ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ ТОЧКИ»**

**ПО ДИСЦИПЛИНЕ «Основы Компьютерного Моделирования Математических Систем»**

**ВАРИАНТ ЗАДАНИЯ № 2**

Выполнил(а) студент группы М8О-209Б-23

Борисов Д.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Ст. преп. каф. 802 Волков Е.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

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

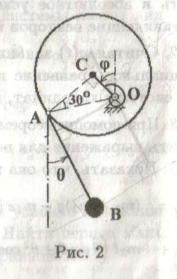
Москва, 2024

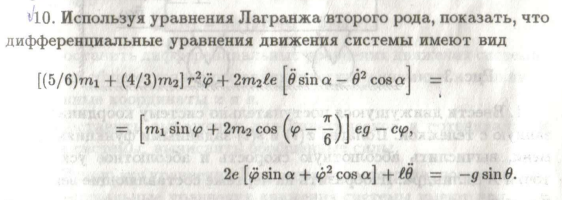
**Вариант № 2**

**Задание:**

построить анимацию движения системы, а также графики законов движения системы (поэкспериментировать с параметрами системы). Исследовать на устойчивость. Показать правильность работы своей механической системы.

**Механическая система:**



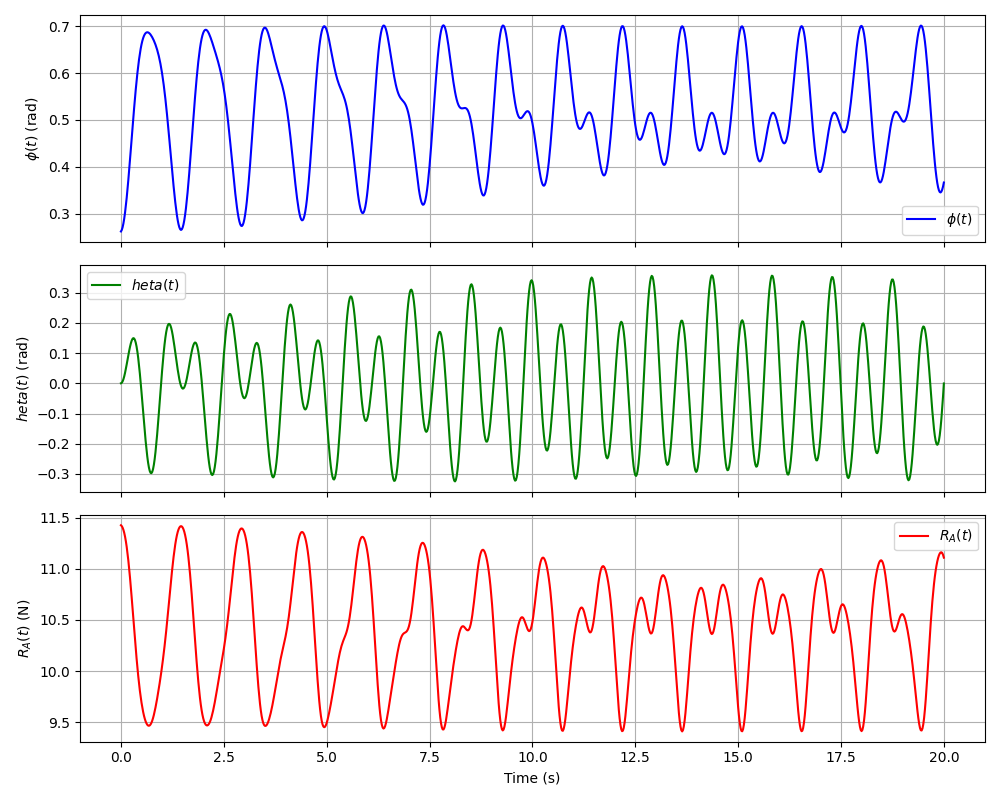
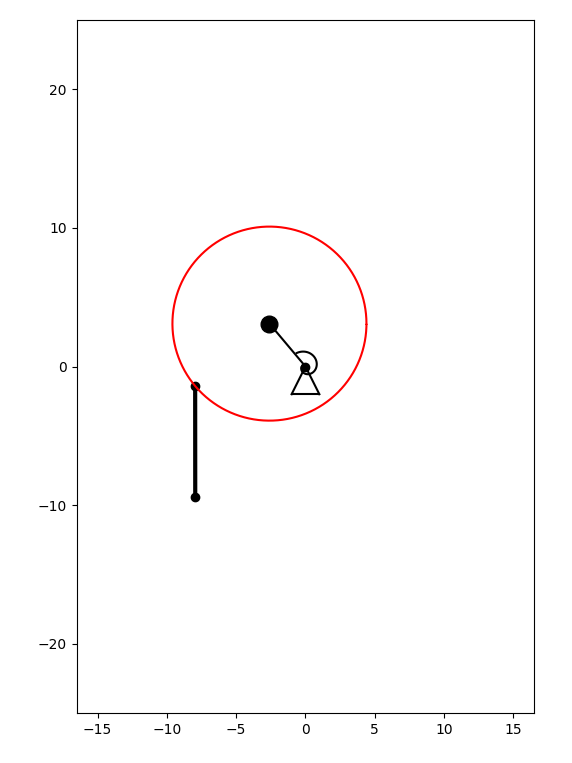


