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 №20**

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

Сисенов Марат Мийрамович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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, а также графики законов движения системы и указанных в задании реакций для разных случаев системы.

**Задание системы 20 варианта формулируется следующим образом:**

Однородная балка DE длины 2а и массы m1 закреплена в не-

подвижном шарнире D и концом Е соединена с пружиной жесткость-

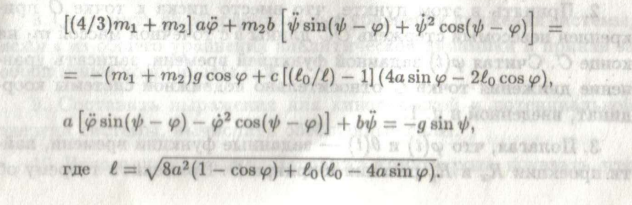
ти с (рис. 20). К середине балки прикреплен невесомый стержень

AD длины и с точечным грузом массы m2 на конце В. Длина неде-

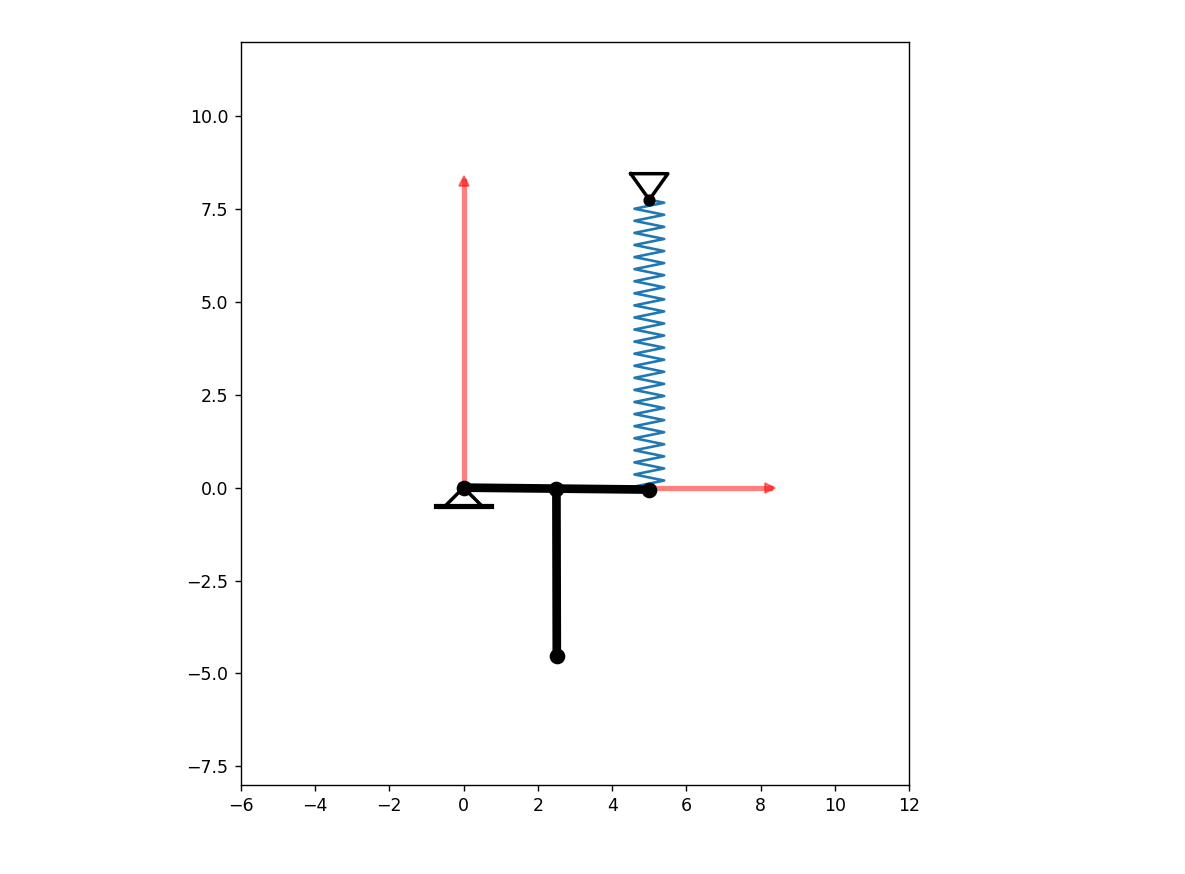
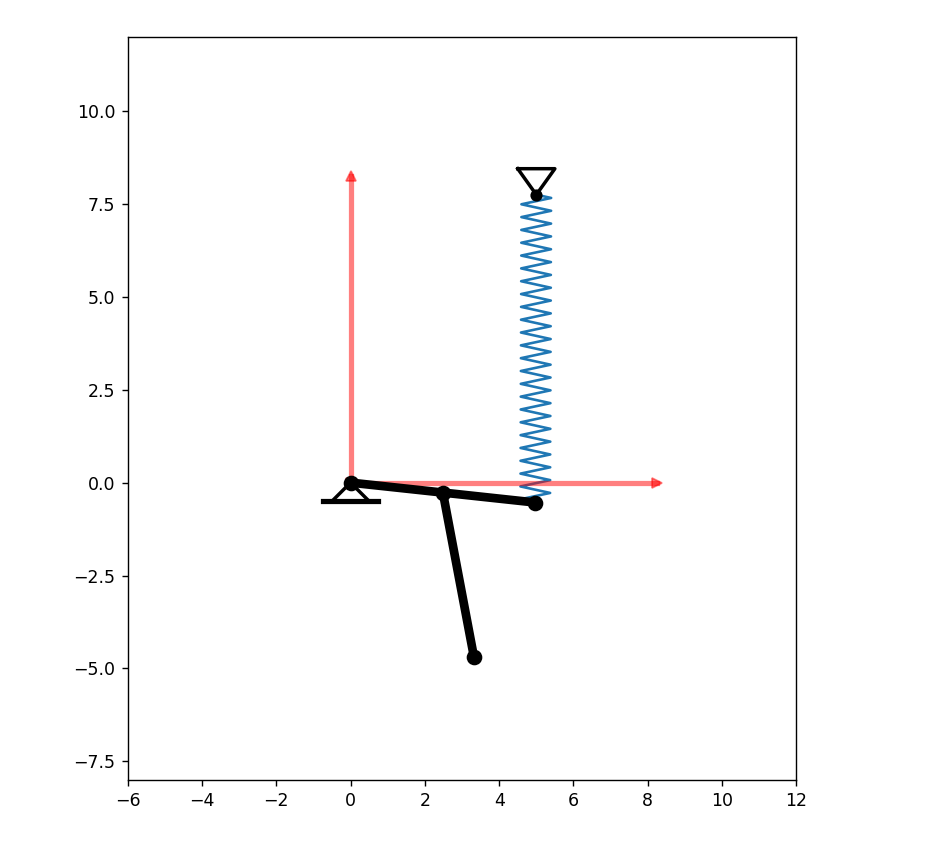
формированной пружины равна l0; при горизонтальном положении

