi2
M = - k * om_2
L = Ekin - Epot
#equations
ur1 = sp.diff(sp.diff(L, om_2), t) - sp.diff(L, tetta) - M
ur2 = sp.diff(sp.diff(L, om), t) - sp.diff(L, phi)
#isolating second derivatives(dV/dt and dom/dt) using Kramer's method
a11 = ur1.coeff(sp.diff(om_2, t), 1)
a12 = ur1.coeff(sp.diff(om, t), 1)
a21 = ur2.coeff(sp.diff(om_2, t), 1)
a22 = ur2.coeff(sp.diff(om, t), 1)
b1 = -(ur1.coeff(sp.diff(om_2, t), 0)).coeff(sp.diff(om, t),
0).subs([(sp.diff(tetta, t), om_2), (sp.diff(phi, t), om)])
b2 = -(ur2.coeff(sp.diff(om_2, t), 0)).coeff(sp.diff(om, t),
0).subs([(sp.diff(tetta, t), om_2), (sp.diff(phi, t), om)])
detA = a11*a22-a12*a21
detA1 = b1*a22-b2*a21
detA2 = a11*b2-b1*a21
dom_tettadt = detA2/detA
dom_phidt = detA1/detA
countOfFrames = 100
# Constructing the system of differential equations
T = np.linspace(0, 10, countOfFrames)
# Pay attention here, the function lambdify translate function from
the sympy to numpy and then form arrays much more
# faster then we did using subs in previous lessons!
f_om_tetta = sp.lambdify([phi, tetta, om_2, om], dom_tettadt, "numpy")
f_om_phi = sp.lambdify([phi, tetta, om_2, om], dom_phidt, "numpy")
y0 = [0.75, 0.75, 15, 1]
sol = odeint(formY, y0, T, args = (f_om_phi, f_om_tetta))

X_ab_func = sp.lambdify(phi, sp.sin(phi) * l)
Y_ab_func = sp.lambdify(phi, - sp.cos(phi) * l)

X_o_func = sp.lambdify(tetta, sp.sin(tetta) * b)
Y_o_func = sp.lambdify(tetta, - sp.cos(tetta) * b)

#constructing corresponding arrays (Very fast, thanks to lambdify)
X_ab = X_ab_func(sol[:, 0])
```

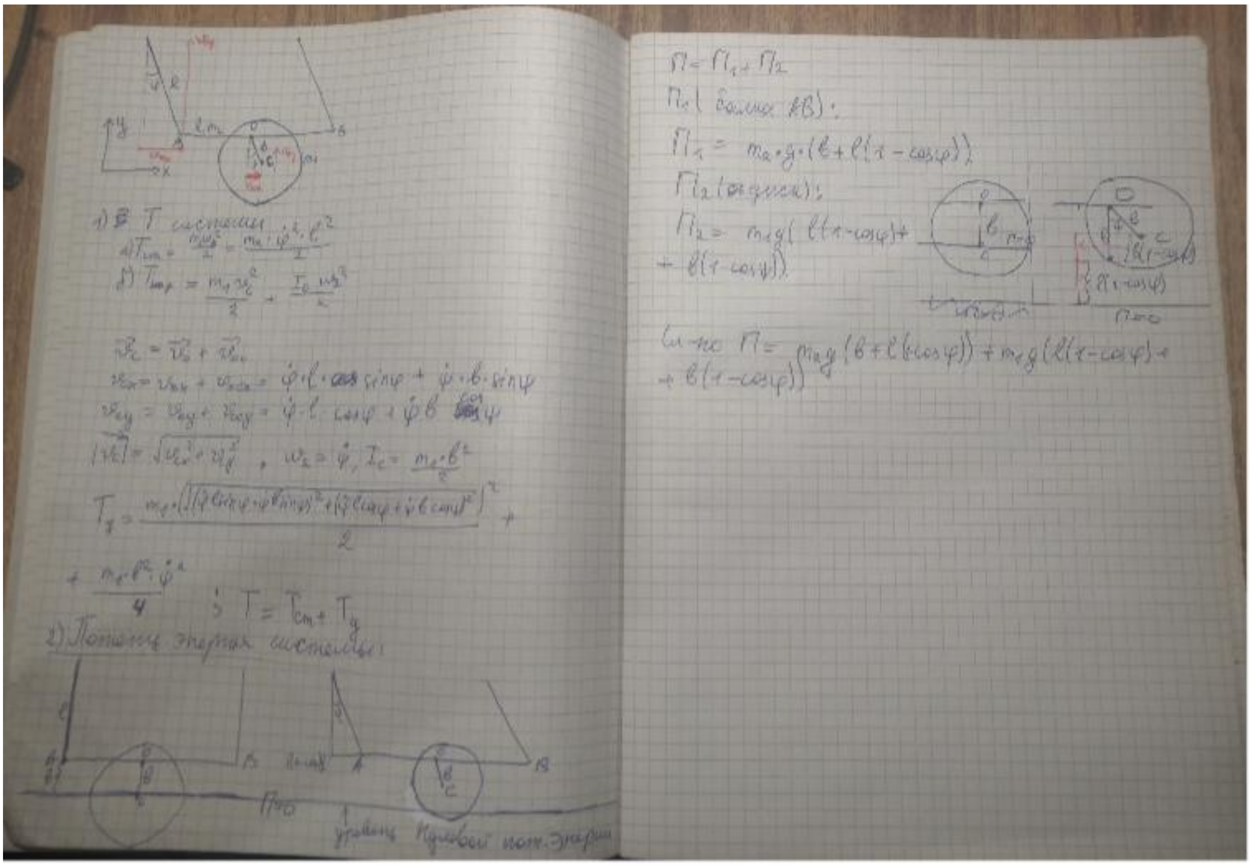
```
Y_ab = Y_ab_func(sol[:, 0])
Phi = sol[:, 1]
X_o = X_o_func(sol[:, 1])
Y_o = Y_o_func(sol[:, 1])
fig = plt.figure(figsize = (17, 8))
ax1 = fig.add_subplot(1, 2, 1)
ax1.axis('equal')
ax1.set(xlim = [-10, 10], ylim = [-10, 10])
ax2 = fig.add_subplot(4, 2, 2)
ax2.plot(T, sol[:, 3])
ax2.set_xlabel('T')
ax2.set_ylabel('Om_phi')
ax3 = fig.add_subplot(4, 2, 4)
ax3.plot(T, sol[:, 2])
ax3.set_xlabel('T')
ax3.set_ylabel('Om_tetta')

Beam_1, = ax1.plot([-2, -2 + X_ab[0]], [0, Y_ab[0]], 'black')
Beam_2, = ax1.plot([2, 2 + X_ab[0]], [0, Y_ab[0]], 'black')
Beam_3, = ax1.plot([- 4 + X_ab[0], 4 + X_ab[0]], [Y_ab[0], Y_ab[0]],
'black')
Beam_4, = ax1.plot([X_ab[0], X_ab[0] + X_o[0]], [Y_ab[0], Y_ab[0] +
Y_o[0]], 'black')
circle, = ax1.plot(*Circle(X_ab[0] + X_o[0], Y_ab[0] + Y_o[0]),
'black')
anim = FuncAnimation(fig, anima, frames=countOfFrames, interval=100,
blit=True)
plt.show()
```

Результат работы:





Вывод формул:



Вывод:

В ходе выполнения работы я столкнулся с решением уравнений Лагранжа второго рода и, освежив знания по их решению, написал программу для анимации системы своего варианта. Уравнения Лагранжа второго рода являются хорошим способом составления уравнений движения механических систем.