**Текст программы**

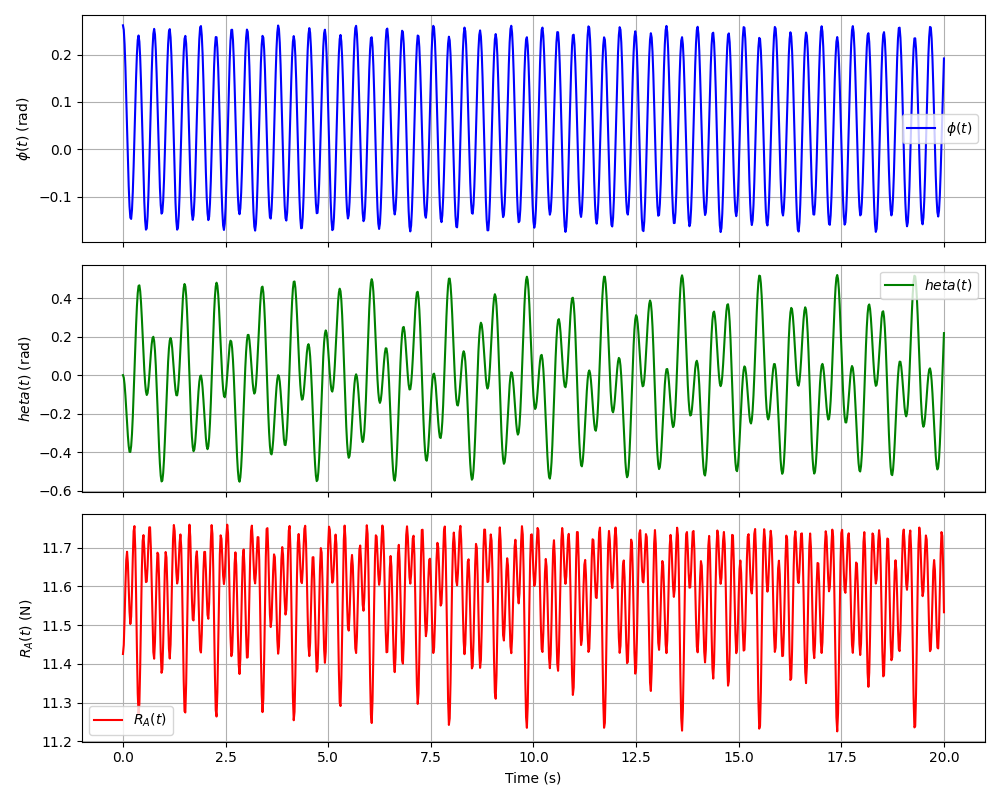
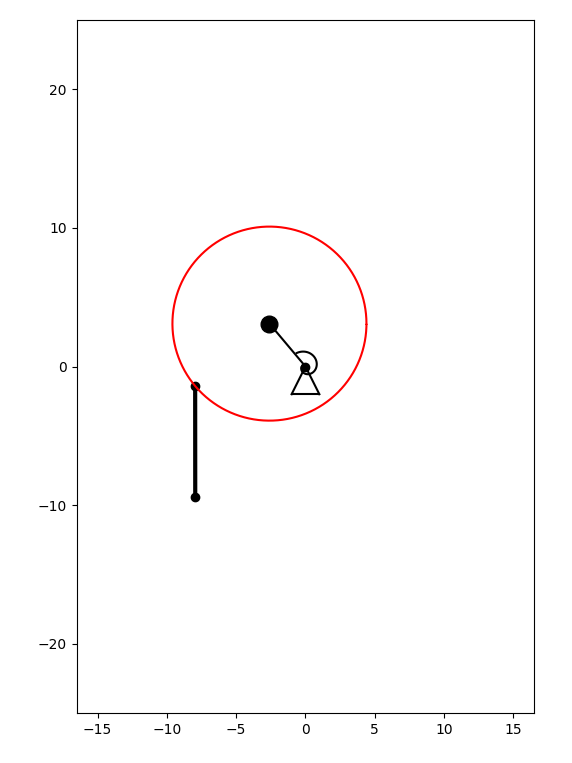
import numpy as np  
import math  
import matplotlib.pyplot as plt  
from matplotlib.animation import FuncAnimation  
from scipy.integrate import odeint  
  
