МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский Авиационный Институт» (Национальный Исследовательский Университет)

Институт: №8 «Компьютерные науки и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа №3

по курсу «Теоретическая механика»

Ⅲ семестр

«Динамика системы. Расчет уравнения Лагранжа»

|  |  |
| --- | --- |
| Группа | М8О-206Б-21 |
| Студент | Чистяков К.С. |
| Преподаватель | Чекина Е.А. |
| Оценка |  |
| Дата |  |

Москва, 2023

Оглавление

[*Задание:* 3](#_Toc126762695)

[*Механическая система*: 3](#_Toc126762696)

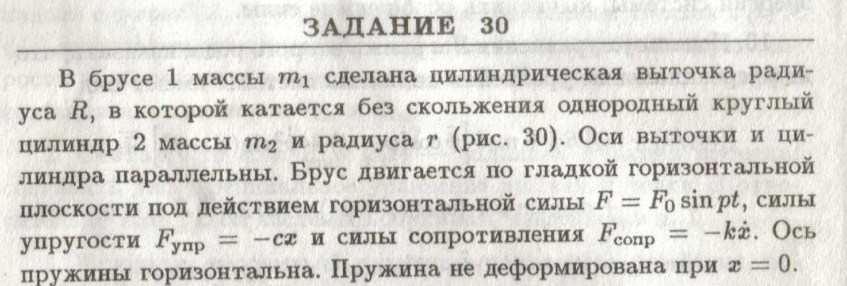
[*Вычисления:* 4](#_Toc126762697)

[*Текст программы:* 6](#_Toc126762698)

[*Результат работы программы:* 9](#_Toc126762699)

