ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ СИСТЕМЫ»**

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

**ВАРИАНТ ЗАДАНИЯ №15**

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

Гусейнова Т.М.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

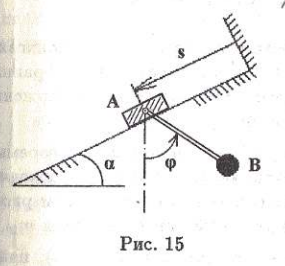
Зав. каф. 802, Бардин Б.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

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

Москва, 2023

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



**Текст программы**

# Вариант № 15

import math as mh

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

def rotate(x, y, theta):

xr = mh.cos(theta) \* x - mh.sin(theta) \* y

yr = (mh.sin(theta)) \* x + np.cos(theta) \* y

return xr, yr

def Film(i):

Point\_A.set\_data(X\_A[i], Y\_A[i])

Point\_B.set\_data(X\_B[i], Y\_B[i])

Box.set\_data(X\_A[i] + X\_Box, Y\_A[i] + Y\_Box)

Line\_AB.set\_data([X\_A[i], X\_B[i]], [Y\_A[i], Y\_B[i]])

return [Point\_A, Point\_B, Box, Line\_AB]

# 400 шагов за 10 секунд

steps = 400

t = np.linspace(10, 0, steps)

S = 5

phi = np.sin(t)

s = t

x = t

#

fig = plt.figure(figsize=[20, 8])

ax = fig.add\_subplot(1, 1, 1) #одна картинка

ax.axis('equal')#равные единичные отрезки

ax.set(xlim=[-3, 10], ylim=[-3, 10])

BoxX = 4

BoxY = 2

bx = BoxX / 2

by = BoxY / 2

l = 3

Pi = np.pi

alpha = Pi / 6

alphay = (55 \* Pi) / 180

tg = np.tan(alpha)

# Платформы

X\_Ground = [10 / np.tan(alphay), 10, -6, 10]

Y\_Ground = [10, 8 \* tg, -8 \* tg, -8 \* tg]

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

#

# Брусок

x1, y1 = rotate(-bx, by, alpha)

x2, y2 = rotate(bx, by, alpha)

x3, y3 = rotate(bx, -by, alpha)

x4, y4 = rotate(-bx, -by, alpha)

X\_Box = np.array([x1, x2, x3, x4, x1])

Y\_Box = np.array([y1, y2, y3, y4, y1])

# Точки А и В

X\_A = BoxX / 2 + x - S

Y\_A = X\_A \* tg

X\_B = X\_A + l \* np.sin(phi)

Y\_B = Y\_A - l \* np.cos(phi)

Point\_A = ax.plot(X\_A[0], Y\_A[0], marker='o')[0]

Point\_B = ax.plot(X\_B[0], Y\_B[0], marker='o', markersize=20, color='black')[0]

Box = ax.plot(X\_A[0] + X\_Box, Y\_A[0] + Y\_Box, color='black', linewidth='2')[0]

Line\_AB = ax.plot([X\_A[0], X\_B[0]], [Y\_A[0], Y\_B[0]], color='black', linewidth='2')[0]

anima = FuncAnimation(fig, Film, frames=steps, interval=10)

