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:

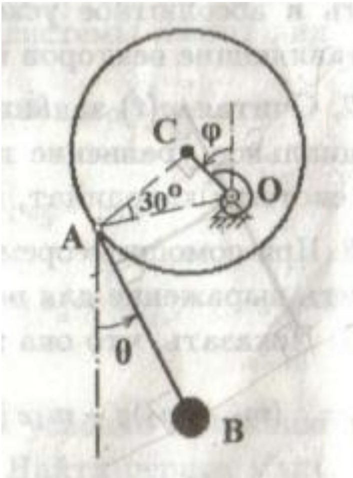


Рис. 2

√10. Используя уравнения Лагранжа второго рода, показать, что дифференциальные уравнения движения системы имеют вид

$$[(5/6)m_1 + (4/3)m_2]\, r^2 \ddot{\varphi} + 2m_2 \ell e \left[\ddot{\theta} \sin\alpha - \dot{\theta}^2 \cos\alpha\right] = \left[m_1 \sin\varphi + 2m_2 \cos\left(\varphi - \frac{\pi}{6}\right)\right] eg - c\varphi,$$

$$2e\left[\ddot{\varphi} \sin\alpha + \dot{\varphi}^2 \cos\alpha\right] + \ell\ddot{\theta} = -g\sin\theta.$$

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 -
```

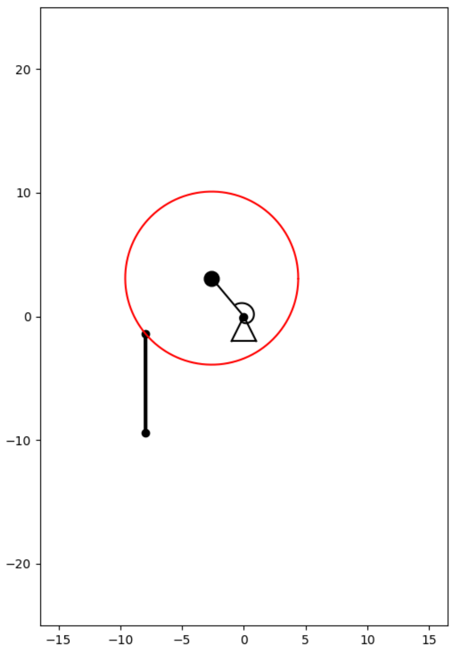
```
c * y[0] + 2 * m2 * l * e * y[3] ** 2 * math.cos(alpha)
    b2 = -g * np.sin(y[1]) - 2 * e * y[2] ** 2 * math.cos(alpha)
    dy[2] = (b1 * a22 - b2 * a12) / (a11 * a22 - a12 * a21)
    dy[3] = (b2 * a11 - b1 * a21) / (a11 * a22 - a12 * a21)
    return dy


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
```

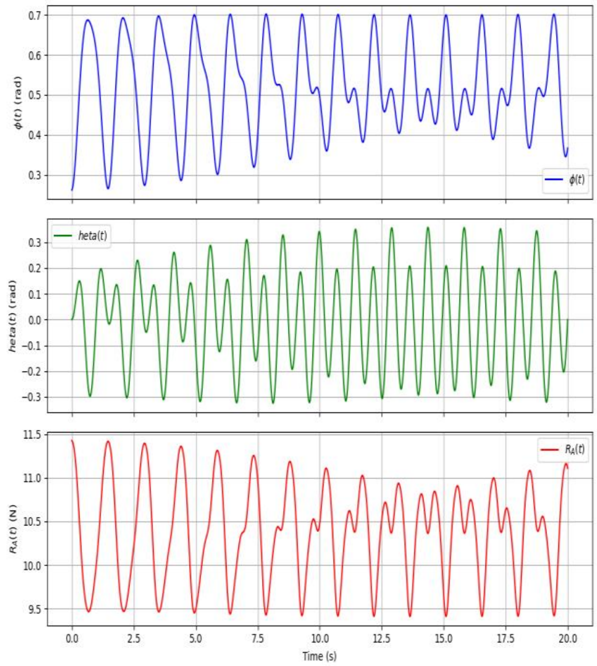
```
# Массив углов для создания спирали (приблизительно равный 2pi)
spiral_angle = np.linspace(0, spiral_branches * 6.28 - PHI[0], 100)

