ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ СИСТЕМЫ»**

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

**ВАРИАНТ ЗАДАНИЯ №21**

Выполнил(а) студент группы М8О-207Б-22

Кострюков Евгений Сергеевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

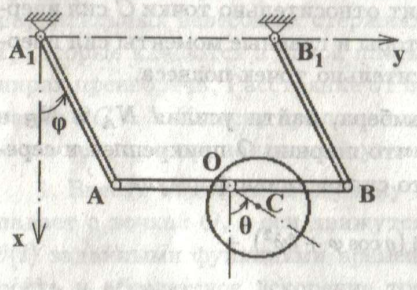
подпись, дата

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

Москва, 2023

*Задание:* построить анимацию движения системы с помощью Python.

*1) Система:*



φ(t), θ(t) — заданные функции времени.

*2) Код программы:*

import numpy as np

import math

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

Steps = 1001

t\_fin = 20

t = np.linspace(0, t\_fin, Steps) # Равномерно распределяем Steps элементов на участке (0; t\_fin)

# Характеристики различных предметов системы

phi\_0 = math.pi / 2

A1A = B1B = 0.75 # длина параллельных стержней

AB = 1 # Длина прикреплённого стержня

OC = 0.125 # Расстояние от шарнира O до центра масс диска

r = 0,25 # Радиус диска

# Случайно-подобранные функции

phi = np.cos(2 \* t) - np.sin(2 \* t) - 3.15

tetta = phi\_0 + 3 \* np.cos(2 \* t) - 3 \* np.sin(2 \* t)

# Координаты подвижных частей стержней

# Конец стержня A1A

A\_X = A1A \* np.sin(phi)

A\_Y = A1A \* np.cos(phi)

# Конец стержня B1B

B\_X = B1B \* np.sin(phi) + AB

B\_Y = B1B \* np.cos(phi)

# Центр масс диска

C\_X = A\_X + AB / 2 + OC \* np.sin(tetta)

C\_Y = A\_Y + OC \* np.cos(tetta)

fig = plt.figure(figsize=[10, 9]) # Создаём окно для отрисовки

ax = fig.add\_subplot(1, 1, 1) # Добавляем ячейку (окно) для отрисовки

ax.axis('equal') # Делаем единичные отрезки осей равными

ax.set(xlim=[-3, 4], ylim=[-1.5, 1.5]) # Пределы по осям

# Координаты поверхности, на которой закрепелны стержни

X\_Ground = [-0.5, 0, 1.5]

Y\_Ground = [0.1, 0.1, 0.1]

# plot() рисует прямые линии, соединяя указанные точки

ax.plot(X\_Ground, Y\_Ground, color='black', linewidth=3) # Отрисовка поверхности

# Опоры, на которых вращаются стержни

Drawed\_Q1A1 = ax.plot([-0.1, 0], [0.1, 0], color='black')

Drawed\_Q2A1 = ax.plot([0.1, 0], [0.1, 0], color='black')

Drawed\_E1A1 = ax.plot([AB - 0.1, AB], [0.1, 0], color='black')

Drawed\_E2A1 = ax.plot([AB + 0.1, AB], [0.1, 0], color='black')

# Параметры, куда помещаем объект отрисовки чего-либо в начальный момент времени

# Стержени

Drawed\_A1A = ax.plot([0, A\_X[0]], [0, A\_Y[0]], color='green')[0]

Drawed\_B1B = ax.plot([1, B\_X[0]], [0, B\_Y[0]], color='green')[0]

Drawed\_AB = ax.plot([A\_X[0], B\_X[0]], [A\_Y[0], B\_Y[0]], color='blue')[0]

# Радиус-вектор

Drawed\_OC = ax.plot([A\_X[0] + AB / 2, C\_X[0]], [A\_Y[0], C\_Y[0]], color='black')[0]

# Отрисовываем точки для стержней

Point\_A1 = ax.plot(0, 0, marker='o', color='black')

Point\_B1 = ax.plot(1, 0, marker='o', color='black')

Point\_A = ax.plot(A\_X[0], A\_Y[0], marker='o', color='green')[0]

Point\_B = ax.plot(B\_X[0], B\_Y[0], marker='o', color='green')[0]

Point\_O = ax.plot([A\_X[0] + AB / 2], A\_Y[0], marker='o', color='black')[0]

Point\_C = ax.plot(C\_X[0], C\_Y[0], marker='o', markersize=70, markerfacecolor='none', markeredgewidth=2, color='red')[0]

# Обновление текущего кадра

def anima(i):

# Изменяем нулевые элементы на i-ые:

Drawed\_A1A.set\_data([0, A\_X[i]], [0, A\_Y[i]])

Drawed\_B1B.set\_data([1, B\_X[i]], [0, B\_Y[i]])