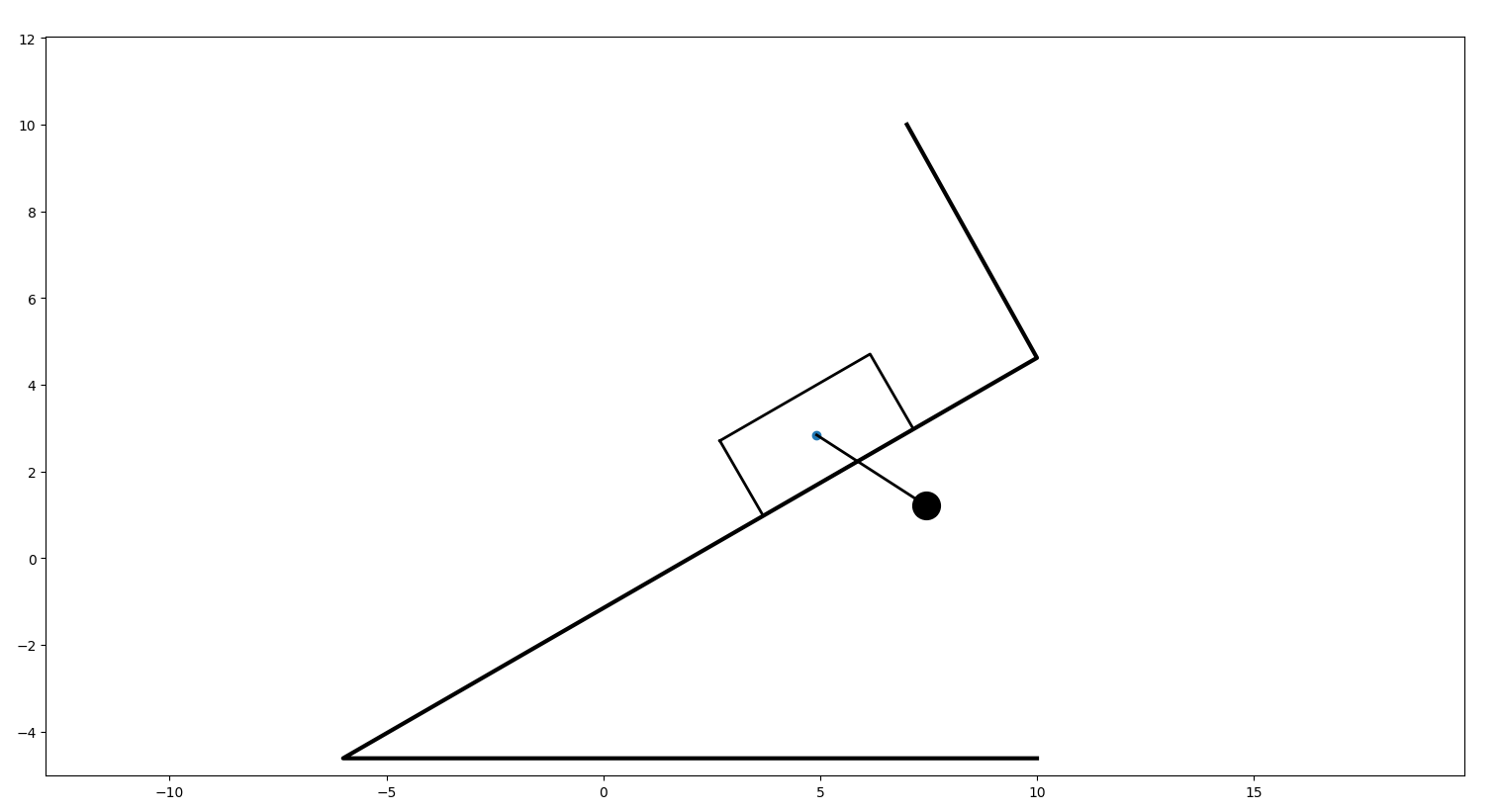
plt.show()

