ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«ДИНАМИКА СИСТЕМЫ»**

**ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»**

**ВАРИАНТ ЗАДАНИЯ №1**

Выполнил(а) студент группы М8О-210Б-22

Андреев Иван Владимирович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Зав. каф. 802, Бардин Б.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

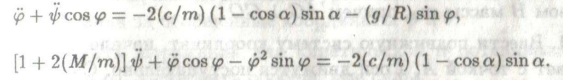
подпись, дата

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

**Задание:** проинтегрировать систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python. Построить анимацию движения системы.

**Задание системы 1 варианта формулируется следующим образом:**

****

**Код программы**

from matplotlib import pyplot as plt

import numpy as np

from math import sqrt

from mpl\_toolkits.mplot3d import Axes3D

from matplotlib.animation import FuncAnimation

from scipy.integrate import odeint

# CONST VARIABLES

DOT\_COLOR = "black"

SPRING\_COLOR = "green"

CIRCLE\_COLOR = "blue"

CENTRE\_COLOR = "white"

CENTRE\_OUTLINE\_COLOR = "blue"

LINE\_COLOR = "black"

A\_POINT\_SIZE = 4

B\_POINT\_SIZE = 8

phi0 = np.pi / 3

psi0 = 0

dphi0 = 0

dpsi0 = 0

y0 = [phi0, psi0, dphi0, dpsi0]

def odesys(y, t, M, m, c, R, g):

dy = np.zeros(4)

dy[0] = y[2]

dy[1] = y[3]

alpha = (y[0]+y[1])/2

a11 = 1

a12 = np.cos(y[0])

b1 = -2\*(c/m)\*(1-np.cos(alpha))\*np.sin(alpha)-(g/R)\*np.sin(y[0])

a21 = np.cos(y[0])

a22 = 1 + 2\*(M/m)

b2 = -2\*(c/m)\*(1-np.cos(alpha))\*np.sin(alpha) + np.sin(y[0])\*y[3]\*\*2

dy[2] = (b1\*a22 - b2\*a12)/(a11\*a22 - a12\*a21)

dy[3] = (b2\*a11 - b1\*a21)/(a11\*a22 - a12\*a21)

return dy

M = 1 # колесо

m = 2 # маленький шарик

c = 40 # жесткость

R1 = 5 # большой радиус

R2 = 4 # маленький радиус

g = 9.81 # свободное падение

T = 1000

t\_fin = 25

t = np.linspace(0, t\_fin, T)

Y = odeint(odesys, y0, t, (M, m, c, R1, g))

phi = Y[:, 0]

psi = Y[:, 1]

dphi = Y[:, 2]

dpsi = Y[:, 3]

vel = R1 \* psi

angle = np.linspace(0, 6.28, T) # 2 Pi

CHART\_SIZE = 20

# Координаты

OX1 = -15

OX2 = 15

OY = -R1

n = 10

CENTRE\_X = 0

CENTRE\_Y = 0

def rot2D(X, Y, Phi):

RotX = X \* np.cos(Phi) - Y \* np.sin(Phi)

RotY = X \* np.sin(Phi) + Y \* np.cos(Phi)

return RotX, RotY

def animation(i):

Centre.set\_data(CENTRE\_X + vel[i], CENTRE\_Y)

C1.set\_data(CX1 + vel[i], CY1)

C2.set\_data(CX2 + vel[i], CY2)

A.set\_data(Ax[i], Ay[i])

B.set\_data(Bx[i], By[i])

horizont = Bx[i] - Ax[i]

vertical = By[i] - Ay[i]

l = np.sqrt(horizont \*\* 2 + vertical \*\* 2)

g = np.pi + np.arctan2(vertical, horizont)

Rx, Ry = rot2D(SpringX \* l, SpringY, g)

Spring.set\_data(Rx + Bx[i], Ry + By[i])

return [C1, C2, Centre, Spring, A, B]

# Инициализируем диаграмму

fig = plt.figure(figsize=[5, 5])

ax = fig.add\_subplot(1, 1, 1)

ax.set(xlim=[-CHART\_SIZE, CHART\_SIZE], ylim=[-CHART\_SIZE, CHART\_SIZE])