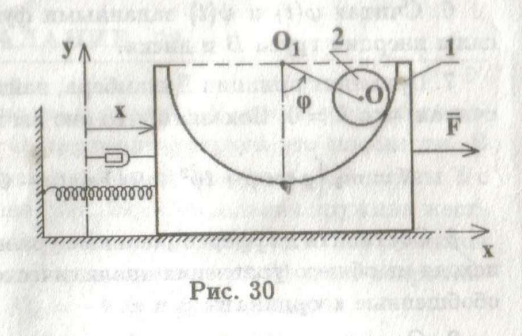
Вариант 30

# *Задание:*

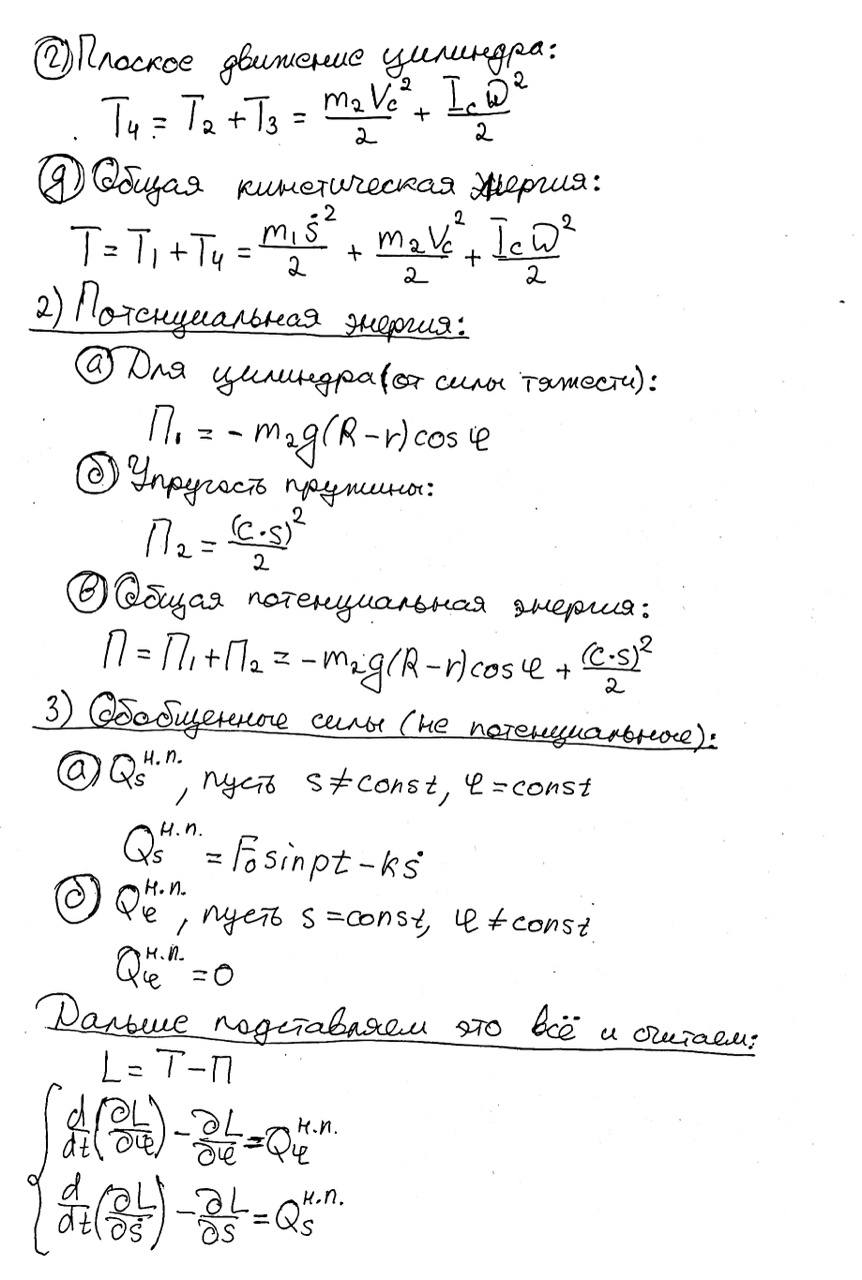
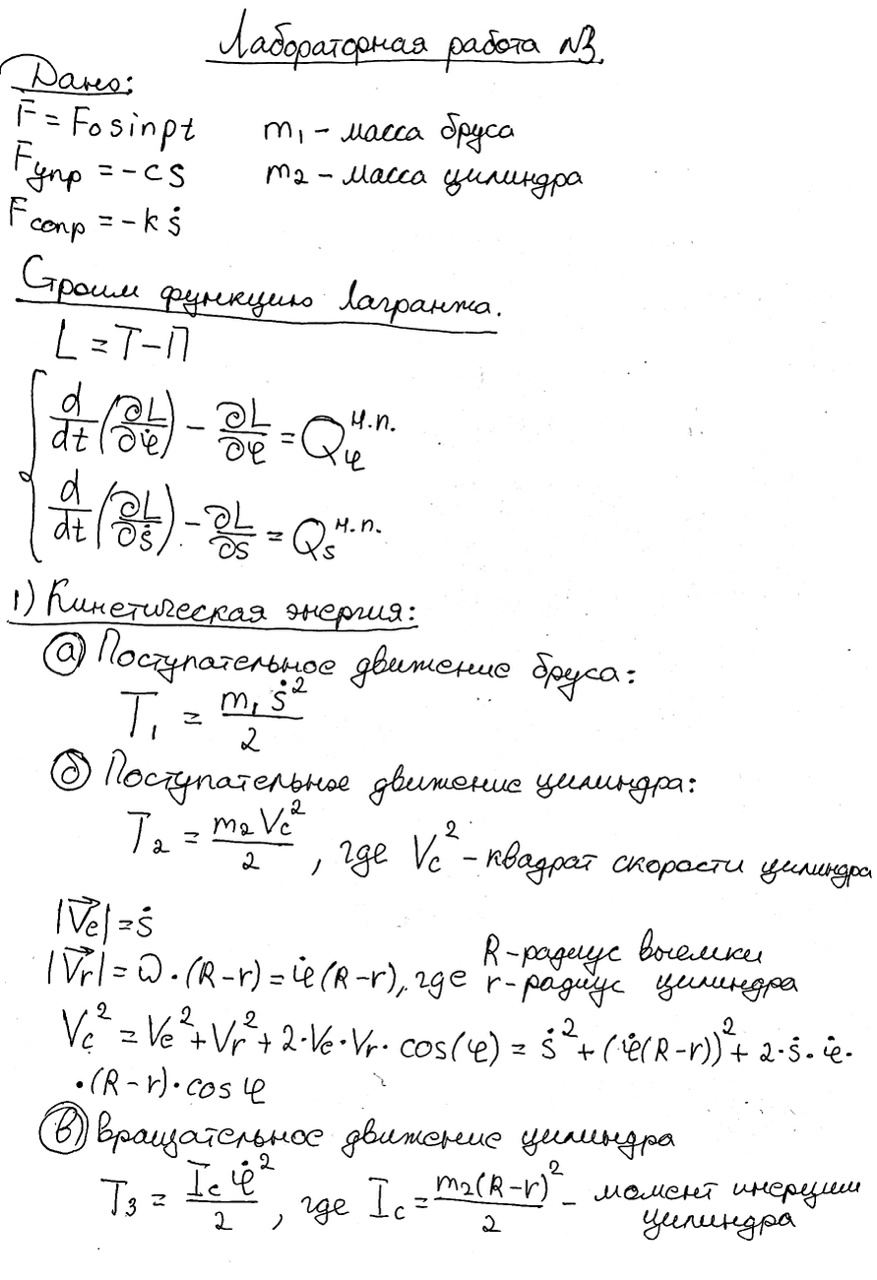