*# Составление системы дифференциальных уравнений*def mechanical\_system\_dynamics(y, t, m1, m2, c, l, e, alpha, g):  
 dy = np.zeros(4)  
 dy[0] = y[2]  
 dy[1] = y[3]  
  
 a11 = ((5 / 6) \* m1 + (4 / 3) \* m2) \* R \* R  
 a12 = 2 \* m2 \* l \* e \* math.sin(alpha)  
 a21 = 2 \* e \* math.sin(alpha)  
 a22 = l  
 b1 = (m1 \* np.sin(y[0]) + 2 \* m2 \* np.cos(y[0] - math.pi / 6)) \* e \* g - c \* y[0] + 2 \* m2 \* l \* e \* y[3] \*\* 2 \* math.cos(alpha)  
 b2 = -g \* np.sin(y[1]) - 2 \* e \* y[2] \*\* 2 \* math.cos(alpha)  
 dy[2] = (b1 \* a22 - b2 \* a12) / (a11 \* a22 - a12 \* a21)  
 dy[3] = (b2 \* a11 - b1 \* a21) / (a11 \* a22 - a12 \* a21)  
 return dy  
  
  
L = 0.2 *# Длина стержня*ALPHA = 30 *# Угол между AC и AO*m1 = 1 *# Масса диска*m2 = 0.2 *# Масса груза*R = 0.2 *# радиус диска*C = 1.95 *# Жесткость пружины*E = R / math.sqrt(3) *# Расстояние от центра диска до горизонтальной оси*G = 9.81 *# Ускорение свободного падения*t\_fin = 20 *# Конечное время для симуляции*TIME = np.linspace(0, t\_fin, 1001) *# Полупериод вращения диска*PHI = math.pi / 12 *# Угол наклона диска во время вращения*TAU = 0 *# Начальный угол θ, задаётся равным нулю.*dphi = math.pi / 36 *# Начальная угловая скорость φ'*dtau = 0 *# Начальная угловая скорость θ'*y0 = [PHI, TAU, dphi, dtau] *# Вектор начальных условий состояния системы*Y = odeint(mechanical\_system\_dynamics, y0, TIME, (m1, m2, C, L, E, ALPHA, G)) *# Численное решение системы дифференциальных уравнений*print(Y.shape)  
  
