Московский авиационный институт (национальный исследовательский университет)

Институт № 8 «Информационные технологии и прикладная математика»

**Лабораторная работа №4**

**по курсу «Теоретическая механика»**

**Положения равновесия системы**

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

Преподаватель: Чекина Евгения Алексеевна

Оценка:

Дата: 08.01.2022

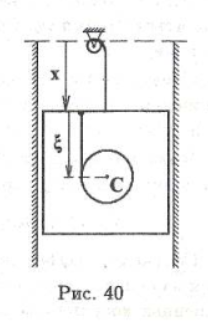
Москва, 2022

**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:**

****

**Вычисления:**

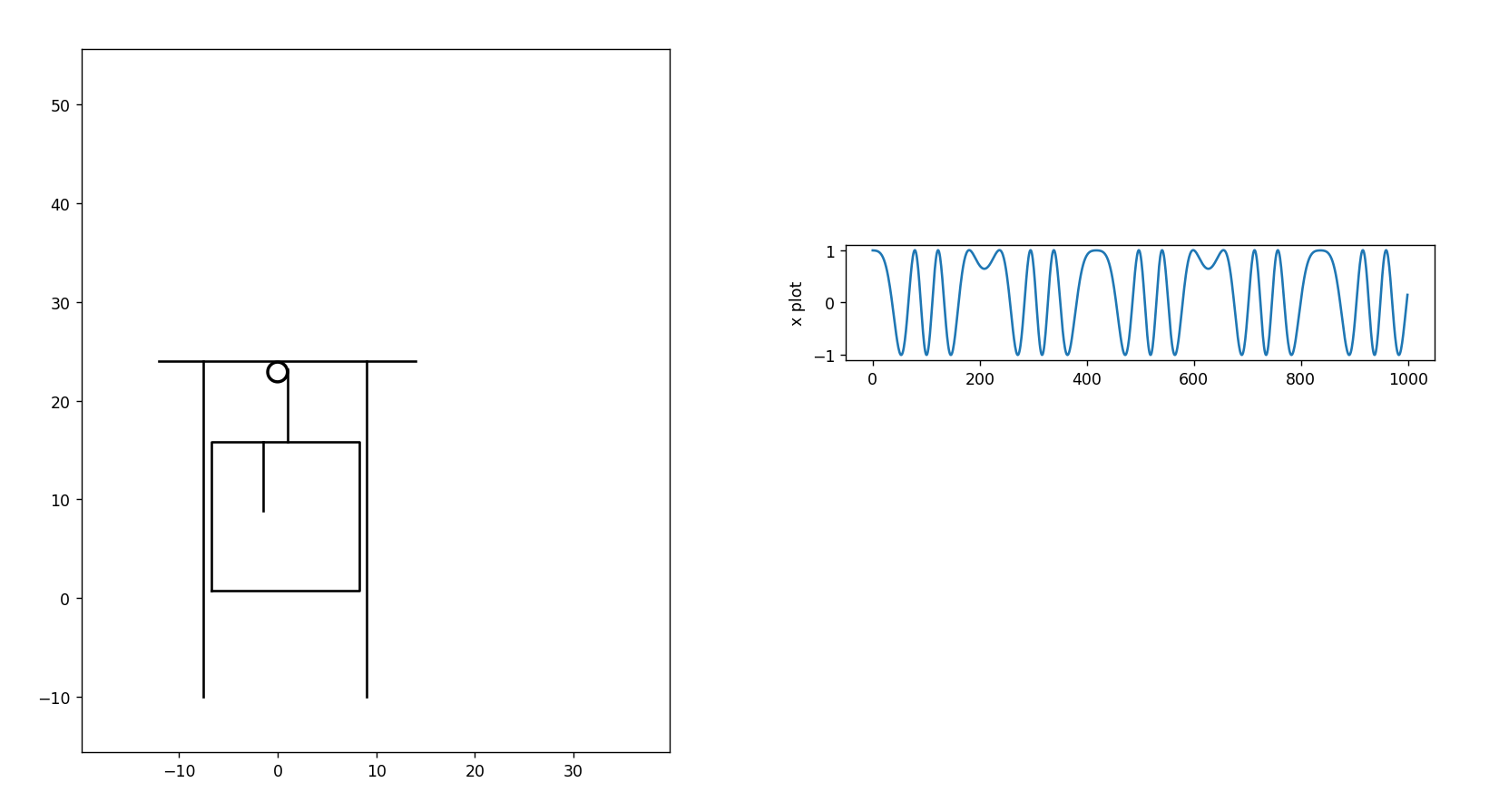
Изображение выглядит как текст

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])  
 Line\_bottom.set\_data([XC[i] - 2.7, XC[i] - 2.7], [1.6 \* YC[i] - 4, YR[i] + 7.5])  
 return Prism, Line\_upper, Line\_bottom  
  
# 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 = 0  
xH = sp.Function('xH')(t)  
xiH = 0  
  
Tc = (M \* xH \* xH) / 2  
  
Pc = -M \* g \* x + (c \* ((x - a) \* (x - a))) / 2  
  
Lc = Tc - Pc  
  
ur1 = sp.diff(sp.diff(Lc, xH), t) - sp.diff(Lc, x)  
  
a11 = ur1.coeff(sp.diff(xH, t), 1)  
  
b1 = -(ur1.coeff(sp.diff(xH, t), 0)).subs(sp.diff(x, t), xH)  
  
dxHdt = b1 / a11  
  
def formY2(y, t, fOm):  
 y1,y2 = y  
 dydt = [y2,fOm(y1,y2)]  
 return dydt  
  
countOfFrames = 1000  
T = np.linspace(0, 15, countOfFrames)  
  
fxH = sp.lambdify([x, xH], dxHdt, "numpy")  
y0 = [0.1, dxi0dt]  
sol = odeint(formY2, y0, T, args = (fxH,))  
  
x = sol[:,0]  
xH = sol[:,1]  
  
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] = sp.cos(x[i]) + 7.5  
 XC[i] = 1.5 \* 0.8  
 YC[i] = 8  
 VP[i] = sp.cos(x[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)  
  
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('x plot')  
  
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 работы я научился находить положения равновесия, проверять их на устойчивость. Экспериментально полученный период малых колебаний совпадает с теоретически выведенным значением.