#t = np.linspace(0, 10, T)

#vel = R1 \* psi

#angle = np.linspace(0, 6.28, T) # 2 Pi

# Уравнения движения для окружностей

CX1 = R1 \* np.sin(angle) + CENTRE\_X

CY1 = R1 \* np.cos(angle) + CENTRE\_Y

CX2 = R2 \* np.sin(angle) + CENTRE\_X

CY2 = R2 \* np.cos(angle) + CENTRE\_Y

BR = (R1 - R2) / 2 + R2

# Уравнения движения точки

Ax = R1 \* np.sin(psi) + vel + CENTRE\_X

Ay = R1 \* np.cos(psi) + CENTRE\_Y

Bx = BR \* np.sin(phi) + vel + CENTRE\_X

By = -BR \* np.cos(phi) + CENTRE\_Y

b = 1 / (n - 2)

sh = 0.5 #

SpringX = np.zeros(n)

SpringY = np.zeros(n)

SpringX[0] = 0

SpringX[n - 1] = 1

SpringY[0] = 0

SpringY[n - 1] = 0

for i in range(n - 2):

SpringX[i + 1] = b \* (i + 1) - b / 2

SpringY[i + 1] = sh \* (-1) \*\* i

horizont = Bx[i] - Ax[i]

vertical = By[i] - Ay[i]

l = np.sqrt(horizont \*\* 2 + vertical \*\* 2)

g = np.pi + np.arctan2(vertical, horizont)

Rx, Ry = rot2D(SpringX \* l, SpringY, g)

# Инициализация объекта

Centre = ax.plot(CENTRE\_X + vel[0], CENTRE\_Y, CENTRE\_COLOR, marker='o', ms=10, mec=CENTRE\_OUTLINE\_COLOR)[0]

Line = ax.plot([OX1, OX2], [OY, OY], LINE\_COLOR)

C1 = ax.plot(CX1 + vel[0], CY1, color=CIRCLE\_COLOR)[0]

C2 = ax.plot(CX2 + vel[0], CY2, color=CIRCLE\_COLOR)[0]

Spring = ax.plot(Rx + Bx[0], Ry + By[0], SPRING\_COLOR)[0]

A = ax.plot(Ax[0], Ay[0], DOT\_COLOR, marker='o', ms=A\_POINT\_SIZE)[0]

B = ax.plot(Bx[0], Ay[0], DOT\_COLOR, marker='o', ms=B\_POINT\_SIZE)[0]

a = FuncAnimation(fig, animation, frames=T, interval=10)

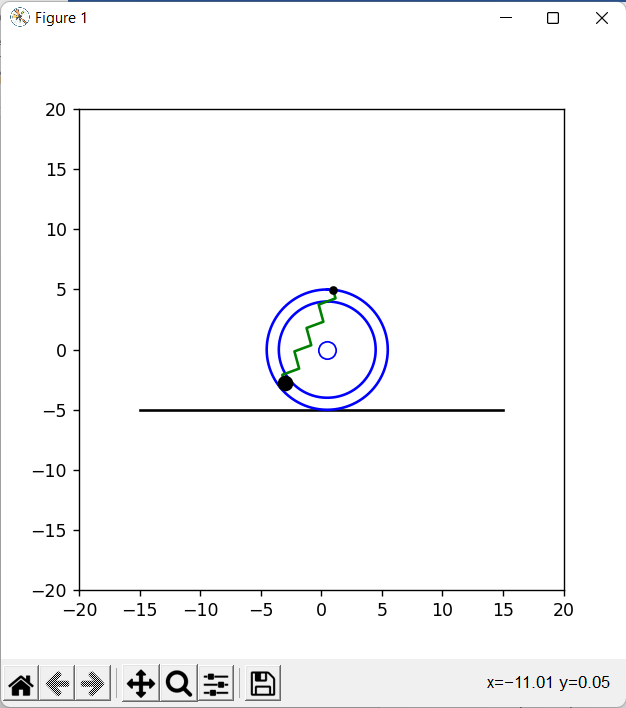
plt.show()

**Примеры работы:**

Приведу несколько примеров работы моей программы.

