МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные технологии и прикладная математика»
Кафедра 802

Лабораторная работа
по дисциплине Теоретическая Механика
По теме:
«Динамика Механических систем»

Вариант: №20

Исполнитель: Шерматов Егор Подпись:

Группа: М80-203Б-21

Руководитель работы: Беличенко М.В.

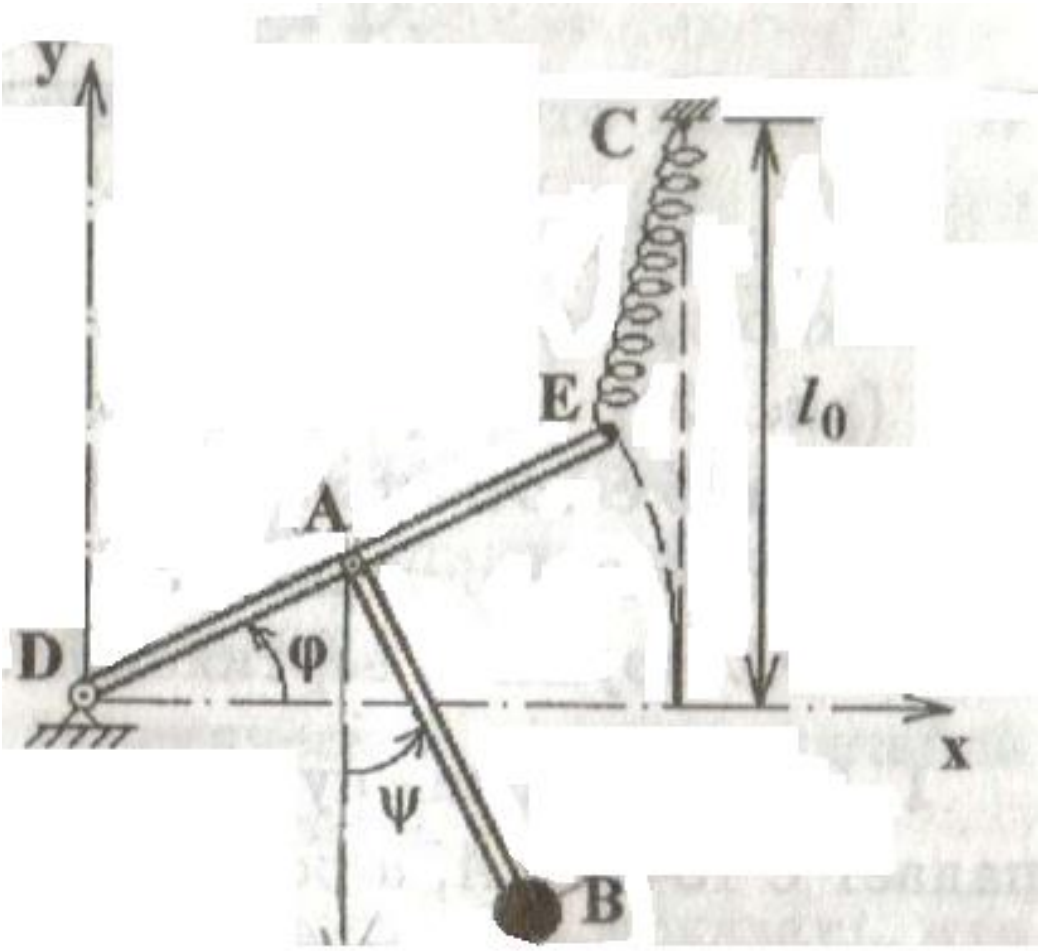
Оценка:

Дата:

Подпись преподавателя

Москва 2023

Задание: Однородная балка *DE* длины *2a* и массы *m1* закреплена в неподвижном шарнире *D* и концом *E* соединена с пружиной жесткости *c*. К середине балки прикреплен невесомый стержень *AB* длины *b* с точечным грузом массы *m2* на конце *B*. Длина недеформированной пружины равно *lo*; при горизонтальном положении пружина вертикальна.



