Московский авиационный институт (национальный исследовательский университет)

Институт № 8 «Информационные технологии и прикладная математика»

**Лабораторная работа №3**

**по курсу «Теоретическая механика» Динамика системы**

Выполнил студент группы М8О-201Б-21 Старцев Иван Романович

Преподаватель: Чекина Евгения Алексеевна

Оценка:

Дата: 07.01.2022

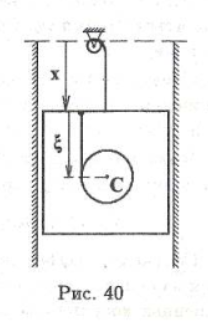
Москва, 2022

**Вариант №40**

**Задание:**

Реализовать анимацию движения системы, отобразить с помощью графиков абсолютную скорость и абсолютное ускорение грузика. Выписать уравнение Лагранжа второго рода для данной системы.

**Механическая система:**

****

**Уравнение Лагранжа:**

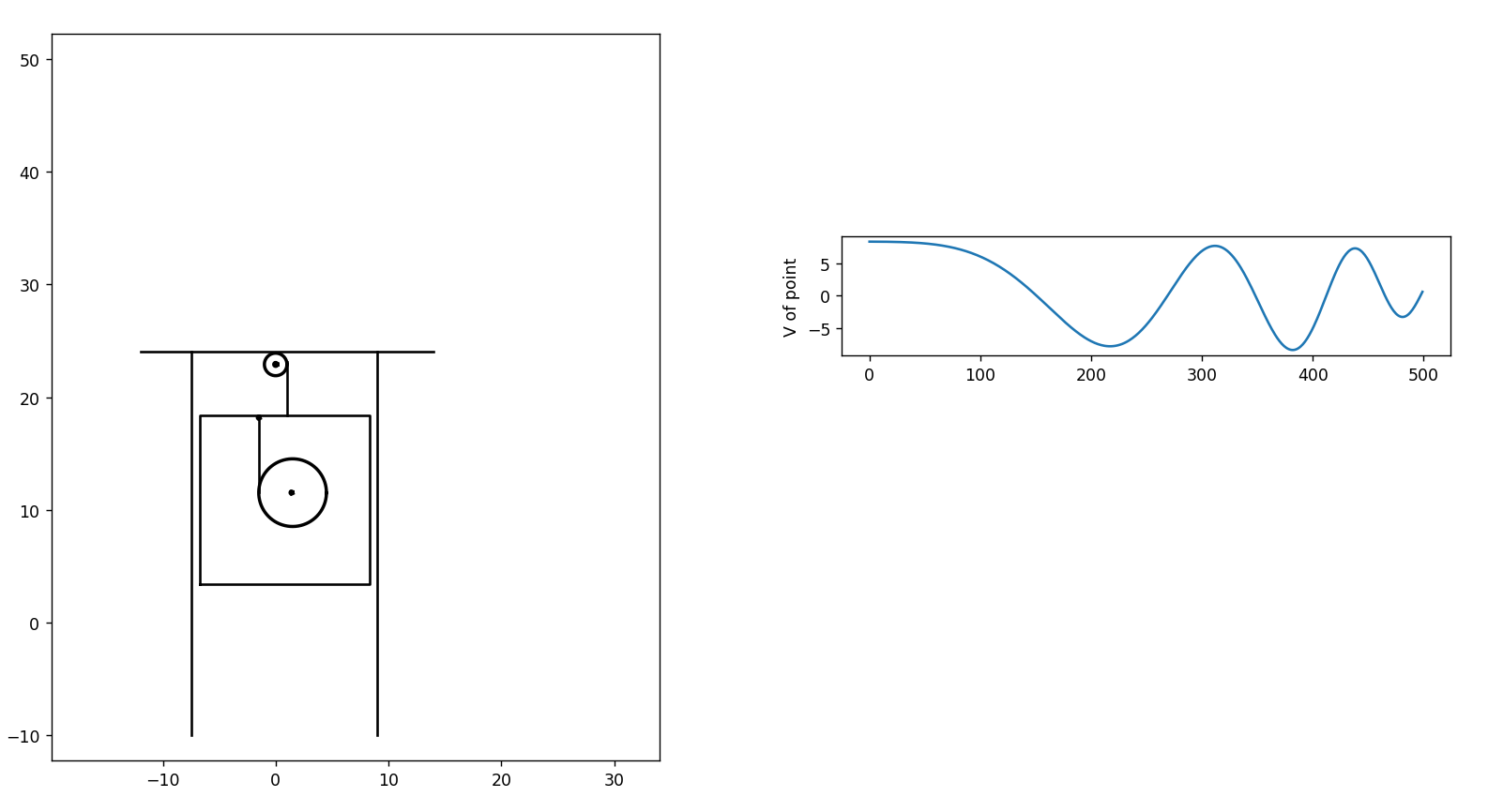
Изображение выглядит как текст, доска

Автоматически созданное описание

**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 Square(x0, y0):  
 PX = [x0 - 7.5, x0 - 7.5, x0 + 7.5, x0 + 7.5, x0 - 7.5]  
 PY = [y0 - 7.5, y0 + 7.5, y0 + 7.5, y0 - 7.5, y0 - 7.5]  
 return PX, PY  
  
def Circle(X, Y, R):  
 CX = [X + R \* math.cos(i/100) for i in range(0, 628)]  
 CY = [Y + R \* math.sin(i/100) for i in range(0, 628)]  
 return CX, CY  
  
# [Xнач, Xкон], [Yнач, Yкон] => для линии надо сделать конечным фиксированную точку, а начало привязать к вершине блока  
def anima(i):  
 PrX, PrY = Square(XR[i], YR[i])  
 Prism.set\_data(PrX, PrY)  
 Line\_upper.set\_data([XR[i] + 0.2, 1], [YR[i] + 7.5, 23.15])  
 CX, CY = Circle(XC[i] + 0.3, 1.6 \* YC[i] - 4, 3)  
