ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ ТОЧКИ»**

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

**ВАРИАНТ ЗАДАНИЯ №21**

Выполнил(а) студент группы  М80-201Б-22

Парфенов Михаил Максимович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

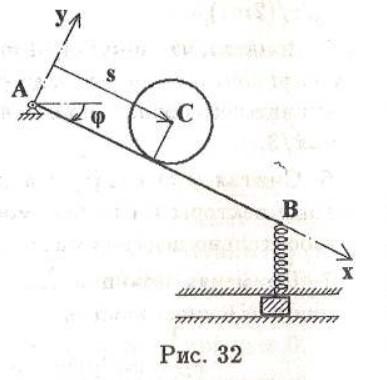
подпись, дата

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.

**Механическая система:**



**Текст программы:**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from scipy.integrate import odeint

def odesys(y, t, P1, P2, l, r, phi\_, g): # функция системы диффуров dy = np.zeros(4)

dy[0] = y[2]

dy[1] = y[3]

* Правило Краммера
* a11 \* s'' + a12 \* phi'' = b1
* a21 \* s'' + a22 \* phi'' = b2
* detA = a11 \* a22 - a12 \* a21
* detA1 = b1 \* a22 - a12 \* b2
* detA2 = a11 \* b2 - b2 \* a21

a11 = 3 \* P2 \* r / 2

a12 = ((P1 \* l\*\*2) / 3) + P2\*y[0]\*\*2 + ((3 \* P2 \* r\*\*2) / 2)

a21 = 3 / 2

a22 = (3 \* r) / 2

b1 = (P1 / 2) \* g \* l \* (1 - (np.sin(y[1]) / np.sin(phi\_))) \* np.cos(y[1]) + P2 \* g \* (y[0] \* np.cos(y[1]) + r \* np.sin(y[1])) - 2 \* P2 \* y[0] \* y[2] \* y[3]

b2 = g \* np.sin(y[1]) + y[0] \* y[3]\*\*2

* dy[2] = detA1 / detA
* dy[3] = detA2 / detA

dy[2] = (b1\*a22 - b2\*a12)/(a11\*a22 - a12\*a21)

dy[3] = (b2\*a11 - b1\*a21)/(a11\*a22 - a12\*a21)

return dy

#ЗАДАЕМ НАЧАЛЬНЫЕ ЗНАЧЕНИЯ!!!

P1 = 150 # вес балки

P2 = 50 # вес диска

l = 10 # длинная балки

r = 0.5 # радиус диска

g = 9.8

phi\_ = np.pi/36

s0 = 0

phi0 = np.pi/18

ds = 0

dphi0 = 0

t\_fin = 15 #сетка времени

t=np.linspace(0,t\_fin,501)

y0 = [s0, phi0, ds, dphi0]

# интегрируем систему диффуров

Y = odeint(odesys, y0, t, (P1, P2, l, r, phi\_, g))

s = Y[:, 0] # получили решения

phi = Y[:, 1]

ds = Y[:, 2]

dphi = Y[:, 3]

# 2 производная по phi

ddphi = np.array([odesys(yi, ti, P1, P2, l, r, phi\_, g)[3] for yi,ti in zip(Y,t)])

# давелние N диска на балку

N = (P2 / g) \* (g \* np.cos(phi) - (2 \* ds + r \* dphi) \* dphi - s \* ddphi)

fig\_for\_graphs = plt.figure(figsize=[13,7]) # построим их графики

ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2,3,1)

ax\_for\_graphs.plot(t,s,color='blue')

ax\_for\_graphs.set\_title("s(t)")

ax\_for\_graphs.set(xlim=[0,t\_fin])

ax\_for\_graphs.grid(True)

ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2,3,2)

ax\_for\_graphs.plot(t,phi,color='red')

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(2,3,3)

ax\_for\_graphs.plot(t,N,color='blue')

ax\_for\_graphs.set\_title('N(t)')

ax\_for\_graphs.set(xlim=[0,t\_fin])

ax\_for\_graphs.grid(True)

xA = 0 #точки крепления шарнира и балки

yA = 5

xB = l #точки крепления пружины и балки

yB = 0

l0 = 5 #длина недеформированной пружины в начальном состоянии с учетом phi\_

n = 13

h = 0.05

xP = np.zeros(2\*n + 1)

yP = np.linspace(0, 1, 2\*n + 1)

ss = 0

* делаем пружину пружиной for i in range(2\*n+1):

xP[i] = h\*np.sin(ss) ss += np.pi/2

yB = l0 - l \* np.sin(phi)

#prizuna

fig = plt.figure(figsize=[15,7])

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

ax.axis('equal')

ax.set(xlim=[-3,12], ylim=[-3,12])

Pruzzhina = ax.plot(xP + xB, yP\*(yB[0] + l0), color='black')[0]

#enviroment

X\_Ground = [0, 0, 12]

Y\_Ground = [9, 0, 0]

ax.plot(X\_Ground,Y\_Ground,color='black',linewidth=3)

ax.plot([0], [5], '-ro', markersize=10, color='blue')

#балка

Stick = ax.plot([xA, xB], [yA, yB[0]], linewidth=7, color='brown')[0]

#точка крепления пружины и балки

B = ax.plot(xB, yB[0], 'o',color=[1,0,0])[0]

* диск по точкам, используя уравнение окружности def disk(x, y, r):

cx = [x + r \* np.sin(i / 100) for i in range(0, 628)] cy = [y + r \* np.cos(i / 100) for i in range(0, 628)] return (cx, cy)

x\_disk = s \* np.cos(phi)

* l - s - r \* np.tan(phi) - вертикальное расстояние от точки B до центра диска
* sin(phi) - учитывает горизонтальное перемещение диска в результате поворота балки.
* r/cos(phi) - компенсация вертикального перемещения диска вдоль наклоненной балки, что учитывает наклон балки.

y\_disk = (l - s - r \* np.tan(phi)) \* np.sin(phi) + r / np.cos(phi)

disk\_ = ax.plot(disk(x\_disk[0], y\_disk[0] + yB[0], r)[0], disk(x\_disk[0], y\_disk[0] + yB[0], r)[1], 'green')[0]

def kadr(i):

B.set\_data(xB, np.abs(yB[i]))

Pruzzhina.set\_data(xP + xB, np.abs(yP\*(yB[i])))

Stick.set\_data([xA, xB], [yA, np.abs(yB[i])])

disk\_.set\_data(disk(x\_disk[i], y\_disk[i] + yB[i], r)[0], disk(x\_disk[i], y\_disk[i] + yB[i], r)[1])

return [B, Pruzzhina, Stick, disk]

kino = FuncAnimation(fig,kadr,interval = 10,frames=len(t))

plt.show()

**Результат работы:**