Задание 12: Задавая численные значения параметров и начальные условия:
m1 = 50 кг, m2 = 0.5 кг, a = b = lo = 1 м, c = 250 Н/м, t0 = 0,
phie0 = 0, psie0 = (pi/18), d(phie0)/dt = d(psie0)/dt = 0,
составить программу решения дифференциальных уравнений и на ЭВМ построить зависимости *phie(t), psie(t), N(t)*

Код программы:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from scipy.integrate import odeint

#region Calculation of coord.
def rotate(origin, point, angle):
    x = origin[0]
    y = origin[1]
    px = point[0]
    py = point[1]
    nx = x + np.cos(angle) * (px - x) - np.sin(angle) * (py - y)
    ny = y + np.sin(angle) * (px - x) + np.cos(angle) * (py - y)
    return np.array([nx, ny])

def distance(x1, x2, y1, y2):
    return ((x1 - x2)**2 + (y1 - y2)**2)**0.5

def EqOfMovement(y,t,m1,m2,a,b,l0,c,g):

    #  y[0,1,2,3] = phi,psi,phi',psi'
    # dy[0,1,2,3] = phi',psi',phi'',psi''

    dy = np.zeros_like(y)
    dy[0] = y[2]; dy[1] = y[3];
    l = np.sqrt(8*(a**2)*(1 - np.cos(y[0])) + l0*(l0 - 4*a*np.sin(y[0])))

    a11 = a*((4/3)*m1 + m2); a12 = m2*b*np.sin(y[1]-y[0])
    b1 = (-1)*(m1+m2)*g*np.cos(y[0]) + c*((l0/l) - 1)*(4*a*np.sin(y[0]) - 2*l0*np.cos(y[0])) -
m2*b*np.cos(y[1]-y[0])*(y[1]**2)

    a21 = a*np.sin(y[1]-y[0]); a22 = b
    b2 = (-1)*g*np.sin(y[1]) + a*np.cos(y[1]-y[0])*(y[0]**2)

    dy[2] = (b1*a22 - b2*a12)/(a11*a22 - a21*a12)
    dy[3] = (a11*b2 - a21*b1)/(a11*a22 - a21*a12)

    return dy

t0 = 0;
y0 = [0,(np.pi)/18,0,0]
#y0 = [(np.pi/3),(np.pi/4),0,0]
t_fin = 20; Nt = 2001
```

```
t = np.linspace(t0, t_fin, Nt)  #time grid

#          (m1,m2,a,b,l0,c,g) - start params
#t0 = 0; y0 = [0,(np.pi)/18,0,0];
m1 = 50; m2 = 0.5; a, b, l0 = 1,1,1; c = 250; g = 9.8
#params_0 = (m1,m2,a,b,l0,c,g)
#params_0 = (10, 9, 1.5, 2, 0.5, 25, g)
#params_0 = (1, 50, 1.5, 0.5, 1, 100, g)
params_0 = (5, 6, 1, 1, 1, 300, g)

Y = odeint(EqOfMovement, y0, t ,params_0)

phi = Y[:, 0]; psi = Y[:, 1]; dphi = Y[:, 2]; dpsi = Y[:, 3]
ddphi = np.array([EqOfMovement(yi,ti,m1,m2,a,b,l0,c,g)[2] for yi,ti in zip(Y,t)])
#endregion

#region Animation
#    (m1,m2,a,b,l0,c,g) - start params
x0, y0 = 0, 0
xD, yD = x0, y0
xA, yA = (x0 + a*np.cos(phi)), (y0 + a*np.sin(phi))
xE, yE = (x0 + 2*a*np.cos(phi)), (y0 + 2*a*np.sin(phi))
#xB, yB = (x0 + a*np.cos(phi) + b*np.sin(psi)), (a*np.sin(phi) - y0 +b*np.cos(psi))
xB, yB = xA + b*np.sin(psi), yA - b*np.cos(psi)
xC, yC = x0 + 2*a, y0 + l0

fig = plt.figure(figsize=[13,9])
ax = fig.add_subplot(1,1,1)
ax.axis('equal')
ax.set(xlim=[-5,5],ylim=[-4,4])