 Circle\_B.set\_data(CX, CY)  
 CBX, CBY = Circle(XC[i] - 2.7, YR[i] + 7.3, 0.05)  
 Circle\_BD.set\_data(CBX, CBY)  
 Line\_bottom.set\_data([XC[i] - 2.7, XC[i] - 2.7], [1.6 \* YC[i] - 4, YR[i] + 7.5])  
 BCX, BCY = Circle(XC[i] + 0.2, 1.6 \* YC[i] - 4, 0.05)  
 Circle\_BС.set\_data(BCX, BCY)  
 return Prism, Line\_upper, Circle\_B, Circle\_BD, Line\_bottom, Circle\_BС  
  
# defining  
m = 1  
M = 100  
c = 100  
a = 0.1  
t0 = 0  
xi0 = 0.1  
dxi0dt = 0.1  
g = 9.81  
R = 1  
  
t = sp.Symbol('t')  
x = sp.Function('x')(t)  
xi = sp.Function('xi')(t)  
xH = sp.Function('xH')(t)  
xiH = sp.Function('xiH')(t)  
  
Jc = (m \* R \* R) / 2  
w = xiH / R  
T1 = (M \* xH \* xH) / 2  
T2 = (m \* (xH + xiH) \* (xH + xiH)) / 2 + (Jc \* w \* w) / 2  
  
Tc = T1 + sp.simplify(T2)  
  
Pc = -M \* g \* x - m \* g \* (x + xi) + (c \* ((x - a) \* (x - a))) / 2  
  
Lc = Tc - Pc  
  
ur1 = sp.diff(sp.diff(Lc, xH), t) - sp.diff(Lc, x)  
ur2 = sp.diff(sp.diff(Lc, xiH), t) - sp.diff(Lc, xi) - g  
  
print(ur1)  
print(ur2)  
  
a11 = ur1.coeff(sp.diff(xH, t), 1)  
a12 = ur1.coeff(sp.diff(xiH, t), 1)  
a21 = ur2.coeff(sp.diff(xH, t), 1)  
a22 = ur2.coeff(sp.diff(xiH, t), 1)  
  
b1 = -(ur1.coeff(sp.diff(xH, t), 0)).coeff(sp.diff(xiH, t), 0).subs([(sp.diff(x, t), xH), (sp.diff(xi, t), xiH)])  
b2 = -(ur2.coeff(sp.diff(xH, t), 0)).coeff(sp.diff(xiH, t), 0).subs([(sp.diff(x, t), xH), (sp.diff(xi, t), xiH)])  
  
detA = a11 \* a22 - a12 \* a21  
detA1 = b1 \* a22 - b2 \* a12  
detA2 = a11 \* b2 - b1 \* a21  
  