PHI = Y[:, 0]  
TAU = Y[:, 1]  
  
*# Функция для нахождения координат дуги окружности для отрисовки диска*def draw\_arc(X, Y, radius):  
 C\_X = [X + radius \* math.cos(i / 100) for i in range(0, 628)]  
 C\_Y = [Y + radius \* math.sin(i / 100) for i in range(0, 628)]  
 return C\_X, C\_Y  
  
L = 8  
ALPHA = 30  
R = 7  
C = 10  
E = R / math.sqrt(3)  
T = np.linspace(0, 10, 1001)  
X\_O = 0  
Y\_O = 0  
X\_C = X\_O - E \* np.sin(PHI)  
Y\_C = Y\_O + E \* np.cos(PHI)  
X\_A = X\_C - R \* np.sin(math.pi / 2 + PHI)  
Y\_A = Y\_C + R \* np.cos(math.pi / 2 + PHI)  
X\_B = X\_A + L \* np.sin(TAU)  
Y\_B = Y\_A - L \* np.cos(TAU)  
  
*# Создаем график и устанавливаем для него параметры*fig = plt.figure(figsize=[13, 9])  
ax = fig.add\_subplot(1, 2, 1)  
ax.axis('equal')  
ax.set(xlim=[-25, 25], ylim=[-25, 25])  
  
*# Количество витков или число, определяющее, сколько раз спираль делает полный оборот*spiral\_branches = 1.1  
*# Начальный радиус спирали*R1 = 0.2  
*# Конечный радиус спирали*R2 = 6  
  
*# Массив углов для создания спирали (приблизительно равный 2pi)*spiral\_angle = np.linspace(0, spiral\_branches \* 6.28 - PHI[0], 100)  
  
*# Вычисление координаты по x для отрисовки спирали Архимеда*spiral\_spring\_x = -(R1 \* spiral\_angle \* (R2 - R1) / spiral\_angle[-1]) \* np.sin(spiral\_angle)  
*# Вычисление координаты по y для отрисовки спирали Архимеда*spiral\_spring\_y = (R1 \* spiral\_angle \* (R2 - R1) / spiral\_angle[-1]) \* np.cos(spiral\_angle)  
  
spiral\_spring = ax.plot(spiral\_spring\_x + X\_O, spiral\_spring\_y + Y\_O, color='black')[0]  
point\_C = ax.plot(X\_C[0], Y\_C[0], marker='o', markersize=12, color='black')[0]  
point\_O = ax.plot(X\_O, Y\_O, marker='o', color='black')[0]  
point\_A = ax.plot(X\_A, Y\_A, marker='o', color='black')[0]  
point\_B = ax.plot(X\_B, Y\_B, marker='o', color='black')[0]  
line\_AB = ax.plot([X\_A[0], X\_B[0]], [Y\_A[0], Y\_B[0]], color='black', linewidth=3)[0]  
line\_OC = ax.plot([X\_O, X\_C[0]], [Y\_O, Y\_C[0]], color='black')[0]  
disk\_arc, = ax.plot(\*draw\_arc(X\_C[0], Y\_C[0], R), 'red')  
triangle, = ax.plot([-1, 0, 1], [-2, 0, -2], color='black')  
line\_tr = ax.plot([- 1, 1], [-2, -2], color='black')[0]  
  
*# Вычисление силы реакции в шарнире A (R\_A)*RA = m1 \* G \* np.cos(PHI) + m2 \* G \* np.cos(TAU) - m2 \* L \* (dphi \*\* 2 + dtau \*\* 2)  
  