балки DE пружина вертикальная.

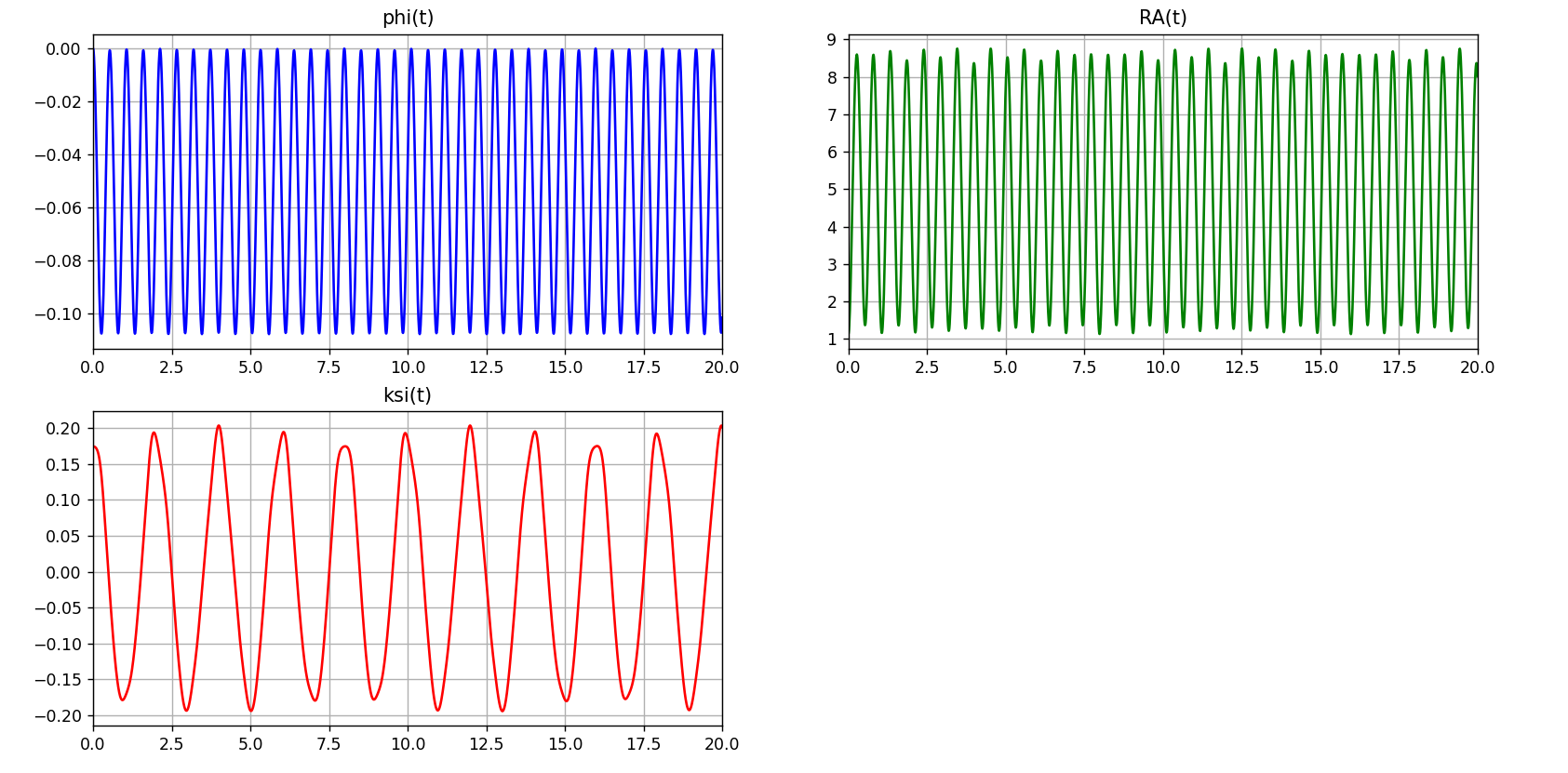
Дифференциальные уравнения движения системы имеют вид:



**Рисунок получившейся анимации движения:**



**Графики силы реакци:**



**Код программы**

import numpy as np  
import matplotlib.pyplot as plt  
from matplotlib.animation import FuncAnimation  
import math  
from scipy.integrate import odeint  
  
def odesys(y, t, m1, m2, a, b, l0, c, g):  
 l = ((8\*(a\*\*2)\*(1 - np.cos(y[0]))) + (l0\*(l0 - 4\*a\*np.sin(y[0]))))\*\*(0.5)  
 dy = np.zeros(4)  
 dy[0] = y[2] #dphi  
 dy[1] = y[3] #dksi  
  
 a11 = a\*((4/3) \* m1 + m2)  
 a12 = m2\*b\*np.sin(y[1] - y[0])  
 a21 = a \* np.sin(y[1] - y[0])  
 a22 = b  
  
 b1 = c \* ((l0/l) - 1) \* (4\*a\*np.sin(y[0]) - 2\*l0\*np.cos(y[0])) - ((m1 + m2)\*g\*np.cos(y[0])) - (m2\*b\*((y[3])\*\*2)\*np.cos(y[1]-y[0]))  
  
 b2 = (a \* ((y[3])\*\*2) \* np.cos(y[1] - y[0])) - (g \* np.sin(y[1]))  
  
 dy[2] = (b1\*a22 - b2\*a12)/(a11\*a22 - a12\*a21) #dphi(t)  
 dy[3] = (b2\*a11 - b1\*a21)/(a11\*a22 - a12\*a21) #dksi(t)  
  
 return dy  
  
m1 = 5  
m2 = 0.5  
a = b = l0 = 1  
c = 250  
g = 9.80665  
   
# функция для поворота  
def Rot2D(X, Y, Alpha):  
 RX = X\*np.cos(Alpha) - Y\*np.sin(Alpha)  
 RY = X\*np.sin(Alpha) + Y\*np.cos(Alpha)  
 return RX, RY  
  
# вспомогательные данные для заполнения массивов, кол-во шагов и тд  
Steps = 2001  
t\_fin = 20  
t = np.linspace(0, t\_fin, Steps)  
  
# Задаю начальные данные системы  
phi0 = 0  
ksi0 = math.pi / 18  
dphi0 = dksi0 = 0  
y0 = [phi0, ksi0, dphi0, dksi0]  
  
Y = odeint(odesys, y0, t, (m1, m2, a, b, l0, c, g))  
  
phi = Y[:,0]  
ksi = Y[:,1]  
dphi = Y[:,2]  
dksi = Y[:,3]  
ddphi = [odesys(y,t,m1, m2, a, b, l0, c, g)[2] for y,t in zip(Y,t)] #вычисляется угловое ускорение ddphi с использованием функции odesys для каждого момента времени t  
ddksi = [odesys(y,t,m1, m2, a, b, l0, c, g)[3] for y,t in zip(Y,t)] #вычисляется угловое ускорение ddksi с использованием функции odesys для каждого момента времени t  
  
RA = m2 \* (g \* np.cos(ksi) + b \* (dksi\*\*2) + a\*(ddphi\*np.cos(ksi - phi) + (dphi\*\*2)\*np.sin(ksi - phi)))  
  
# создаю окно для отрисовки графиков и отрисовываю их  
fig\_for\_graphs = plt.figure(figsize=[13,7])  
  
ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 1)  
ax\_for\_graphs.plot(t, phi, color='Blue')  
ax\_for\_graphs.set\_title("phi(t)")  
ax\_for\_graphs.set(xlim=[0, t\_fin])  
ax\_for\_graphs.grid(True)  
  
ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 3)  
ax\_for\_graphs.plot(t, ksi, color='Red')  
ax\_for\_graphs.set\_title("ksi(t)")  
ax\_for\_graphs.set(xlim=[0, t\_fin])  
ax\_for\_graphs.grid(True)  
  
ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 2)  
ax\_for\_graphs.plot(t, RA, color='Green')  
ax\_for\_graphs.set\_title("RA(t)")  
ax\_for\_graphs.set(xlim=[0, t\_fin])  
ax\_for\_graphs.grid(True)  
  
a = 2.5  
lenDE = 2\*a #длина стержня DE  
l = a \* 1.8 #длина стержня АВ  
l0 = 1.55 \* lenDE #высота, на которой закреплена пружина  
D = np.array([0, 0])  
  
# прорисовка пружины  
K = 50  
Sh = 0.4  
b = 1/(K-2)  
X\_Spr = np.zeros(K)  
Y\_Spr = np.zeros(K)  
X\_Spr[0] = 0  
Y\_Spr[0] = 0  
X\_Spr[K-1] = 1  
Y\_Spr[K-1] = 0  
  
for i in range(K-2):  
 X\_Spr[i+1] = b\*((i+1) - 1/2)  
 Y\_Spr[i+1] = Sh\*(-1)\*\*i  
  
# Координаты точки Е  
Ex = lenDE \* np.cos(phi)  
Ey = lenDE \* np.sin(phi)  
  
# смещение точки B относительно точки А  
Bx = l \* np.sin(ksi)  
By = l \* np.cos(ksi)  
  
# подсчитываю отрезки, необходимые для вычисления угла наклона пружины  
Spr\_x = lenDE - Ex  
Spr\_y = l0 - Ey  
# это длина пружины в каждый момент времени  
length\_Spr = (Spr\_x\*\*2 + Spr\_y\*\*2)\*\*(0.5)  
  
# Отрисовка окна, его параметризация  
fig = plt.figure(figsize=[20, 10])  
ax = fig.add\_subplot(1, 1, 1)  
ax.set\_aspect('equal')#чтобы оси были равнымич  
ax.set\_ylim([-8, 12])  
ax.set\_xlim([-6, 12])  
  
# ПРУЖИНА  
  
# получаю координаты пружины после поворота  
Spr\_x\_L\_fi, Spr\_y\_fi = Rot2D(X\_Spr, Y\_Spr, -(math.pi/2 + abs(math.atan2( Spr\_x[0], Spr\_y[0]))))  
# задаю пружину уже после повторота, причём сразу перемещаю её в конечную позицию и растягиваю на длину  
WArrow, = ax.plot(Spr\_x\_L\_fi + lenDE, (Spr\_y\_fi\*length\_Spr[0]) + l0)  
#крепёж для пружины  
ax.plot(2\*a, l0, color='black', linewidth=5, marker='o')  
ax.plot([2\*a-0.5, 2\*a+0.5, 2\*a, 2\*a-0.5], [l0+0.7, l0+0.7, l0, l0+0.7], color='black', linewidth=2, )  
  
ax.plot([-0.5, 0.5, 0, -0.5], [-0.5, -0.5, 0, -0.5], color='black', linewidth=2)  
ax.plot([-0.75, 0.75], [-0.5, -0.5], color='black', linewidth=3)  
# рисую оси  
ax.plot([0, 0], [0, 8.25], color='red', linewidth=3, linestyle='solid', alpha=0.5, marker = '^')  
ax.plot([0, 8.25], [0, 0], color='red', linewidth=3, linestyle='solid', alpha=0.5, marker = '>')  
# рисую отрезок DE  
Drawed\_DE = ax.plot(np.array([0, Ex[0]]), np.array([0, Ey[0]]), color='black', linewidth=5, marker='o', markersize=8)[0]  
# рисую отрезок АВ  
Drawed\_AB = ax.plot(np.array([Ex[0]/2, Ex[0]/2 + Bx[0]]), np.array([Ey[0]/2, Ey[0]/2 - By[0]]), color='black', linewidth=5, marker='o', markersize=8)[0]  
  
def anima(i):  
  
 # отрисовываю отрезок DE  
 Drawed\_DE.set\_data(np.array([0, Ex[i]]), np.array([0, Ey[i]]))  
 # получаю координаты пружины после поворота  
 Spr\_x\_L\_fi, Spr\_y\_fi = Rot2D(X\_Spr\*length\_Spr[i], Y\_Spr, -(math.pi/2 + abs(math.atan2(Spr\_x[i], Spr\_y[i]))))  
 # задаю пружину уже после повторота, причём сразу перемещаю её в конечную позицию и растягиваю на длину  
 WArrow.set\_data(Spr\_x\_L\_fi + lenDE, (Spr\_y\_fi) + l0)  
  
  
 Drawed\_AB.set\_data(np.array([Ex[i]/2, Ex[i]/2 + Bx[i]]), np.array([Ey[i]/2, Ey[i]/2 - By[i]]))  
 return Drawed\_DE  
  
anim = FuncAnimation(fig, anima, frames=len(t), interval=100, repeat=False)  
  
plt.show()