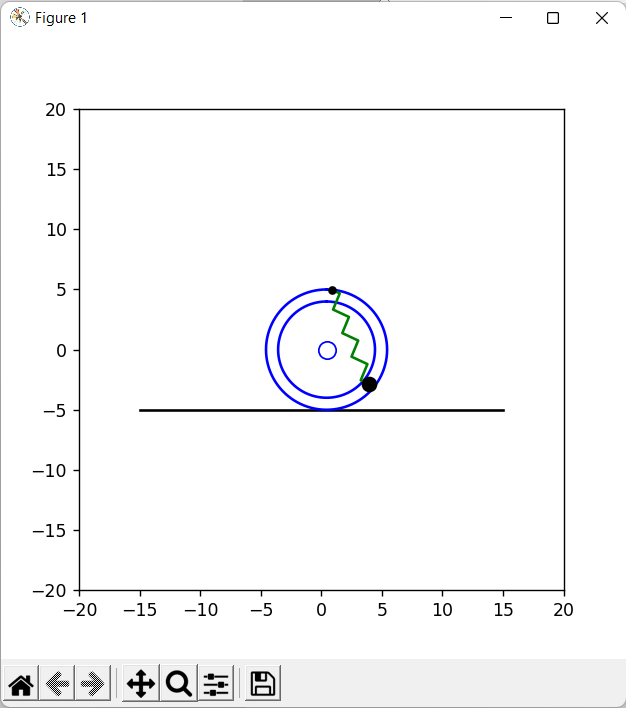
1. m = 2; M = 1; R = 5; c = 40; g = 9.81;

y0 = [Pi/3 0 0 0];

****

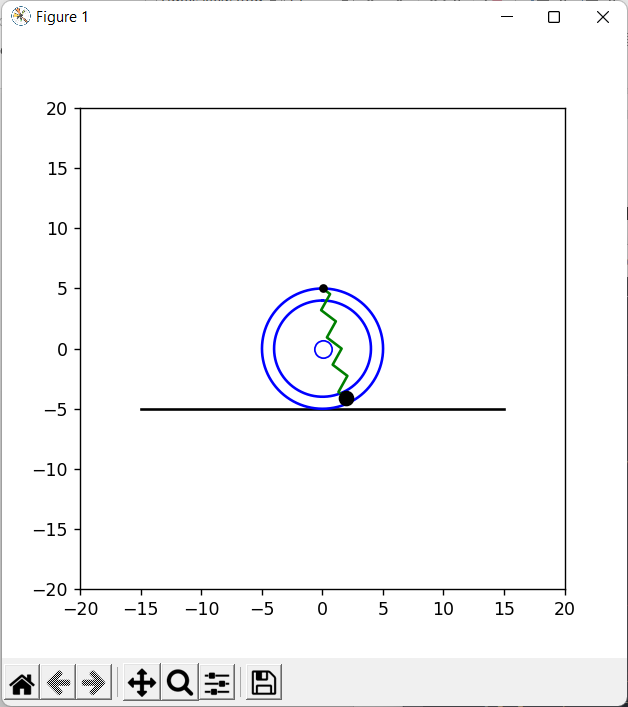
1. m = 200; M = 1; R = 5; c = 0; g = 9.81;

y0 = [Pi/3 0 0 0];



3. m = 2; M = 100; R = 5; c = 0; g = 9.81;

y0 = [Pi/3 0 0 0];



**Вывод**

Я успешно завершил лабораторную работу по теоретической механике с использованием языка программирования Python и библиотек matplotlib и numpy. В ходе работы я создал схематическую анимацию движения двух стержней и пружины, а также решение системы дифференциальных уравнений, описывающих данное движение.

Эта лабораторная работа позволила мне освоить работу с 2D-анимацией в библиотеке matplotlib и создать основу для следующих исследований. Моя программа использует реальные законы движения, что позволяет наблюдать за поведением системы в условиях, приближенных к реальным. Полученные навыки и результаты этой работы будут полезны для дальнейших экспериментов и исследований.