  
*# функция для отрисовки текущего состояния системы*def draw(i):  
 disk\_arc.set\_data(\*draw\_arc(X\_C[i], Y\_C[i], R))  
 point\_O.set\_data(X\_O, Y\_O)  
 point\_C.set\_data(X\_C[i], Y\_C[i])  
 point\_A.set\_data(X\_A[i], Y\_A[i])  
 line\_OC.set\_data([X\_O, X\_C[i]], [Y\_O, Y\_C[i]])  
 point\_B.set\_data(X\_B[i], Y\_B[i])  
 line\_AB.set\_data([X\_A[i], X\_B[i]], [Y\_A[i], Y\_B[i]])  
 spiral\_angle = np.linspace(0, spiral\_branches \* 5.6 + PHI[i], 100)  
 spiral\_spring\_x = -(R1 \* spiral\_angle \* (R2 - R1) / spiral\_angle[-1]) \* np.sin(spiral\_angle)  
 spiral\_spring\_y = (R1 \* spiral\_angle \* (R2 - R1) / spiral\_angle[-1]) \* np.cos(spiral\_angle)  
 spiral\_spring.set\_data(spiral\_spring\_x + X\_O, spiral\_spring\_y + Y\_O)  
 return [disk\_arc, point\_O, point\_C, line\_OC, spiral\_spring, point\_A, point\_B, line\_AB]  
  
  
anim = FuncAnimation(fig, draw, frames=1000, interval=10, repeat=False)  
fig.suptitle('Borisov LAB\_3', fontsize=14)  
  
  
  
*# Построение графиков*fig, axs = plt.subplots(3, 1, figsize=(10, 8), sharex=True)  
  
*# График phi(t)*axs[0].plot(TIME, PHI, label='$\phi(t)$', color='blue')  
axs[0].set\_ylabel('$\phi(t)$ (rad)')  
axs[0].grid(True)  
axs[0].legend()  
  
*# График theta(t)*axs[1].plot(TIME, TAU, label='$\theta(t)$', color='green')  
axs[1].set\_ylabel('$\theta(t)$ (rad)')  
axs[1].grid(True)  
axs[1].legend()  
  
*# График R\_A(t)*axs[2].plot(TIME, RA, label='$R\_A(t)$', color='red')  
axs[2].set\_xlabel('Time (s)')  
axs[2].set\_ylabel('$R\_A(t)$ (N)')  
  
axs[2].grid(True)  
axs[2].legend()  
  
*# чтобы не накладывались названия*plt.tight\_layout()  
plt.show()