Решить уравнение Лагранжа для системы, заданной вариантом.

# *Механическая система*:

******

# *Вычисления:*



# *Текст программы:*

# Подключаем библиотеки

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

# Функция для задания шарика

def cylinder(x, y, r):

cx = [x + r \* np.sin(i / 100) for i in range(0, 628)]

cy = [y + r \* np.cos(i / 100) for i in range(0, 628)]

return (cx, cy)

# Функция для задания выемки

def recess(R):

rx = [2 + R \* np.sin(i / 100) for i in range(160, 470)]

ry = [3 + R \* np.cos(i / 100) for i in range(160, 470)]

return (rx, ry)

# Функция для системы ОДУ

def formY(y, t, fV, fOm):

y1, y2, y3, y4 = y

dydt = [y3, y4, fV(y1, y2, y3, y4, t), fOm(y1, y2, y3, y4, t)]

return dydt

# Задаем параметры системы

m1 = 5 # Масса бруса

m2 = 1 # Масса цилиндра

g = 9.81 # Ускорение свободного падения

r = 0.5 # Радиус цилиндра

R = 1.5 # Радиус внутренней выемки

# Коэффициенты

k = 2

c = 10

F0 = 5

p = 1

# Определение t как символа

t = sp.Symbol('t')

# Определяем s, phi, v=ds/dt, om=dphi/dt как функции от 't'

s = sp.Function('s')(t)

phi = sp.Function('phi')(t)

V = sp.Function('V')(t) # s с точкой

om = sp.Function('om')(t) # phi с точкой

# Строим уравнения Лагранжа

# Находим кинетическую энергию бруса

TBr = (m1 \* V\*\*2) / 2

# Вычисляем квадрат скорости цилиндра

Vc2 = V\*\*2 + (om \* (R - r))\*\*2 + 2 \* (om \* (R - r)) \* V \* sp.cos(phi)

# Момент инерции цилиндра

Jc = (m2 \* (R - r)\*\*2) / 2

# Кинетическая энергия цилиндра

TCy = (m2 \* Vc2) / 2 + (Jc \* om\*\*2) / 2

# Кинетическая энергия общая

TT = TBr + TCy

# Находим потенциальную энергию

Pi1 = -m2 \* g \* (R - r) \* (sp.cos(phi)) # Потенциальная энергии цилиндра от силы тяжести

Pi2 = (c \* s\*\*2) / 2 # Потенциальная энергия упругости пружины

Pi = Pi1 + Pi2 # Сумма потенциальных сил

# Определяем не потенциальную энергию

Qs = F0 \* sp.sin(p\*t) - k \* V

# Функция Лагранжа

L = (TT - Pi).subs([(sp.diff(s, t), V), (sp.diff(phi, t), om)])

print(L)

# Уравнения Лагранжа

ur1 = (sp.diff(sp.diff(L, V), t) - sp.diff(L, s) - Qs).subs([(sp.diff(s, t), V), (sp.diff(phi, t), om)]).simplify()

ur2 = (sp.diff(sp.diff(L, om), t) - sp.diff(L, phi)).subs([(sp.diff(s, t), V), (sp.diff(phi, t), om)]).simplify()