#spring = ax.plot([xE[0],xC], [yE[0],yC], color='green')[0]  #затычка
n = 16; h = 0.05
xSpringARR = [] #x of spring
ySpringARR = [] #y of spring
for i in range(Nt):
    spX = np.linspace(0, distance(xC, xE[i], yC, yE[i]), 2*n+1)  # correct length spring
    spY = np.zeros(2*n+1)
    ss = 0
    #length of spring
    for j in range(2*n+1):
        spY[j] = h*np.sin(ss)
        ss += 3.14 / 2
    #rotation of spring
    for k in range(2*n+1):
        if yE[i] > yC and xE[i] < xC:
            spX[k], spY[k] = rotate([0, 0], [spX[k], spY[k]], np.pi - np.math.atan(abs(yE[i]-yC)/abs(xC-xE[i])))
        elif yE[i] > yC and xE[i] > xC: #=
            spX[k], spY[k] = rotate([0, 0], [spX[k], spY[k]], np.math.atan(abs(yE[i] - yC) / abs(xE[i] - xC)))
        elif yE[i] < yC and xE[i] > xC:
            spX[k], spY[k] = rotate([0, 0], [spX[k], spY[k]], -np.math.atan(abs(yC - yE[i]) / abs(xE[i] - xC)))
```

```
        elif yE[i] < yC and xE[i] < xC: #=
            spX[k], spY[k] = rotate([0, 0], [spX[k], spY[k]], np.pi + np.math.atan(abs(yC - yE[i]) / abs(xC - xE[i])))
        elif yE[i] == yC:
            if xC <= xE[i]:
                spX[k], spY[k] = rotate([0, 0], [spX[k], spY[k]], 0)
            else:
                spX[k], spY[k] = rotate([0, 0], [spX[k], spY[k]], np.pi)
        elif xE[i] == xC:
            if yE[i] >= yC:
                spX[k], spY[k] = rotate([0, 0], [spX[k], spY[k]], np.pi/2)
            else:
                spX[k], spY[k] = rotate([0, 0], [spX[k], spY[k]], -np.pi/2)
    #add coord to array of spring coord
    for k in range(2*n+1):
        spX[k] += xC
        spY[k] += yC
    xSpringARR.append(spX)
    ySpringARR.append(spY)

spring = ax.plot(xSpringARR[0], ySpringARR[0], color=[0,0,0])[0]


wall_vertical = ax.plot([0, 0], [0, 3], color='blue', linewidth = 1)
wall_horizontal = ax.plot([0, 3], [0, 0], color='blue', linewidth = 1)

DE = ax.plot([xD,xE[0]], [yD,yE[0]], color='red', linewidth = 4)[0]
AB = ax.plot([xA[0],xB[0]], [yA[0],yB[0]], color='black')[0]

D = ax.plot(xD, yD, 'o', color='red')[0]
A = ax.plot(xA[0], yA[0], 'o', color='red')[0]
E = ax.plot(xE[0], yE[0], 'o', color='red')[0]
B = ax.plot(xB[0], yB[0], 'o', color='black')[0]
C = ax.plot(xC, yC, 'o', color='green')[0]

def kadr(i):
    D.set_data(xD,yD)
    A.set_data(xA[i],yA[i])
    E.set_data(xE[i],yE[i])
    B.set_data(xB[i],yB[i])
    C.set_data(xC,yC)

    DE.set_data([xD,xE[i]], [yD,yE[i]])
    AB.set_data([xA[i],xB[i]], [yA[i],yB[i]])

    spring.set_data(xSpringARR[i], ySpringARR[i])


    return [D, A, E, B, C, DE, AB, spring]

kino = FuncAnimation(fig, kadr, interval = t[1]-t[2], frames=len(t))
```

```
N_A = m2*(g*np.cos(psi) + b*(dpsi**2) + a*(ddphi*np.cos(psi-phi) + (dphi**2)*np.sin(psi-phi)));
#endregion