# 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()
```

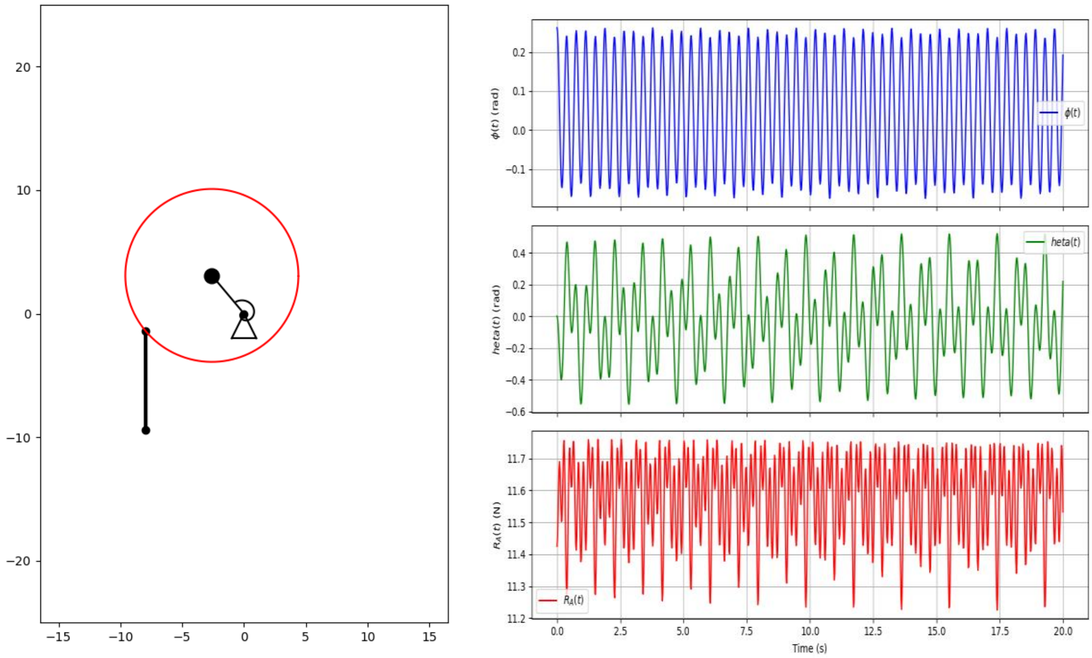
Результат работы программы:

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.

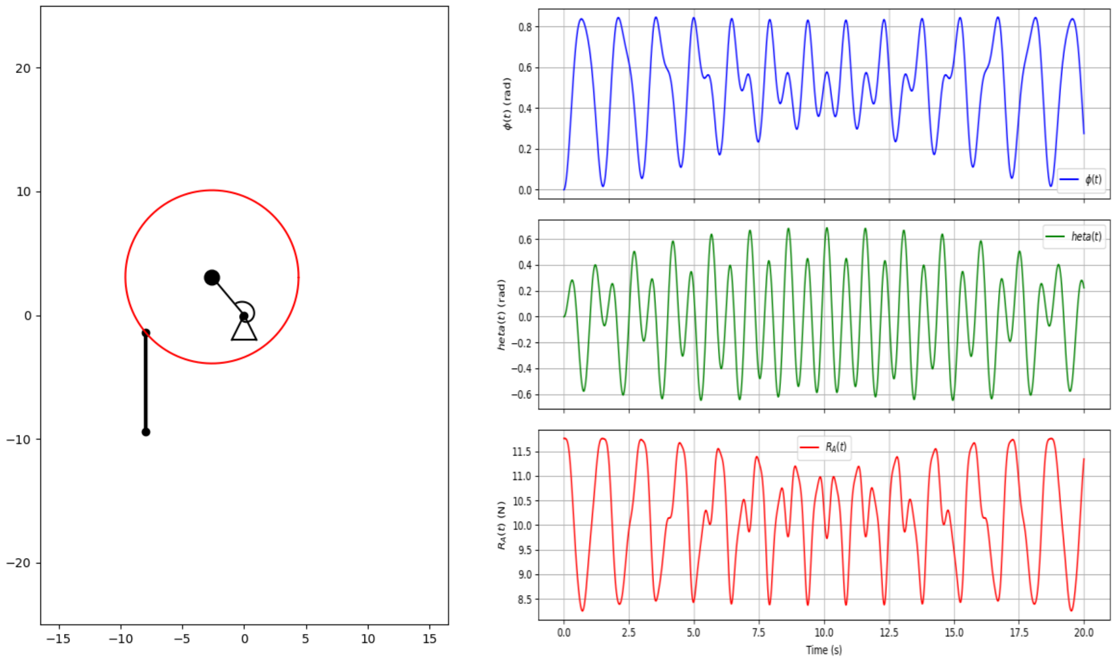




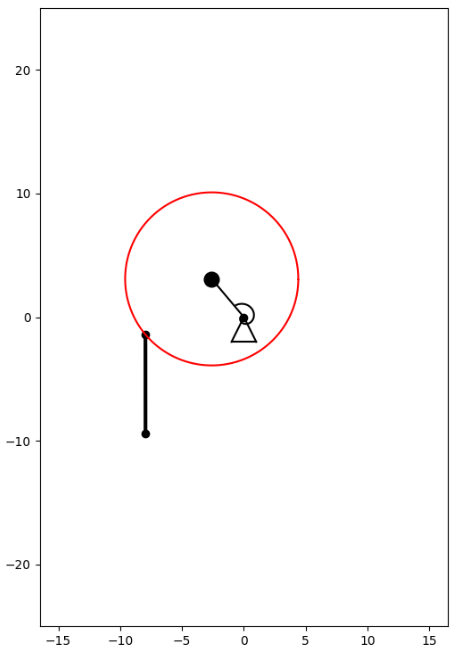