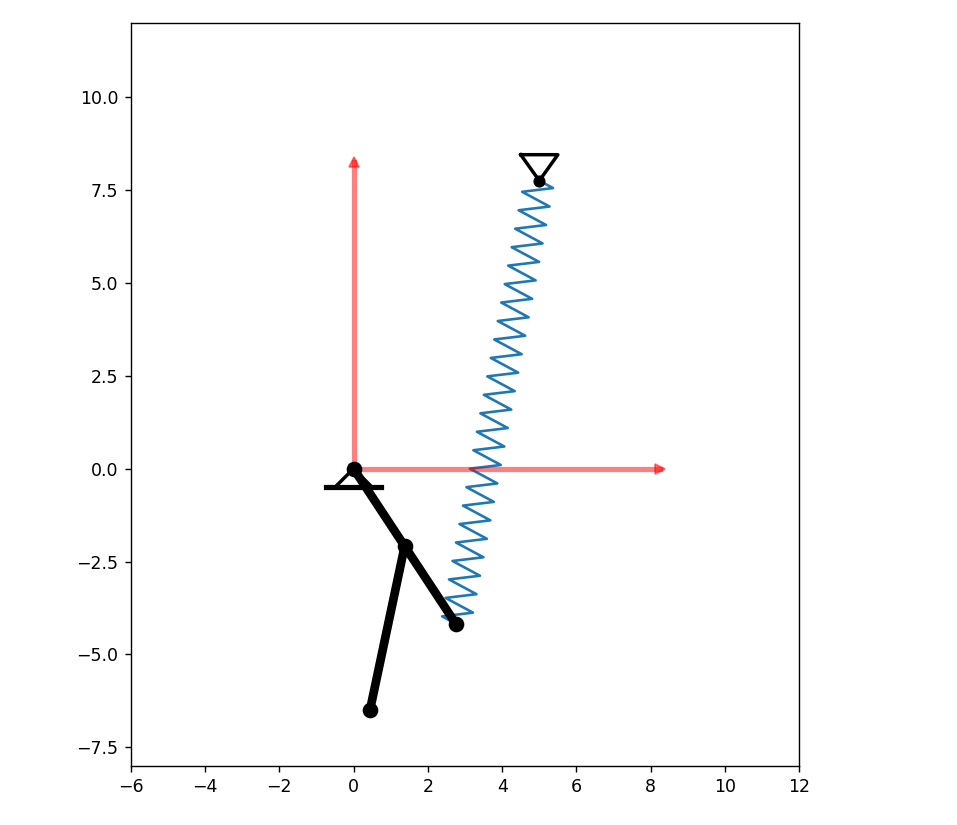
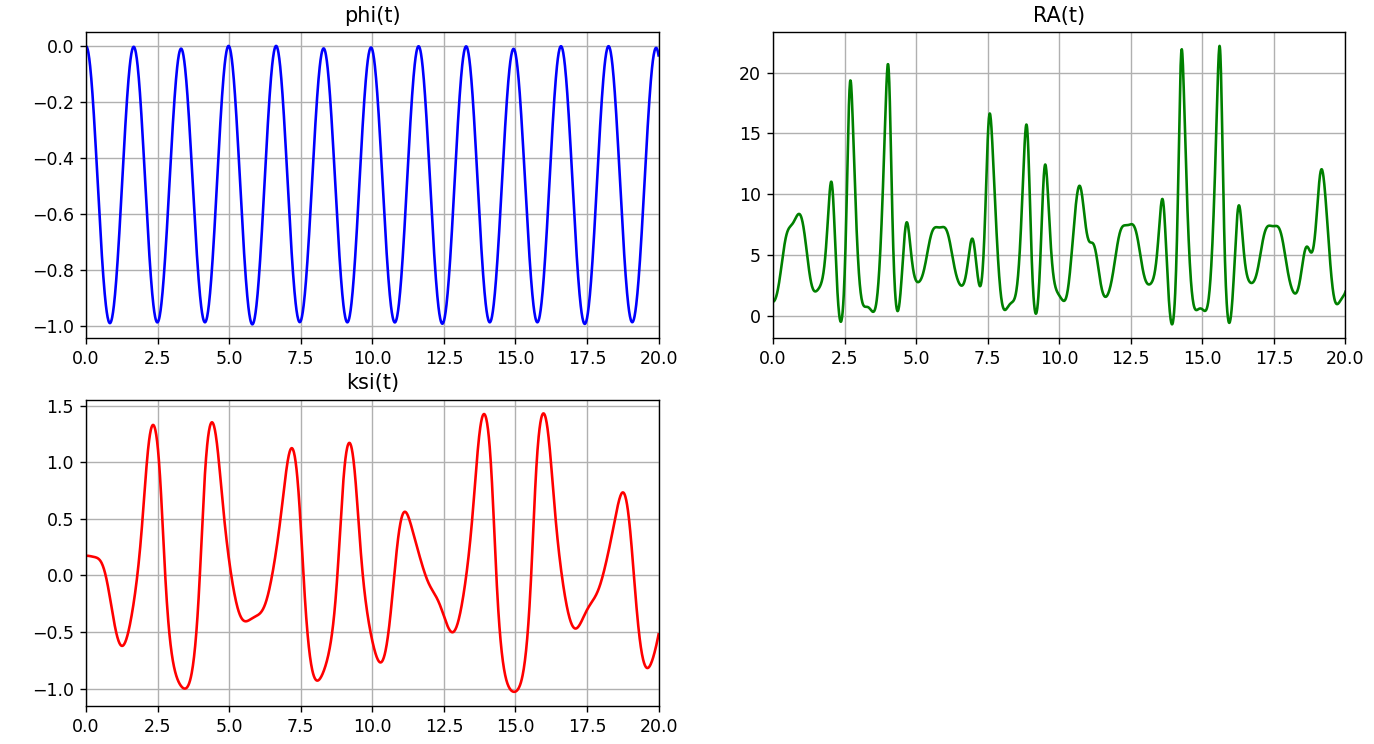
**Примеры работы:**