#region Graphs of Coord.
fig0 = plt.figure(figsize=[13,9])

ax1 = fig0.add_subplot(2,2,1)
ax1.plot(t,phi,color=[1,0,0])
ax1.set_title('Phie(t)')

ax2 = fig0.add_subplot(2,2,2)
ax2.plot(t,psi,color=[0,1,0])
ax2.set_title('Psie(t)')

ax3 = fig0.add_subplot(2,2,3)
ax3.plot(t,dphi,color=[0,0,1])
ax3.set_title('dPhie(t)')

ax4 = fig0.add_subplot(2,2,4)
ax4.plot(t,dpsi,color=[0,0,0])
ax4.set_title('dPsie(t)')

fig4R_A =plt.figure(figsize=[13,9])
ax5 = fig4R_A.add_subplot(2,2,1)
ax5.plot(t,N_A,color=[0,0,0])
ax5.set_title('N')

plt.show()
#endregion
```

Ссылка на открытый репозиторий github:
https://github.com/HaPPyDutCHoGGG/termech

Результаты работы

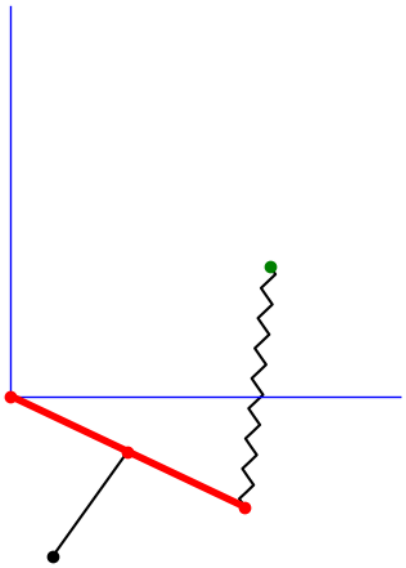
1) параметры:

