ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ОТЧЕТ

О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«ДИНАМИКА МЕХАНИЧЕСКОЙ СИСТЕМЫ»

ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»

ВАРИАНТ ЗАДАНИЯ 1

Выполнил(а) студент группы М8О-206Б-22

Жаднов Михаил Денисович_____
подпись, дата

Проверил и принял

Зав. каф. 802, Бардин Б.С._____
подпись, дата

с оценкой _____

Москва, 2023

Задание: Реализовать анимацию движения системы заданой таким образом. Круглая трубка, имеющая форму колеса радиуса R, катится без проскальзывания в вертикальной плоскости по неподвижной горизонтальной направляющей. В трубке находится шарик B массы m. Масса колеса равномерно распределена по его ободу и равна M.

Код:

```
import matplotlib.pyplot as p
from matplotlib.animation import FuncAnimation
import numpy as n

T = n.linspace(0, 200, 1000)
Psi = 0.07*T*n.pi
Phi = ((n.pi / 3) * n.cos(0.5*T))

fgr = p.figure()
plt = fgr.add_subplot(1,1,1)
plt.axis('equal')

m = 2
M = 1
R = 1
c = 40

# r is imaginable, i suppose
r = 0.1 # радиус шарика

plt.plot([-R, R],[0,0]) # OX axis
# plt.plot([0, 0],[0, 2*R]) # OY axis (mid)

# for ball
Alph = n.linspace(0, 2*n.pi, 1000)
Yball = r*n.cos(Alph)
Xball = r*n.sin(Alph)

# for ball way
Beth = n.linspace(-n.pi, 0, 1000)
Xc = R*n.cos(Beth)
Yc = R*n.sin(Beth) + R

Xsp_b = R*n.cos(Alph)
Ysp_b = R*n.sin(Alph) + R
Xsp = (R - 2*r)*n.cos(Alph)
Ysp = (R - 2*r)*n.sin(Alph) + R

# A point
Xa = R*n.sin(Psi[0])
Ya = R*n.cos(Psi[0]) + R
A = plt.plot(Xa, Ya, marker = 'o')[0]
```

```
# B point ~ Ball
Xb = (R-r)*n.sin(Phi[0])
Yb = -(R-r)*n.cos(Phi[0]) + R # нижняя полуокружность
Ball = plt.plot(Xb + Xball, Yb + Yball)[0]

# получаем пружину от точки start до end
def get_spring(coils, width, start, end):
start, end = n.array(start).reshape((2,)), n.array(end).reshape((2,))
len = n.linalg.norm(n.subtract(end, start))
u_t = n.subtract(end, start) / len
u_n = n.array([[0, -1], [1, 0]]).dot(u_t)
spring_coords = n.zeros((2, coils + 2))
spring_coords[:,0], spring_coords[:,-1] = start, end
normal_dist = n.sqrt(max(0, width ** 2 - (len ** 2 / coils ** 2))) / 2
for i in n.arange(1, coils + 1):
spring_coords[:,-i] = (start
+ ((len * (2 * i - 1) * u_t) / (2 * coils))
+ (normal_dist * (-1) ** i * u_n))
return spring_coords[0,2:], spring_coords[1,2:]

pS = plt.plot(*get_spring(70, 0.1, [Xa, Ya], [Xb, Yb]), color='black')[0]

Sphere_b = plt.plot(Xsp_b, Ysp_b)[0]
Sphere = plt.plot(Xsp, Ysp)[0]

def run(i):
Xa = R*n.sin(Psi[i])
Ya = R*n.cos(Psi[i]) + R
Xb = (R-r)*n.sin(Phi[i])
Yb = -(R-r)*n.cos(Phi[i]) + R
Ball.set_data((Xb + Xball), (Yb + Yball))
A.set_data(Xa, Ya)
# AB.set_data([Xa, Xb], [Ya, Yb])
# Way.set_data(Xc, Yc)
Sphere_b.set_data(Xsp_b, Ysp_b)
Sphere.set_data(Xsp, Ysp)

pS.set_data(*get_spring(35, 0.06, [Xa, Ya], [Xb, Yb]))

return
anim = FuncAnimation(fgr, run, frames=len(T), interval = 1)

fgr.show()

quit = input()
```

Протокол тестирования