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 №4

по курсу «Теоретическая механика»

Ⅲ семестр

«Динамика системы. Расчет уравнения малых колебаний»

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

Москва, 2023

Оглавление

[*Задание:* 3](#_Toc126762468)

[*Механическая система*: 3](#_Toc126762469)

[*Вычисления:* 4](#_Toc126762470)

[*Текст программы:* 5](#_Toc126762471)

[*Результат работы программы:* 8](#_Toc126762472)

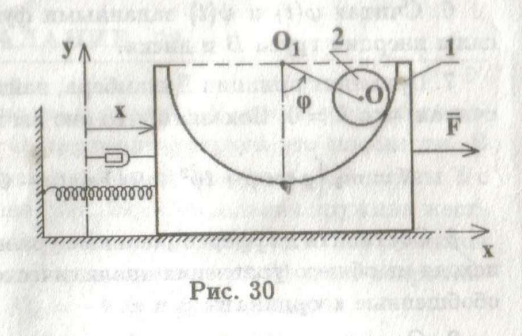
[*Вывод:* 8](#_Toc126762473)

Вариант 30

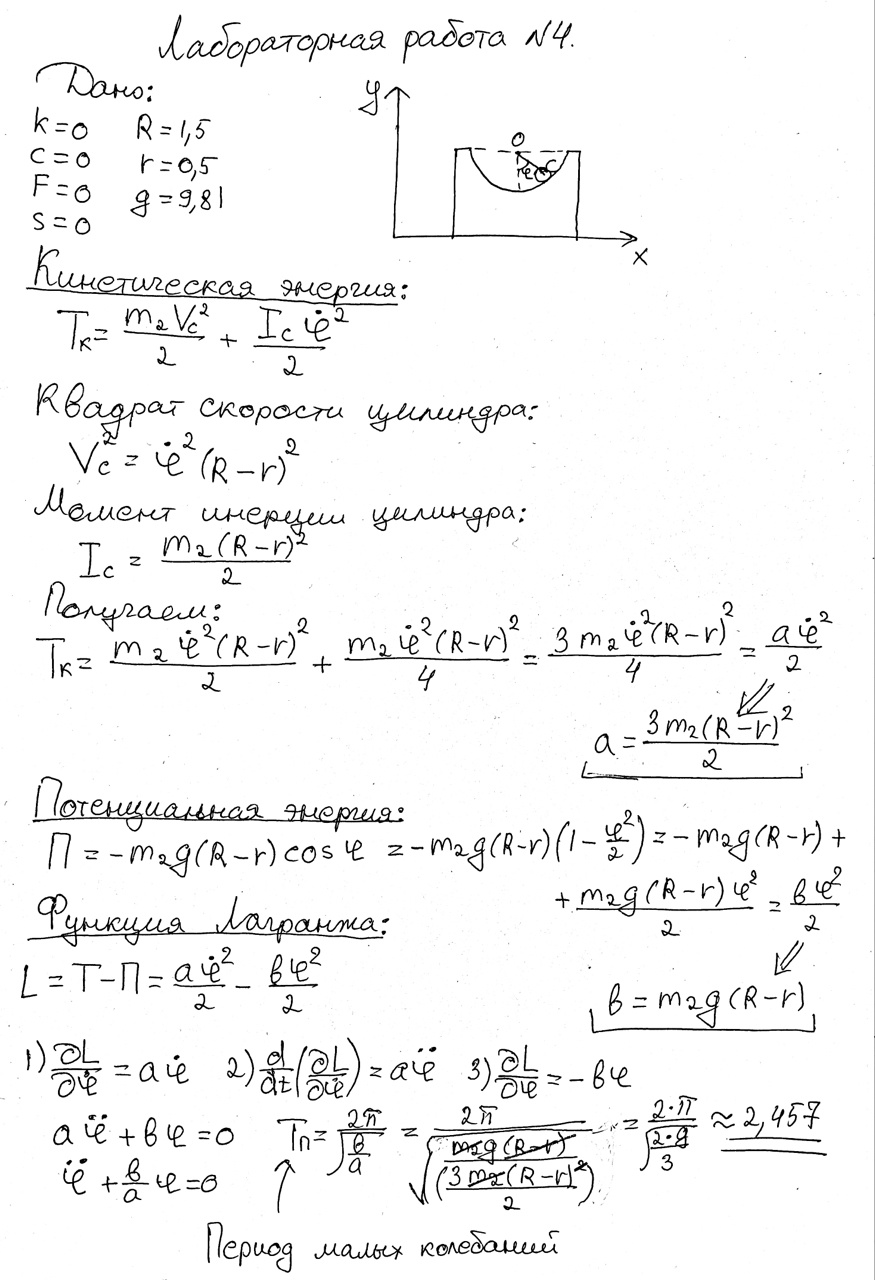
# *Задание:*

Полагая, что с=0, k=0, =0, s=0 получить уравнение малых колебаний цилиндра в окрестности его нижнего ( положения. Найти период малых колебаний.

# *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 цилиндра

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 = [4 + 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, fOm):

y1, y2 = y

dydt = [y2, fOm(y1, y2)]

return dydt

# Задаем параметры системы

m2 = 1 # Масса цилиндра

g = 9.81 # Ускорение свободного падения

r = 0.5 # Радиус цилиндра

R = 1.5 # Радиус внутренней выемки

# Коэффициенты

k = 0

c = 0

F0 = 0

# Определение t как символа

t = sp.Symbol('t')

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

s = 0

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

V = 0

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

# Строим уравнение Лагранжа

# Вычисляем квадрат скорости цилиндра

Vc2 = (om \* (R - r))\*\*2

# Момент инерции цилиндра

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

# Кинетическая энергия цилиндра

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

# Кинетическая энергия общая

TT = TCy

# Находим потенциальную энергию

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

# Потенциальная энергия общая

Pi = Pi1

# Функция Лагранжа

L = (TT - Pi).subs(sp.diff(phi, t), om)

print(L)

# Уравнение Лагранжа

ur1 = (sp.diff(sp.diff(L, om), t) - sp.diff(L, phi)).simplify()