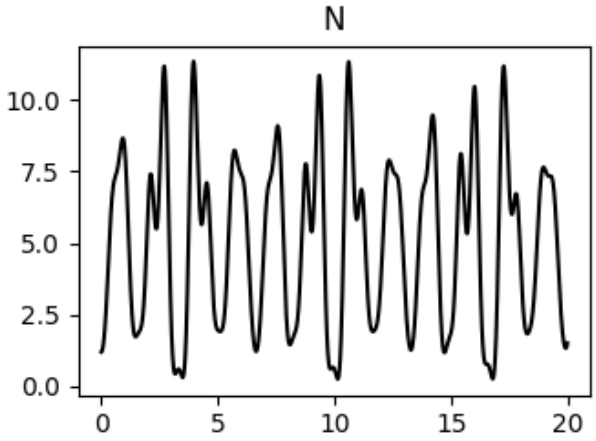
m1 = 50 кг, m2 = 0.5 кг, a = b = lo = 1 м, c = 250 Н/м.

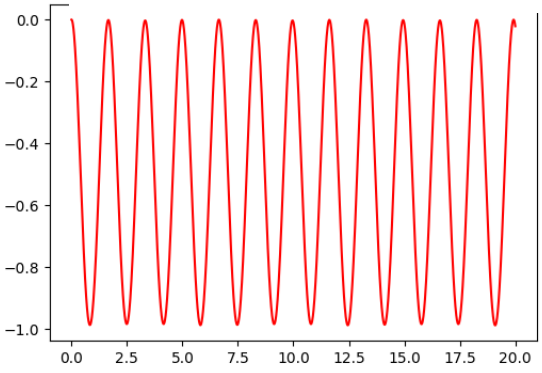
y0 = [0,(np.pi)/18,0,0]

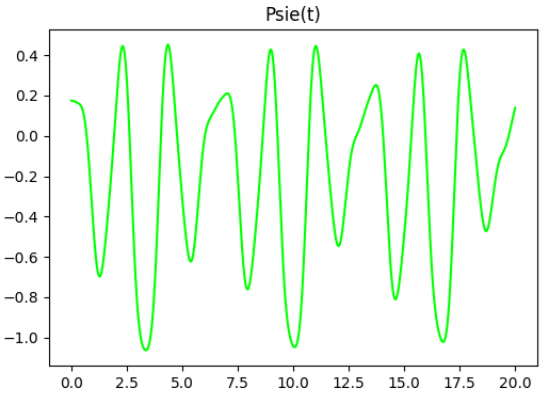
Результат:

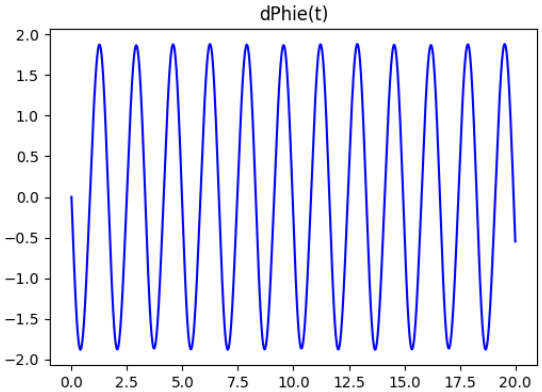
балка *DE(красная линия)* колеблется под синей горизонтальной осью, достигая ее справа от вертикальной, но не пересекая. Груз *B(черная точка на конце черного отрезка AB)*, прикрепленный к центру балки шарниром с невесомым стержнем также совершает колебания с отклонением *psie* меньше 90 градусов. Снизу представлена работа системы и графики зависимостей

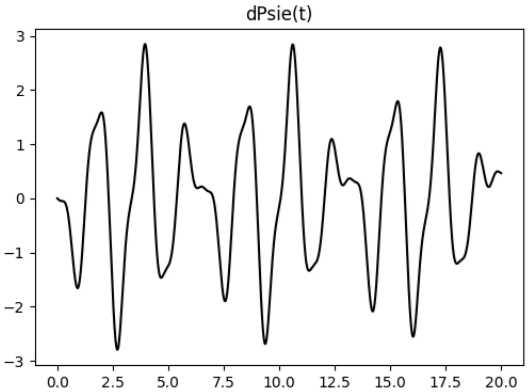




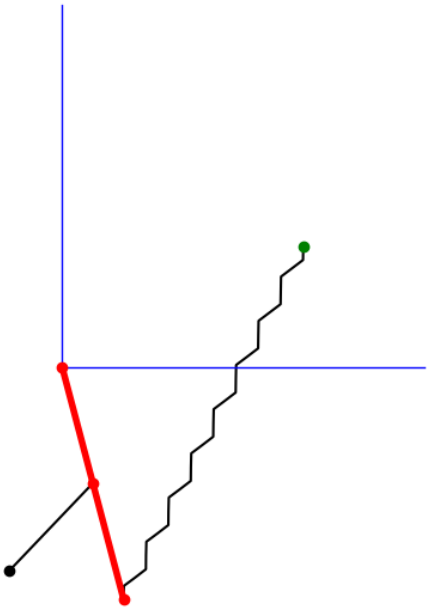