dxHdt = detA1 / detA  
dxiHdt = detA2 / detA  
  
  
def formY(y, t, fV, fOm):  
 y1, y2, y3, y4 = y  
 dydt = [y3, y4, fV(y1, y2, y3, y4), fOm(y1, y2, y3, y4)]  
 return dydt  
  
countOfFrames = 500  
T = np.linspace(0, 2, countOfFrames)  
  
fxH = sp.lambdify([x, xi, xH, xiH], dxHdt, "numpy")  
fxiH = sp.lambdify([x, xi, xH, xiH], dxiHdt, "numpy")  
y0 = [0.1, xi0, dxi0dt, dxi0dt]  
sol = odeint(formY, y0, T, args = (fxH, fxiH))  
  
x = sol[:,0]  
xi = sol[:,1]  
xH = sol[:,2]  
xiH = sol[:,3]  
  
XR = [0] \* len(x)  
YR = [0] \* len(x)  
XC = [0] \* len(x)  
YC = [0] \* len(x)  
l = [0] \* len(x)  
VP = [0] \* len(x)  
WP = [0] \* len(x)  
  
for i in range(len(x)):  
 XR[i] = 0.8  
 YR[i] = 3 \* (sp.cos(x[i]) + sp.cos(xi[i])) + 7.5  
 XC[i] = 1.5 \* 0.8  
 YC[i] = 2.5 \* (sp.cos(xi[i])) + 8  
 VP[i] = 3 \* (sp.cos(x[i]) + sp.cos(xi[i])) + 2.5 \* sp.cos(xi[i])  
 l[i] = i  
  
fig = plt.figure(figsize = (17, 10))  
ax1 = fig.add\_subplot(121)  
ax1.axis('equal')  
ax1.set(xlim=[x.min() - 20, x.max() + 20], ylim=[19, 21])  
  
A\_R = 1  
A\_X = 0  
A\_Y = 22.9  
Circle\_A = ax1.plot(\*Circle(A\_X, A\_Y, A\_R), 'black', linewidth=2)  
  
AD\_R = 0.1  
AD\_X = 0  
AD\_Y = 22.9  
Circle\_AD = ax1.plot(\*Circle(AD\_X, AD\_Y, AD\_R), 'black', linewidth=3)  
  
B\_R = 3  
B\_X = 0  
B\_Y = 10  
Circle\_B = ax1.plot(\*Circle(B\_X, B\_Y, B\_R), 'black', linewidth=2)[0]  
  
BD\_R = 0.05  
BD\_X = -2.5  
BD\_Y = 18.9  
Circle\_BD = ax1.plot(\*Circle(BD\_X, BD\_Y, BD\_R), 'black', linewidth=3)[0]  
  
BС\_R = 0.05  
BС\_X = x[0] + 0.2  
BС\_Y = x[0] + 1.8  
Circle\_BС = ax1.plot(\*Circle(BС\_X, BС\_Y, BС\_R), 'black', linewidth=3)[0]  
  
upper\_line\_x = [-12, 14]  
upper\_line\_y = [24, 24]  
plt.plot(upper\_line\_x, upper\_line\_y, 'black')  
side\_line1\_x = [-7.5, -7.5]  
side\_line1\_y = [-10, 24]  
plt.plot(side\_line1\_x, side\_line1\_y, 'black')  
side\_line2\_x = [9, 9]  
side\_line2\_y = [-10, 24]  
plt.plot(side\_line2\_x, side\_line2\_y, 'black')  
  
PrX, PrY = Square(x[0], x[0])  
Prism = ax1.plot(PrX, PrY, 'black')[0]  
  
Line\_upper = ax1.plot([1, 1], [22.5, 19], 'black')[0]  
Line\_bottom = ax1.plot([-1.9, -1.9], [11, 18.8], 'black')[0]  
  
ax2 = fig.add\_subplot(424)  
ax2.plot(l, VP)  
ax2.set\_ylabel('V of point')  
  
plt.subplots\_adjust(wspace = 0.3, hspace = 0.7)  
  
anim = FuncAnimation(fig, anima, frames = 320, interval = 0.01, blit = True)  
  
plt.show()

**Результат работы программы:**

****

**Вывод:**

В ходе выполнения данной лабораторной работы были получены навыки анимации системы на языке Python. Так же было получено умение выписывать уравнение Лагранжа второго рода для динамической системы.