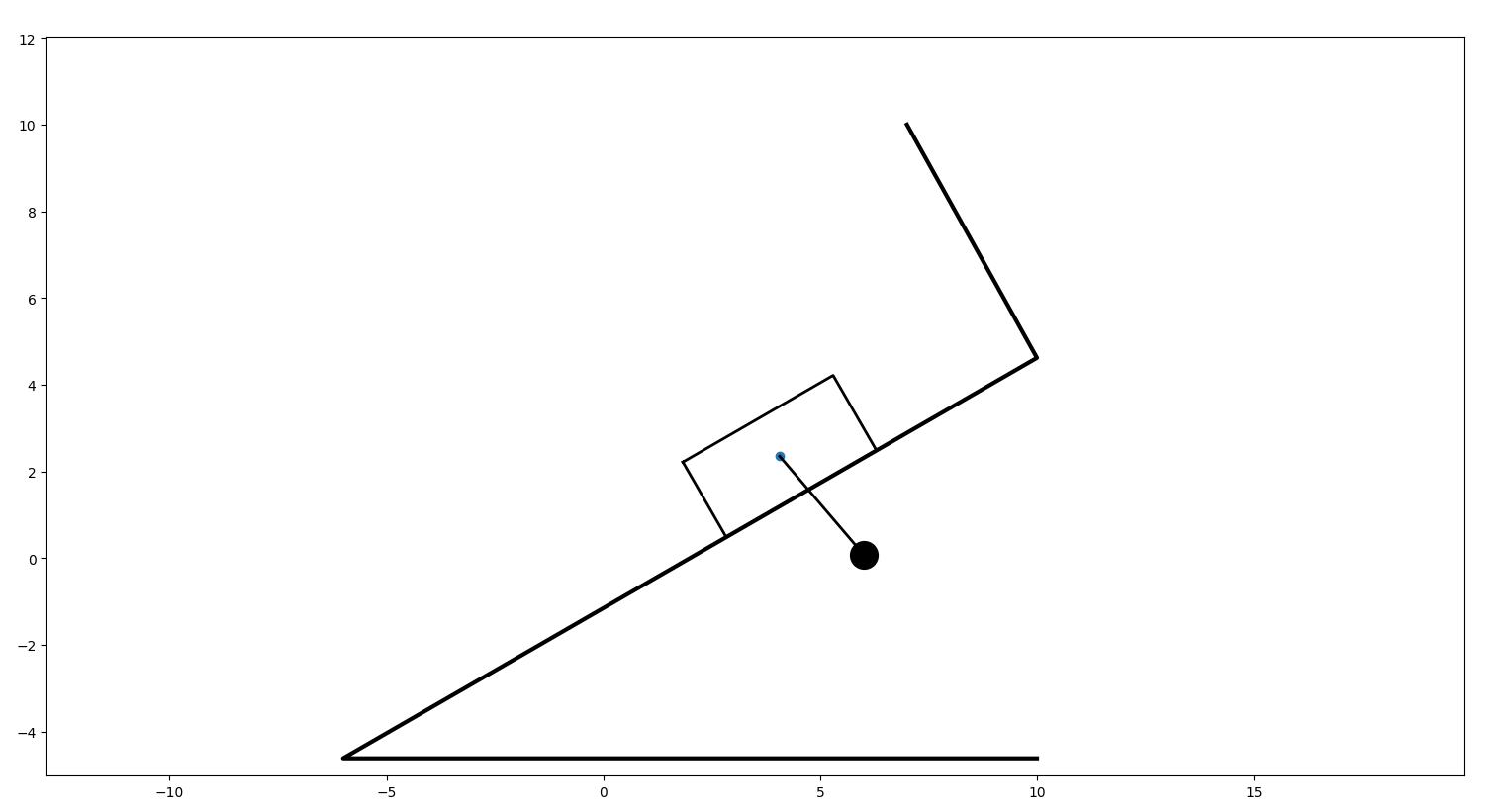
**Описание**

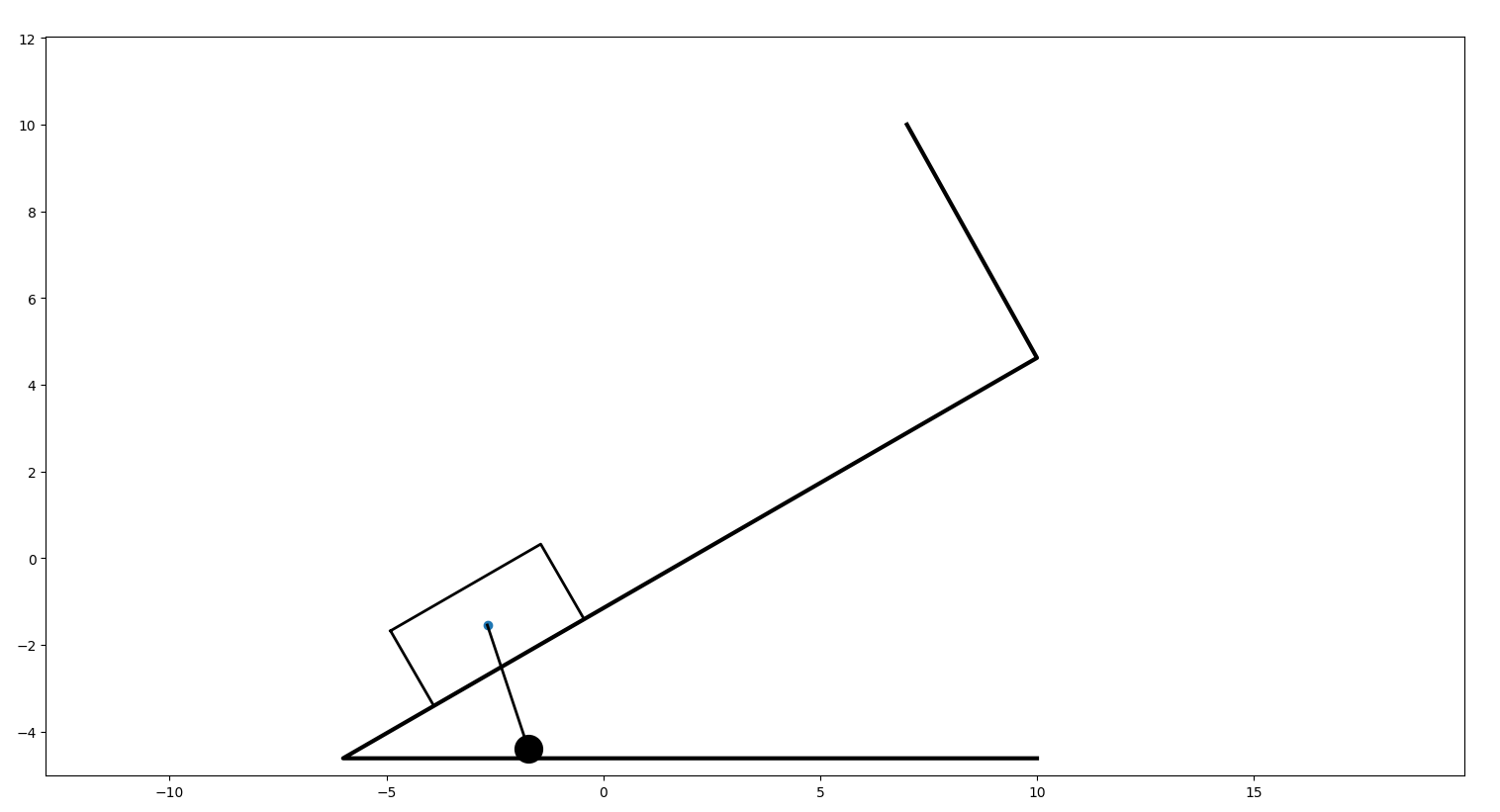
В данной программе происходит анимация движения точек и окружностей на графике.   
  
В функции rotate определяем координаты после изменения угла отклонения, посредством умножения на матрицу поворота. Функция Film — функция обновления кадров.  
  
Затем задаются значения переменных steps, t, которые определяют количество кадров анимации, равномерное распределение 400 кадров за 10 секнуд.

Задаем размеры бруска, размеры и положение наклонной плоскости с помощью переменных BoxX = 4, BoxY = 2, X\_Ground, Y\_Ground.  
Далее используется символическая математика для определения координат и скорости точек A и B.   
  
Функция anima обновляет координаты точек и окружностей на каждом кадре анимации.  
  
И, наконец, создается анимация функцией FuncAnimation, которая отображает анимацию на графике.

**Результат работы программы**







**Вывод**

Я училась строить анимацию движения системы с помощью Python.