Приведу несколько примеров работы моей программы.

1. Начальные условия из учебника

m1 = 50кг, m2 = 0.5кг, a = b = l0 = 1м, c = 250 Н/м, t0 = 0, φ0 = 0, ψ0 =π/18,

dφ0 = ψ0 = 0.

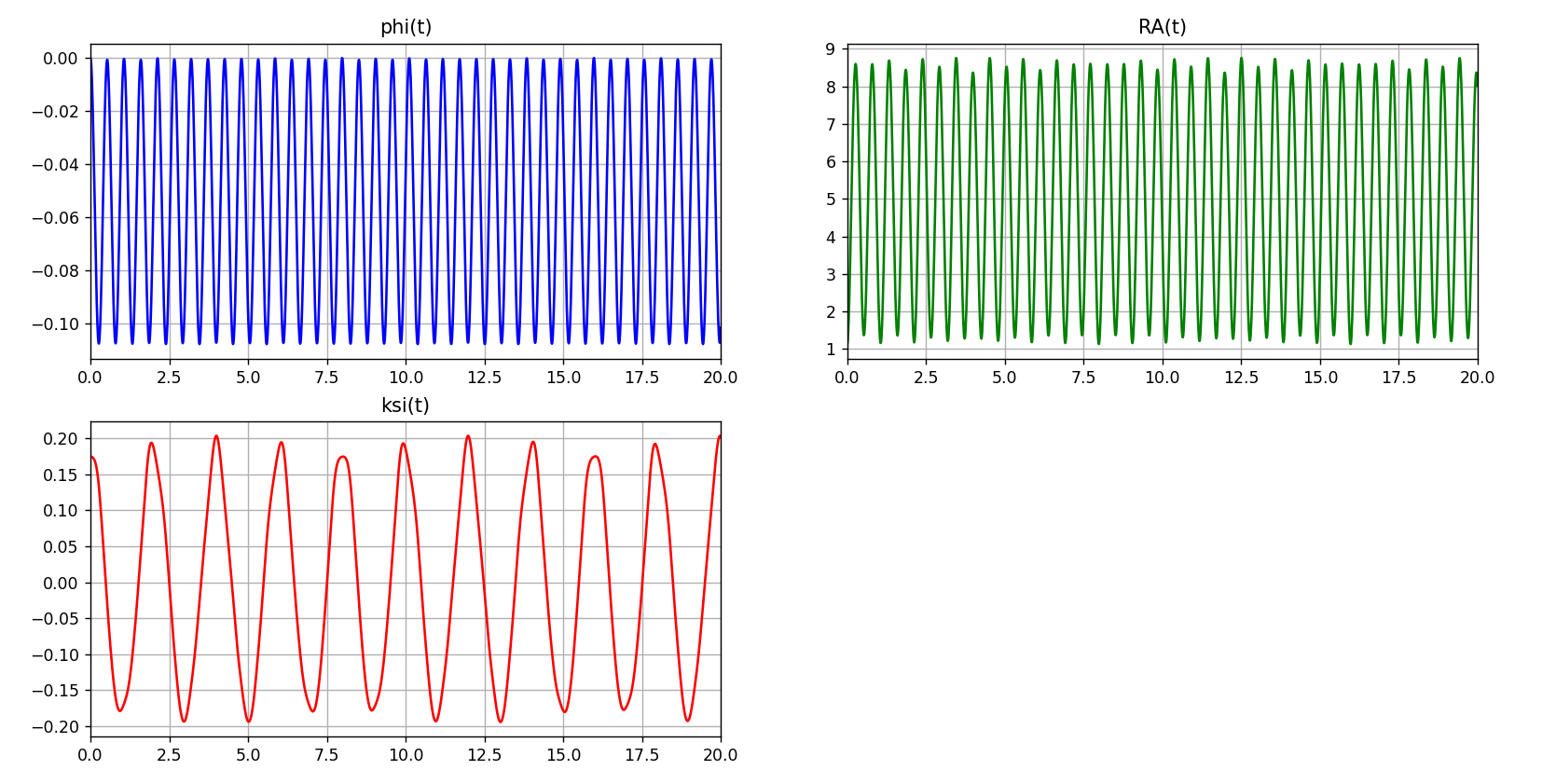
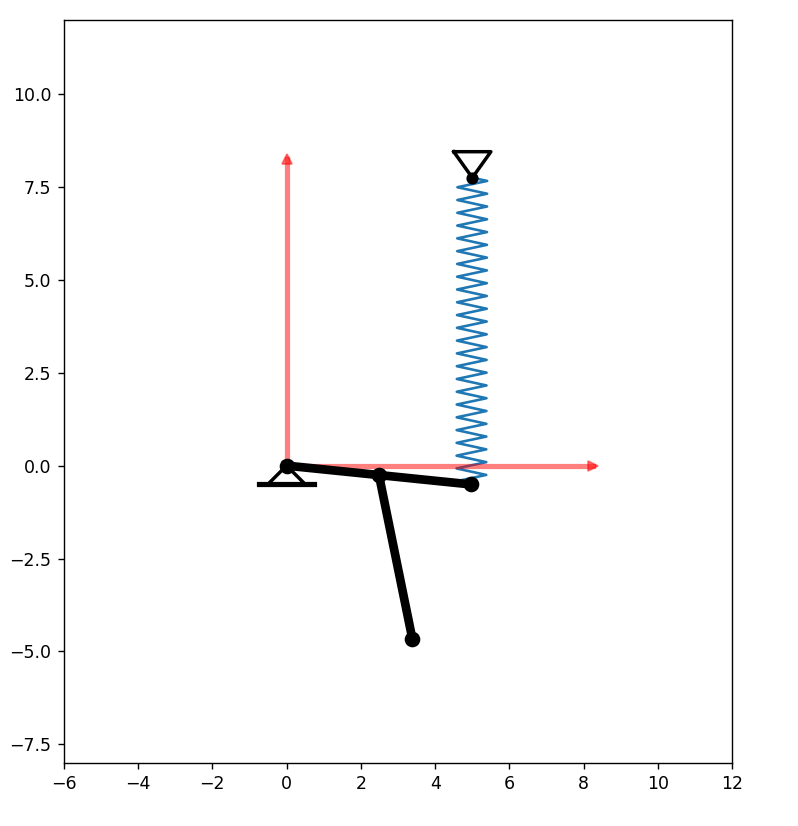
 

