ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ CИСТЕМЫ»**

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

**ВАРИАНТ ЗАДАНИЯ № 22**

Выполнил(а) студент группы М8О-212Б-22

Куценко М.Д.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

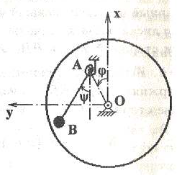
подпись, дата

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

Москва, 2023

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

*Вариант 22:*

Однородный диск массы **m1** и радиуса **R** вращается вокруг горизонтальной неподвижной оси, перпендикулярной его плоскости и проходящей через центр O. В точке A (OA=**a**) к диску шарнирно прикреплён невесомый стержень AB длины **l** с грузом B массы **m2** на конце. Спиральная пружина жесткости с прикреплена своими концами к диску и стрежню. Пружина не деформирована при **φ**=**ψ**=0. Трение на оси O отсутствует.

*Код:*

import numpy as np

import math

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation, FFMpegWriter

### ИЗМЕНЯЕМЫЕ ДАННЫЕ

X\_C = 5 # координаты центра диска

Y\_C = 5

R = 4 # радиус диска

A = 2 # расстояние между шарниром и центром диска

L = 2 # длина стержня, на котором шарнирно прикреплён груз

### СТАТИЧЕСКАЯ ЧАСТЬ

ang = np.linspace(0, 2\*math.pi, 80) # углы для отрисовки кругов

X\_Disk = X\_C + R\*np.cos(ang) # координаты диска

Y\_Disk = Y\_C + R\*np.sin(ang)

X\_Sm = X\_C + 0.2\*np.cos(ang) # координаты маленького круга в центре диска

Y\_Sm = Y\_C + 0.2\*np.sin(ang)

X\_Side\_1 = [X\_C+0.2\*np.cos(math.pi\*5/4), X\_C+0.5\*np.cos(math.pi\*5/4)] # боковые линии (центр)

Y\_Side\_1 = [X\_C+0.2\*np.sin(math.pi\*5/4), Y\_C+0.5\*np.sin(math.pi\*5/4)]

X\_Side\_2 = [X\_C+0.2\*np.cos(math.pi/-4), X\_C+0.5\*np.cos(math.pi/-4)]

Y\_Side\_2 = [X\_C+0.2\*np.sin(math.pi/-4), Y\_C+0.5\*np.sin(math.pi/-4)]

X\_Bottom = [X\_Side\_1[1]-0.1, X\_Side\_2[1]+0.1] # линия-закреп центра

Y\_Bottom = [Y\_Side\_1[1], Y\_Side\_2[1]]

X\_Lines\_1 = np.linspace(float(X\_Bottom[0])+0.05, float(X\_Bottom[1])-0.05, 5) # полоски на линии-закрепа центра диска

X\_Lines\_2 = X\_Lines\_1 + 0.3\*np.cos(math.pi\*9/8)

Y\_Lines\_1 = np.full(5, Y\_Bottom[0])

Y\_Lines\_2 = Y\_Lines\_1 + 0.3\*np.sin(math.pi\*9/8)

### ДИНАМИЧЕСКАЯ ЧАСТЬ

Steps = 250

t\_fin = 5

t = np.linspace(0, t\_fin, Steps) # время

phi = np.zeros\_like(t) # угол между вертикальной осью и радиус-вектором к шарниру

psi = np.zeros\_like(t) # угол между вертикальной осью и стержнем

X\_Sh = np.zeros\_like(t) # координаты шарнира

Y\_Sh = np.zeros\_like(t)

X\_Gr = np.zeros\_like(t) # координаты груза

Y\_Gr = np.zeros\_like(t)

for i in np.arange(len(t)): # просчёт основных величин

phi[i] = 1.5\*np.sin(1.7\*t[i]) + 3.75\*np.cos(1.2\*t[i])

psi[i] = np.sin(1.7\*t[i]) + 2.5\*np.cos(1.2\*t[i])

X\_Sh[i] = X\_C + A\*np.cos(phi[i]+math.pi/2)

Y\_Sh[i] = Y\_C + A\*np.sin(phi[i]+math.pi/2)

X\_Gr[i] = X\_Sh[i] + L\*np.cos(-psi[i]-math.pi/2)

Y\_Gr[i] = Y\_Sh[i] + L\*np.sin(-psi[i]-math.pi/2)

### ПЕРЕХОД К ОТРИСОВКЕ

fig = plt.figure() # задаём пространство для отрисовки

ax = fig.add\_subplot(1, 1, 1)

ax.axis('equal')

ax.set(xlim = [0, 10], ylim = [0, 10])