# Выделяем вторые производные (dV/dt и dom/dt), используя метод Крамера

a11 = ur1.coeff(sp.diff(V, t), 1)

a12 = ur1.coeff(sp.diff(om, t), 1)

a21 = ur2.coeff(sp.diff(V, t), 1)

a22 = ur2.coeff(sp.diff(om, t), 1)

b1 = -(ur1.coeff(sp.diff(V, t), 0)).coeff(sp.diff(om, t), 0).subs([(sp.diff(s, t), V), (sp.diff(phi, t), om)])

b2 = -(ur2.coeff(sp.diff(V, t), 0)).coeff(sp.diff(om, t), 0).subs([(sp.diff(s, t), V), (sp.diff(phi, t), om)])

detA = a11 \* a22 - a12 \* a21

detA1 = b1 \* a22 - b2 \* a12

detA2 = a11 \* b2 - b1 \* a21

dVdt = detA1 / detA

domdt = detA2 / detA

# Создаем систему дифференциальных уравнений

T = np.linspace(0, 10, 500)

fV = sp.lambdify([s, phi, V, om, t], dVdt, "numpy")

fOm = sp.lambdify([s, phi, V, om, t], domdt, "numpy")

y0 = [4, 1, 0, 0] # Вектор начальных значений: s(0), phi(0), v(0), om(0)

# Решение системы дифференциальных уравнений

sol = odeint(formY, y0, T, args = (fV, fOm))

# sol - Наше решение

# sol[:,0] - s

# sol[:,1] - phi

# sol[:,2] - ds/dt

# sol[:,3] - dphi/dt

# Задание пружины

n = 20 # Количество витков

k = 1 / (n - 2)

width = 0.3 # Ширина пружины

# Создание пустых массивов, заполненных нулями

Sx = np.zeros(n)

Sy = np.zeros(n)

Sx[0] = 0

Sx[n-1] = 1

Sy[0] = -0.3

Sy[n-1] = 0

for i in range(n-2):

Sx[i+1] = k \* (i + 1)

Sy[i+1] = width \* (-1)\*\*i

# Основание брусса

O1 = np.array([0, 0, 4, 4])

O2 = np.array([3, 0, 0, 3])

# Построение функций

Tension\_def = sp.lambdify(s, abs(s))

Cx\_def = sp.lambdify(phi, (R - r) \* sp.sin(phi) + 2)

Cy\_def = sp.lambdify(phi, -(R - r) \* sp.cos(phi) + 3)

Tension = Tension\_def(sol[:, 0])

C\_x = Cx\_def(sol[:, 1])

C\_y = Cy\_def(sol[:, 1])

# Рисование

fig = plt.figure(figsize=[17, 8])

# Первая часть

ax1 = fig.add\_subplot(1, 2, 1)

ax1.set(xlim=[-1, 9], ylim=[-1, 9])

ax1.set\_xlabel('Ось x')

ax1.set\_ylabel('Ось y')

OsY = ax1.plot([0, 0], [0, 5], 'black')[0]

OsX = ax1.plot([0, 8.5], [0, 0], 'black')[0]

Box = ax1.plot(O1 + Tension[0], O2, 'brown')[0]

Recess = ax1.plot(recess(R)[0] + Tension[0], recess(R)[1], 'brown')[0]

line1 = ax1.plot([Tension[0] + 0.5, Tension[0]], [3, 3], 'brown')[0]

line2 = ax1.plot([Tension[0] + 3.5, Tension[0] + 4], [3, 3], 'brown')[0]

Spring = ax1.plot(Sx \* Tension[0], Sy + 1.5, 'red')[0]

Cylinder, = ax1.plot(cylinder(C\_x[0], C\_y[0], r)[0] + Tension[0], cylinder(C\_x[0], C\_y[0], r)[1], 'green')

# Вторая часть

ax2 = fig.add\_subplot(4, 2, 2)

ax2.set(xlim=[0, 10], ylim=[min(sol[:, 2]), max(sol[:, 2])])

TVx = [T[0]]

TVy = [sol[:, 2][0]]

TV, = ax2.plot(TVx, TVy, '-')

ax2.set\_xlabel('T')

ax2.set\_ylabel('V')

ax3 = fig.add\_subplot(4, 2, 4)

ax3.set(xlim=[0,10], ylim=[min(sol[:, 3]), max(sol[:, 3])])

TOmx = [T[0]]

TOmy = [sol[:, 3][0]]

TOm, = ax3.plot(TOmx, TOmy, '-')

ax3.set\_xlabel('T')

ax3.set\_ylabel('Om')

plt.subplots\_adjust(wspace=0.3, hspace=0.4)

# Анимация

def Animation(i):

Box.set\_data(O1 + Tension[i], O2)

Recess.set\_data(recess(R)[0] + Tension[i], recess(R)[1])

line1.set\_data([Tension[i] + 0.5, Tension[i]], [3, 3])

line2.set\_data([Tension[i] + 3.5, Tension[i] + 4], [3, 3])