2) m1 = 1, m2 = 0.2, l = 0.2, r = 0.2, c = 10, g = 9.81, phi = π/12, tau = 0, dphi = π/36, dtau = 0. – коэффициент жесткости увеличен.

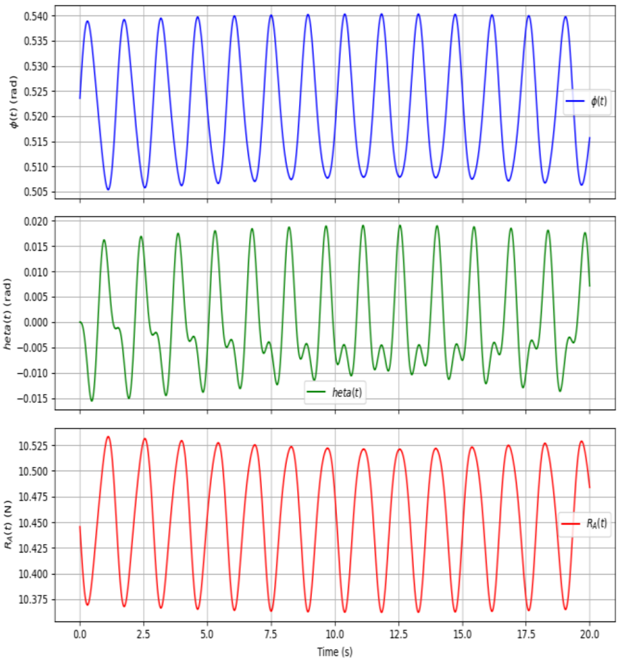


3) m1 = 1, m2 = 0.2, l = 0.2, r = 1, c = 1.95, g = 9.81, phi = 0, tau = 0, dphi = 0, dtau = 0.



4) 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.