### СТАТИЧЕСКАЯ ОТРИСОВКА

ax.plot(X\_C, Y\_C, marker = 'o', markersize=2, color = 'blue') # отрисовка центра диска

ax.plot(X\_Disk, Y\_Disk, color = 'blue') # отрисовка диска

ax.plot(X\_Sm, Y\_Sm, color = 'blue') # отрисовка кружка вокруг центра диска

ax.plot(X\_Side\_1, Y\_Side\_1, color = 'blue') # отрисовка боковых линий от центра диска

ax.plot(X\_Side\_2, Y\_Side\_2, color = 'blue')

ax.plot(X\_Bottom, Y\_Bottom, color = 'blue') # отрисовка линии-закрепа центра диска

for i in np.arange(len(X\_Lines\_1)): # отрисовка штрихов на линии-закрепе центра диска

ax.plot([X\_Lines\_1[i], X\_Lines\_2[i]], [Y\_Lines\_1[i], Y\_Lines\_2[i]], color = 'darkblue')

### ДИНАМИЧЕСКАЯ ОТРИСОВКА

LEN = 0.4 # длина линии-закрепа пружинки

WIDE = 0.2 # ширина линии-закрепа пружинки

X\_DSHT = 0.1\*np.cos(math.pi/4) # сдвиги штрихов по координатам

Y\_DSHT = 0.1\*np.cos(math.pi/4)

R1 = 0.3 # радиусы спиральной пружины

R2 = 0.1

thetta = np.linspace(0, 3/2\*math.pi+psi[0], 100) # угол проворота спиральной пружины

X\_SpiralSpr = (R1 + thetta\*(R2-R1)/thetta[-1])\*np.cos(thetta) # координаты точек спиральной пружины

Y\_SpiralSpr = -(R1 + thetta\*(R2-R1)/thetta[-1])\*np.sin(thetta)

spr, = ax.plot(X\_SpiralSpr+X\_Sh[0], Y\_SpiralSpr+Y\_Sh[0], color = 'green') # отрисовка спиральной пружины

pl1, = ax.plot([X\_Sh[0]+R1-WIDE/2, X\_Sh[0]+R1-WIDE/2+X\_DSHT], [Y\_Sh[0]+LEN, Y\_Sh[0]+LEN+Y\_DSHT], color = 'darkgreen') # штрихи на линии-закрепе спиральки

pl2, = ax.plot([X\_Sh[0]+R1, X\_Sh[0]+R1+X\_DSHT], [Y\_Sh[0]+LEN, Y\_Sh[0]+LEN+Y\_DSHT], color = 'darkgreen')

pl3, = ax.plot([X\_Sh[0]+R1+WIDE/2, X\_Sh[0]+R1+WIDE/2+X\_DSHT], [Y\_Sh[0]+LEN, Y\_Sh[0]+LEN+Y\_DSHT], color = 'darkgreen')

hl, = ax.plot([X\_Sh[0]+R1-WIDE/2-0.05, X\_Sh[0]+R1+WIDE/2+0.05], [Y\_Sh[0]+LEN, Y\_Sh[0]+LEN], color = 'green') # отрисовка вертикальной линии от спиральки

upl, = ax.plot([X\_Sh[0]+R1-0.0015, X\_Sh[0]+R1-0.0015], [Y\_Sh[0], Y\_Sh[0]+LEN], color = 'green') # отрисовка линии-закрепа спирали

sh, = ax.plot(X\_Sh[0], Y\_Sh[0], marker='o', markersize = 5, color = 'orange') # отрисовка шарнира

st, = ax.plot([X\_Sh[0], X\_Gr[0]], [Y\_Sh[0], Y\_Gr[0]], color = 'orange') # отрисовка стержня

gr, = ax.plot(X\_Gr[0], Y\_Gr[0], marker = 'o', markersize = 20, color = 'orange') # отрисовка грузика

def anima(i): # функция анимации

thetta = np.linspace(0, 3/2\*math.pi+psi[i], 100)

X\_SpiralSpr = (R1 + thetta\*(R2-R1)/thetta[-1])\*np.cos(thetta)