Drawed\_AB.set\_data([A\_X[i], B\_X[i]], [A\_Y[i], B\_Y[i]])

Point\_A.set\_data(A\_X[i], A\_Y[i])

Point\_B.set\_data(B\_X[i], B\_Y[i])

Point\_O.set\_data([A\_X[i] + AB / 2], A\_Y[i])

Point\_C.set\_data(C\_X[i], C\_Y[i])

Drawed\_OC.set\_data([A\_X[i] + AB / 2, C\_X[i]], [A\_Y[i], C\_Y[i]])

return [Point\_A, Point\_B, Point\_O, Point\_C, Drawed\_A1A, Drawed\_B1B, Drawed\_AB, Drawed\_OC]

anim = FuncAnimation(fig, anima, frames=len(t), interval=10, repeat=False) # Создаём анимацию

plt.show() # Отрисовываем

*3) Пояснение к коду:*

К*од создает анимацию движения механической системы, состоящей из вращающихся стержней и диска. Основные компоненты кода:*

*1. Импортируем библиотеки, необходимые для работы с числами, математическими операциями, графикой и анимацией.*

*2. Настраиваем параметры анимации:*

*Создаём параметры для временной шкалы. `Steps` определяет количество шагов анимации, а `t\_fin` задает общее время анимации. `t` - это массив времени с равномерно распределенными значениями.*

*3. Определяем характеристики различных компонентов системы, такие как начальный угол `phi\_0`, длины стержней `A1A` и `B1B`, длина прикрепленного стержня `AB`, расстояние от шарнира `O` до центра масс диска `OC`.*

*4. Случайным образом задаём функции для углов стержней и диска от времени.*

*5. Рассчитываем координаты концов стержней (подвижных частей стержней) и центра масс диска в зависимости от времени.*

*6. Создаём графическое окно с заданными размерами и осями.*

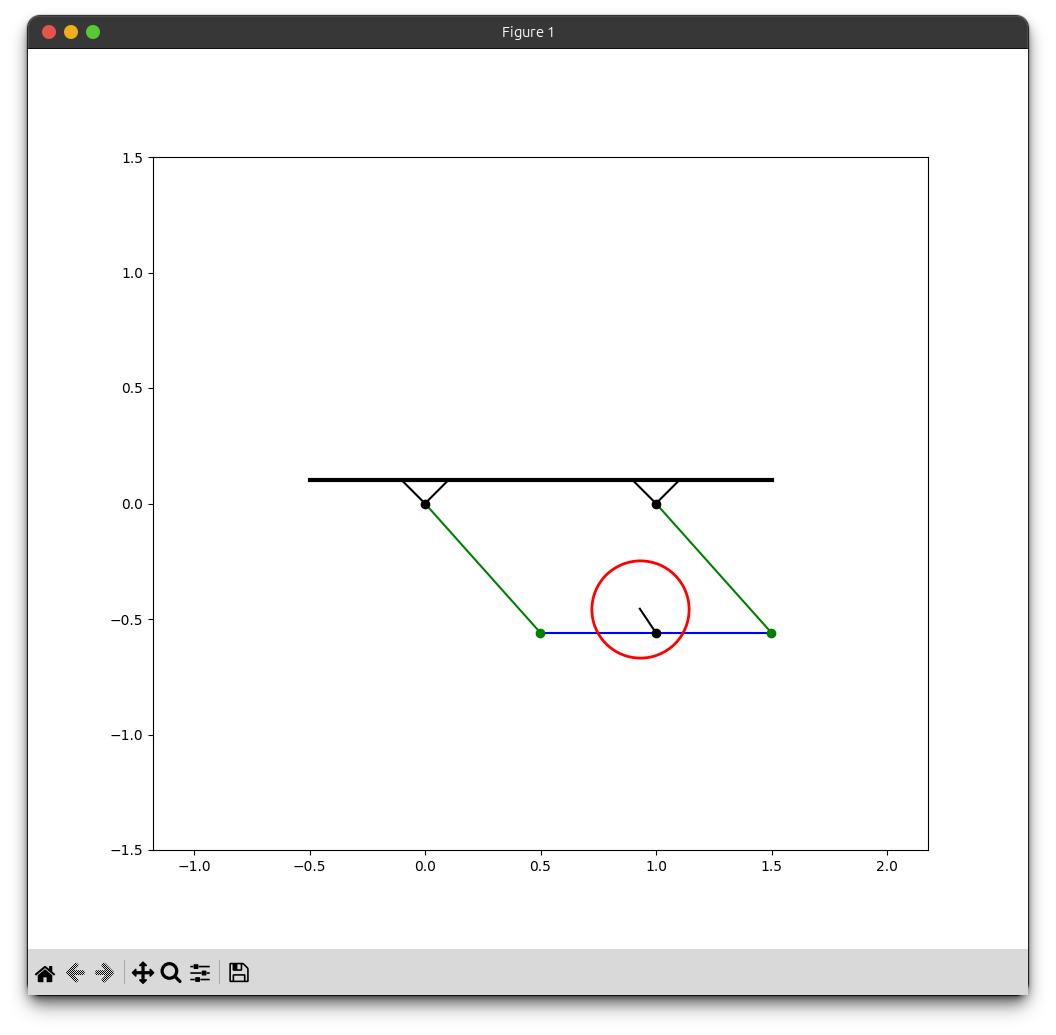
*7. Отрисовываем поверхность и опоры для стержней.*