**Результат работы программы:**

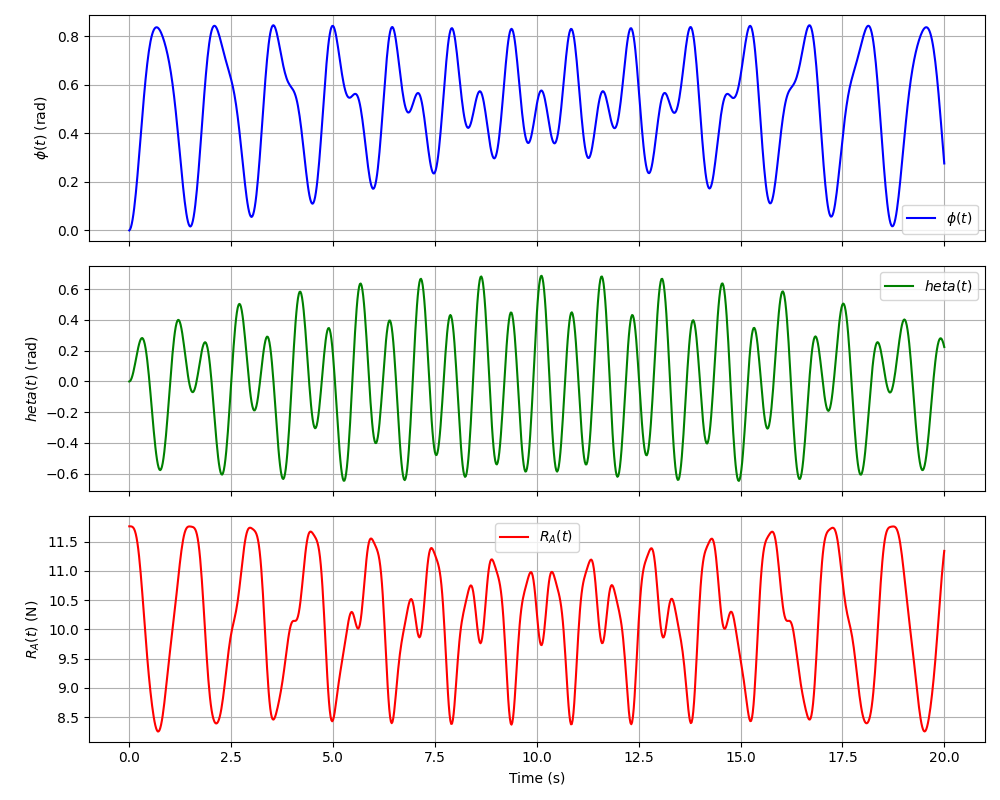
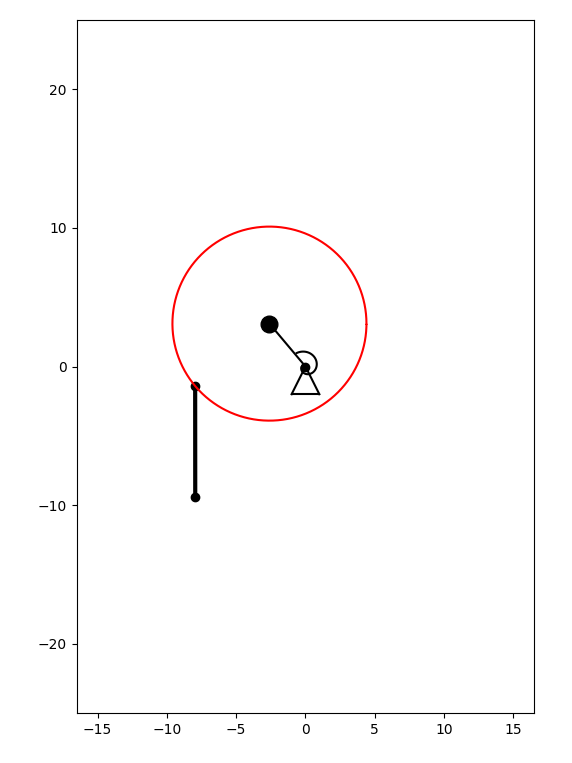
1. m1 = 1, m2 = 0.2, l = 0.2, r = 0.2, c = 1.95, g = 9.81, phi = π/12, tau = 0, dphi = π/36, dtau = 0.