Y\_SpiralSpr = -(R1 + thetta\*(R2-R1)/thetta[-1])\*np.sin(thetta)

spr.set\_data(X\_SpiralSpr+X\_Sh[i], Y\_SpiralSpr+Y\_Sh[i])

pl1.set\_data([X\_Sh[i]+R1-WIDE/2, X\_Sh[i]+R1-WIDE/2+X\_DSHT], [Y\_Sh[i]+LEN, Y\_Sh[i]+LEN+Y\_DSHT])

pl2.set\_data([X\_Sh[i]+R1, X\_Sh[i]+R1+X\_DSHT], [Y\_Sh[i]+LEN, Y\_Sh[i]+LEN+Y\_DSHT])

pl3.set\_data([X\_Sh[i]+R1+WIDE/2, X\_Sh[i]+R1+WIDE/2+X\_DSHT], [Y\_Sh[i]+LEN, Y\_Sh[i]+LEN+Y\_DSHT])

hl.set\_data([X\_Sh[i]+R1-WIDE/2-0.05, X\_Sh[i]+R1+WIDE/2+0.05], [Y\_Sh[i]+LEN, Y\_Sh[i]+LEN])

upl.set\_data([X\_Sh[i]+R1-0.0015, X\_Sh[i]+R1-0.0015], [Y\_Sh[i], Y\_Sh[i]+LEN])

sh.set\_data(X\_Sh[i], Y\_Sh[i])

st.set\_data([X\_Sh[i], X\_Gr[i]], [Y\_Sh[i], Y\_Gr[i]])

gr.set\_data(X\_Gr[i], Y\_Gr[i])

return spr, hl, upl, sh, st, gr

anim = FuncAnimation(fig, anima, frames=Steps, interval=50, repeat=False) # создаём разовую анимацию

fig.suptitle('Kutsenko LW2', fontsize=14) # добавляем название

anim.save("./Animation.mp4", writer="ffmpeg") # сохраняем анимацию

*Процесс выполнения работы:*

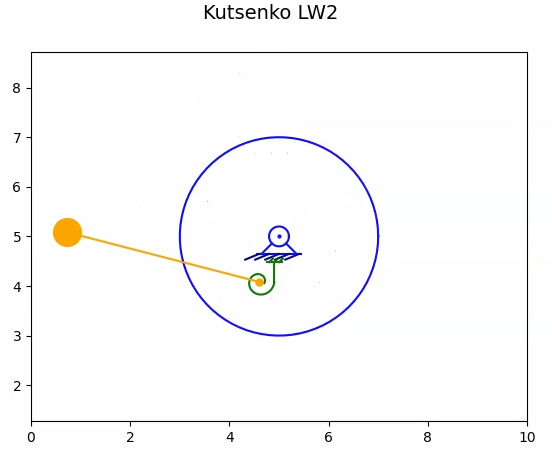
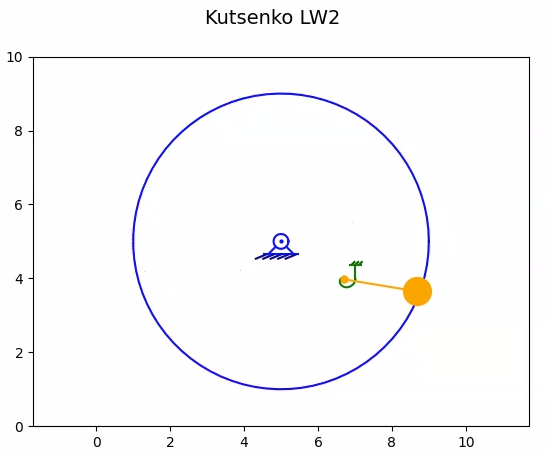
Данная лабораторная работа выполнялась мной поэтапно.

Сначала я построил статичный диск и его крепление. Затем я построил шарнир, находя его координаты относительно центра диска с использованием параметра **a** иугла поворота **φ** шарнира против часовой стрелки относительно вертикали, проходящей через центр диска. Используя координаты шарнира, длину стержня **l** и угол поворота **ψ** груза по часовой стрелке относительно вертикали, проходящей через шарнир,я построил стержень и груз.

Далее, используя координаты шарнира, я построил вокруг него спиральную пружину, прикреплённую своими концами к диску и шарниру, и закручивающуюся в сторону шарнира на 1.5π+**ψ**.

В заключение я переделал отрисовку в динамическую, рассчитав позиции шарнира, груза для моментов времени заранее и использовав функцию анимации.

*Результат работы программы:*



На данных изображениях можно увидеть анимацию системы при двух различных значениях радиуса диска, длины стержня и расстояния до его центра. Зависимость углов поворота от времени в данном случае произвольна.

*Вывод:* научился использованию средств языка Python для моделирования сложного движения элементов системы. Вместо произвольных зависимостей теперь можно вставить закон движения, чтобы увидеть реальное движение заданной системы.