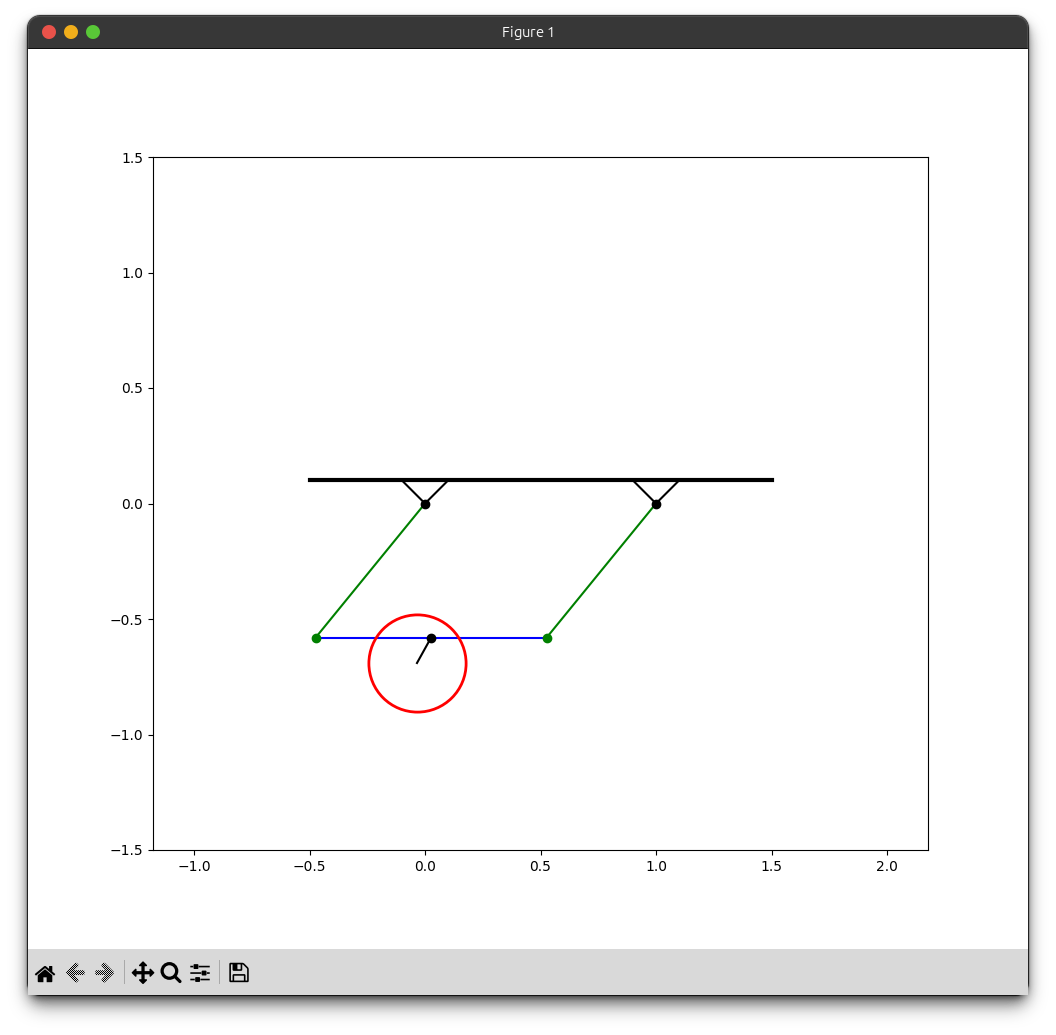
*8. Отрисовываем подвижные части системы и точки, представляющие концы стержней и центр масс диска.*

*9. Создаём функцию для обновления кадра, и затем создаём анимацию, используя библиотеку Matplotlib.*

*10. Запускаем отображение созданной анимации. Каждый кадр анимации представляет собой обновленное положение частей системы в зависимости от времени.*

*4) Результат работы программы:*

**

**

*5) Вывод:*

Лабораторная работа успешно сделана. В ходе работы над ней мы научились создавать анимацию, представляющую изменение положения элементов системы с течением времени, что обеспечивает интуитивное представление о динамике системы и может быть полезным для анализа её поведения.