# Выделяем вторые производные (dV/dt и dom/dt)

a22 = ur1.coeff(sp.diff(om, t), 1)

b2 = -ur1.coeff(sp.diff(om, t), 0).subs(sp.diff(phi, t), om)

domdt = b2 / a22

# Создаем систему дифференциальных уравнений

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

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

y0 = [0.1, 0] # Вектор начальных значений: phi(0), om(0)

# Решение системы дифференциальных уравнений

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

# sol - Наше решение

# sol[:,0] - phi

# sol[:,1] - dphi/dt

# Задание пружины

n = 15 # Количество витков

k = 1 / (n - 2)

width = 0.4 # 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

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

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

C\_x = Cx\_def(sol[:, 1]) + 2

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 + 2, O2, 'brown')[0]

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

line1 = ax1.plot([0.5 + 2, 2], [3, 3], 'brown')[0]

line2 = ax1.plot([3.5 + 2, 4 + 2], [3, 3], 'brown')[0]

Spring = ax1.plot(Sx \* 2, Sy + 1.5, 'red')[0]

Cylinder, = ax1.plot(cylinder(C\_x[0], C\_y[0], r)[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[:, 0]), max(sol[:, 0])])

TPhix = [T[0]]

TPhiy = [sol[:, 0][0]]

TPhi, = ax2.plot(TPhix, TPhiy, '-')

ax2.set\_xlabel('T')

ax2.set\_ylabel('Phi')

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

ax3.set(xlim=[0,10], ylim=[min(sol[:, 1]), max(sol[:, 1])])

TOmx = [T[0]]

TOmy = [sol[:, 1][0]]

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

ax3.set\_xlabel('T')

ax3.set\_ylabel('Om')

plt.subplots\_adjust(wspace=0.3, hspace=0.6)

# Анимация

def Animation(i):

Box.set\_data(O1 + 2, O2)

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

line1.set\_data([0.5 + 2, 0 + 2], [3, 3])

line2.set\_data([3.5 + 2, 4 + 2], [3, 3])

Spring.set\_data(Sx \* 2, Sy + 1.5)

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

TPhix.append(T[i])

TPhiy.append(sol[:, 0][i])

TOmx.append(T[i])