Движение стержней медленное, они начинают движение с маленького угла и с небольшим ускорением, из-за чего двигаются небыстро. Стержень и пружина совершают колебательное движение, которое можно считать незатухающим, так как мы не учитываем внешние силы.

1. Уменьшим массу в 10 раз относительно начальных значений

m1 = 5кг, m2 = 0.5кг, a = b = l0 = 1м, c = 250 Н/м, t0 = 0, φ0 = 0, ψ0 =π/18,

dφ0 = ψ0 = 0.

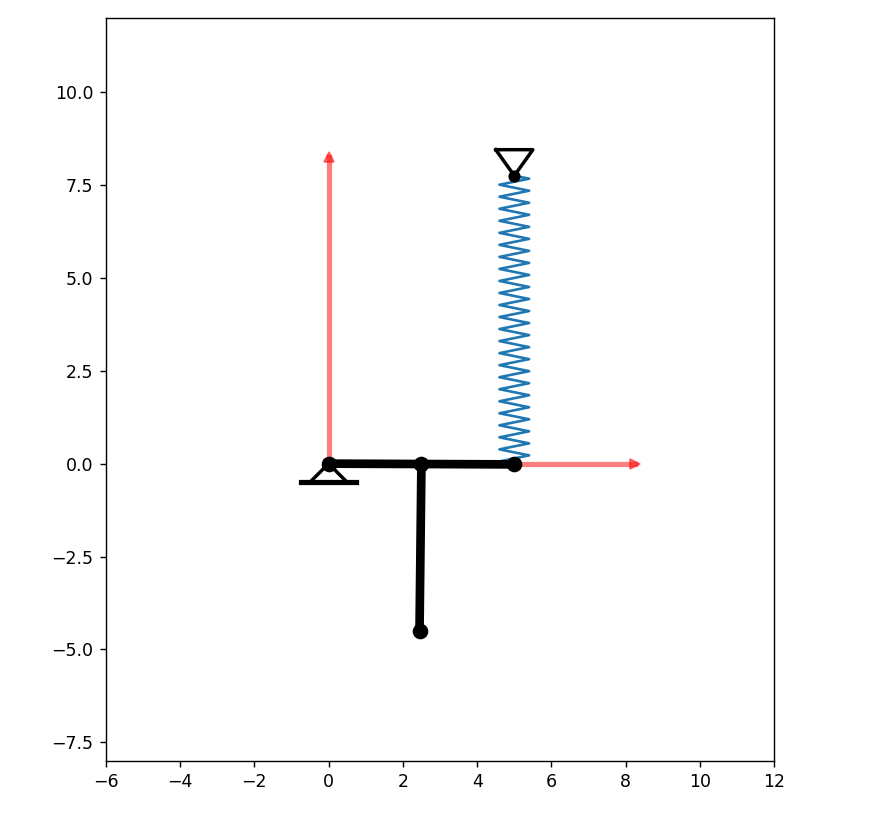
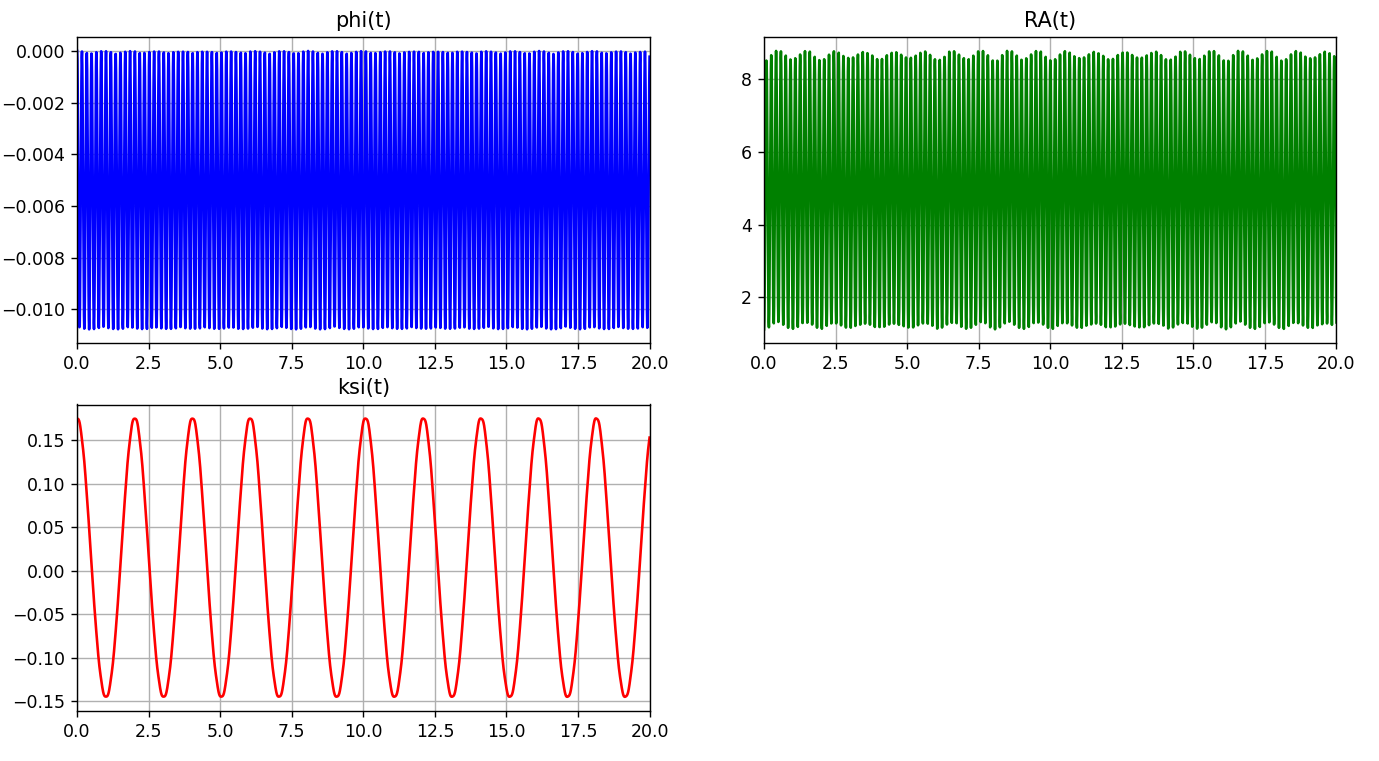


Видно, что стержень отклоняется недалеко от начального положения, так как мы уменьшили массу в 10 раз и его силы тяжести хватает только на небольшое отклонение.

3. Увеличим жесткость пружины в 10 раз:

m1 = 5кг, m2 = 0.5кг, a = b = l0 = 1м, c = 2500 Н/м, t0 = 0, φ0 = 0, ψ0 =π/18,

dφ0 = ψ0 = 0.

Стержни совершают еле заметные колебания, так как пружина имеет довольную большую жесткость, и силы тяжести стержней не хватает на большое отклонение от начала координат.

**Вывод**

Я успешно выполнил лабораторную работу по теоретической механике. С помощью языка программирования Python и библиотек matplotlib и numpy я схематично проанимировал движение двух стержней и пружины и решил систему дифференциальных уравнений.

Благодаря этой лабораторной работе, я научился работать с 2д анимацией в matplotlib и реализовал основание для выполнения следующей лабораторной работы.

В моей программе используются реальные законы движения, благодаря чему можно посмотреть, как эта система будет вести себя в реальной жизни.