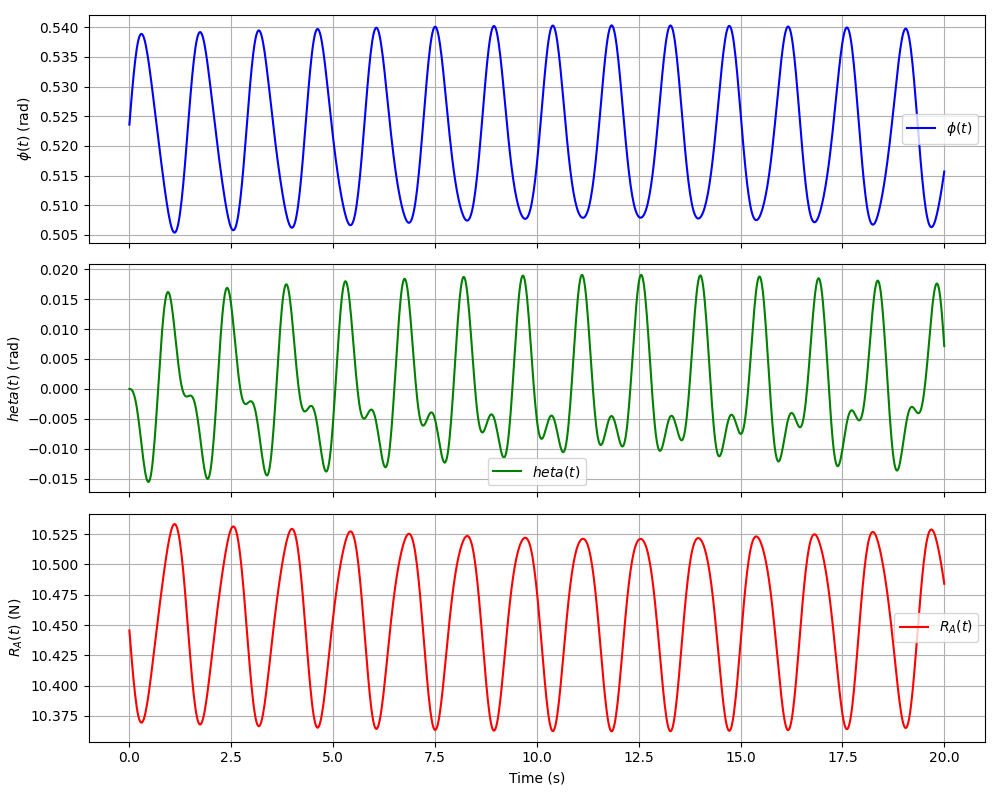
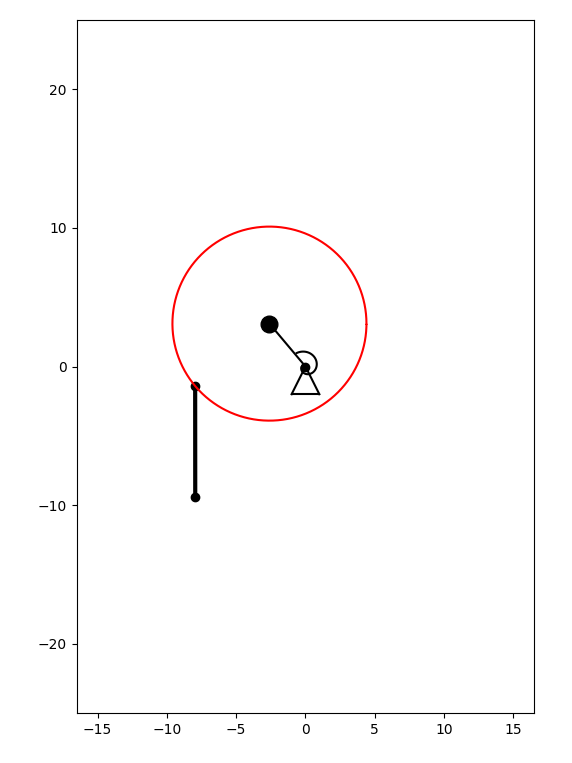
1. m1 = 1, m2 = 0.2, l = 0.2, r = 0.2, c = 10, g = 9.81, phi = π/12, tau = 0, dphi = π/36, dtau = 0. – коэффициент жесткости увеличен.



1. m1 = 1, m2 = 0.2, l = 0.2, r = 1, c = 1.95, g = 9.81, phi = 0, tau = 0, dphi = 0, dtau = 0.



1. m1 = 1, m2 = 0.2, l = 0.2, r = 1, c = 1.95, g = 9.81, phi = 0, tau = π/6, dphi = 0 , dtau = 0



**Вывод:**

В ходе лабораторной работы с помощью языка программирования Python я смог построить анимацию движения системы, а также графики законов движения системы, поэкспериментировала с различными значениями для системы.