TOmy.append(sol[:, 1][i])

TPhi.set\_data(TPhix, TPhiy)

TOm.set\_data(TOmx, TOmy)

if i == 500-1:

TPhix.clear()

TPhiy.clear()

TOmx.clear()

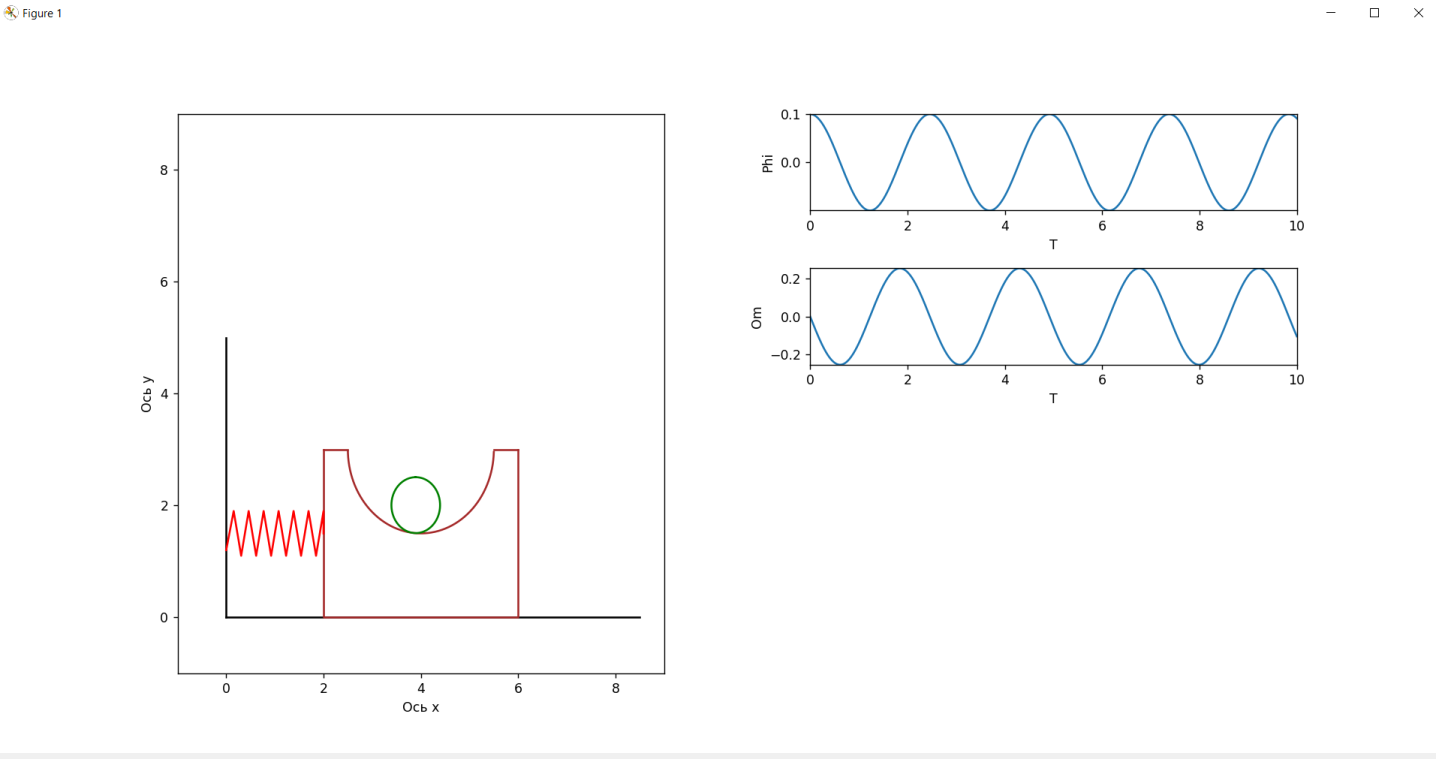
TOmy.clear()

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

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

plt.show()

# *Результат работы программы:*

******

# *Вывод:*

Моделирование малых колебаний прошло в соответствии с аналитическими выкладками, периоды малых колебаний в моделировании и аналитических выкладках с очень высокой точностью совпадают.