Spring.set\_data(Sx \* Tension[i], Sy + 1.5)

Cylinder.set\_data(cylinder(C\_x[i], C\_y[i], r)[0] + Tension[i], cylinder(C\_x[i], C\_y[i], r)[1])

TVx.append(T[i])

TVy.append(sol[:, 2][i])

TOmx.append(T[i])

TOmy.append(sol[:, 3][i])

TV.set\_data(TVx, TVy)

TOm.set\_data(TOmx, TOmy)

if i == 500-1:

TVx.clear()

TVy.clear()

TOmx.clear()

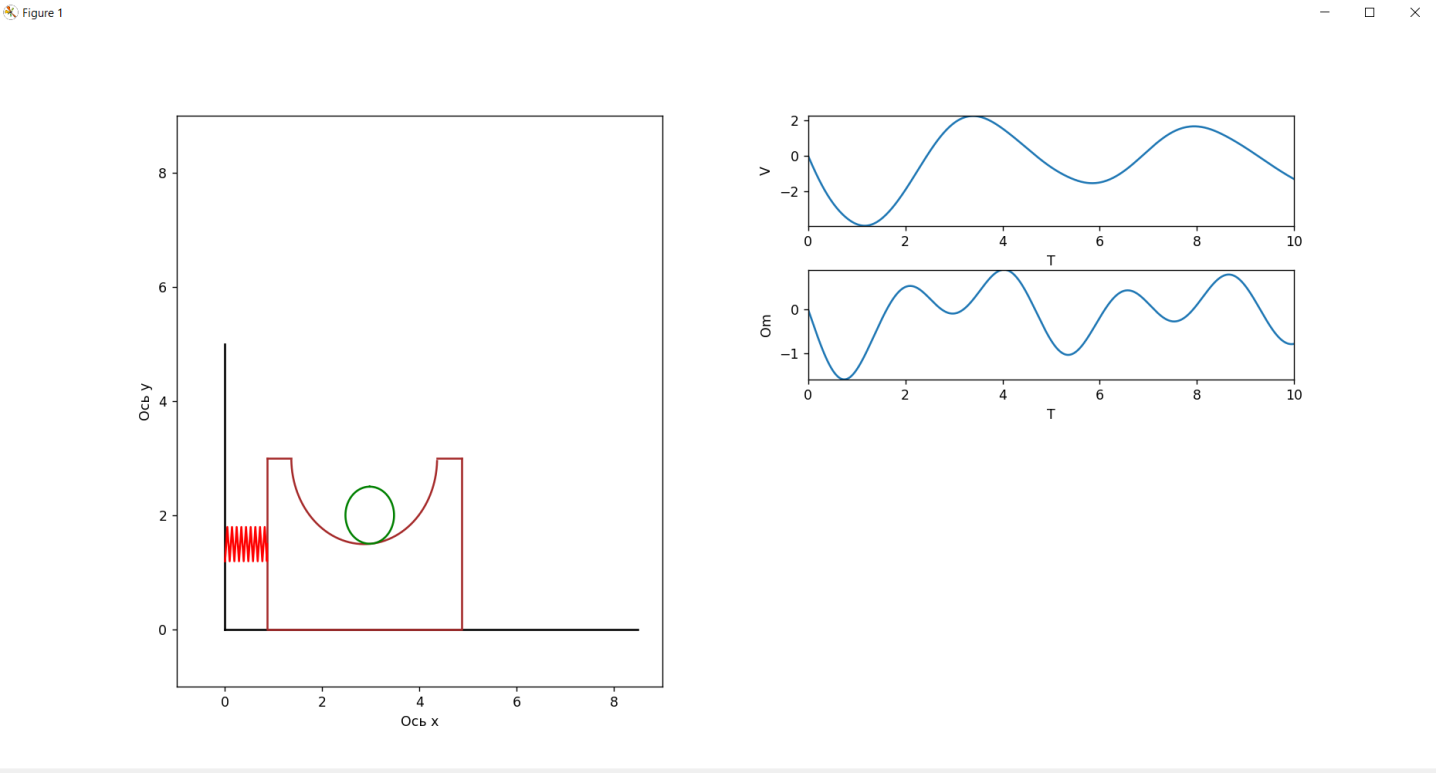
TOmy.clear()

return Box, Recess, line1, line2, Spring, Cylinder, TV, TOm

plot = FuncAnimation(fig, Animation, frames=1000, interval=0.001)

plt.show()

# *Результат работы программы:*

******