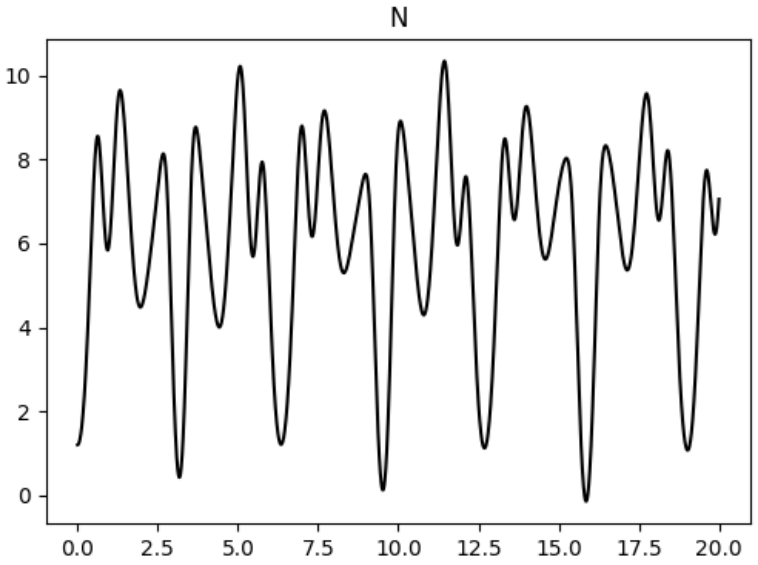


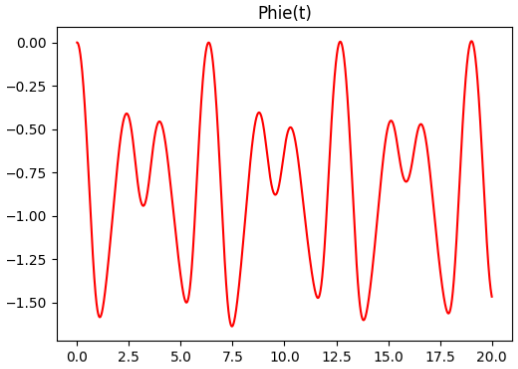


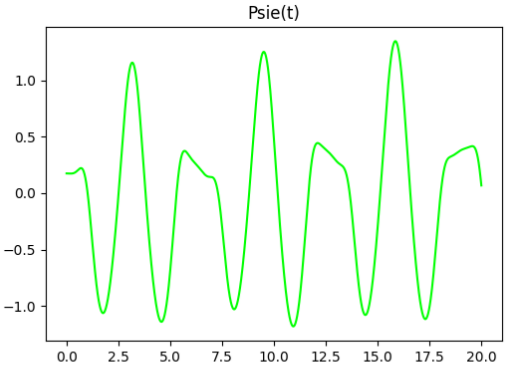


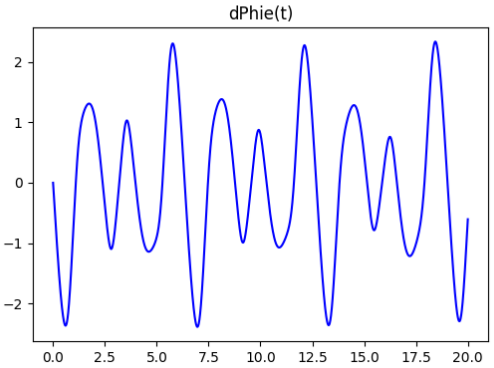
2) параметры:
m1 = 10 кг, m2 = 9 кг, a = 1.5, b = 2, lo = 0.5 м, c = 25 Н/м.
y0 = [0,(np.pi)/18,0,0]
Результат: балка *DE(красная линия)* колеблется под синей горизонтальной осью, достигая вертикального нижнего положения, но не отклоняясь далее. Груз *B(черная точка на конце черного отрезка AB)*, прикрепленный к центру балки шарниром с невесомым стержнем также совершает колебания с отклонением *psie* менее 90 градусов. Снизу представлена работа системы и графики зависимостей

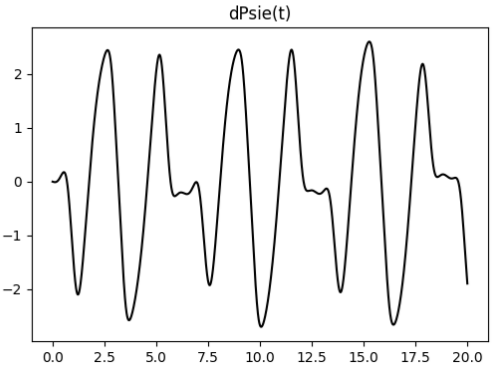




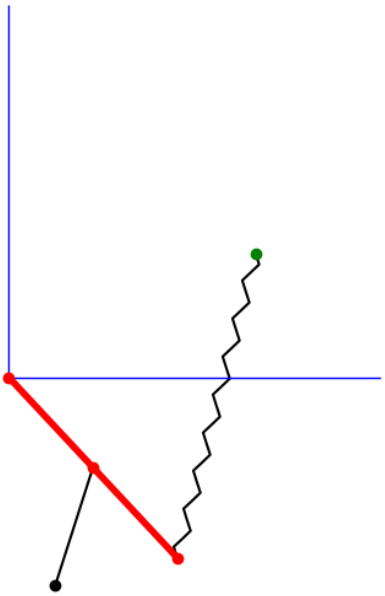


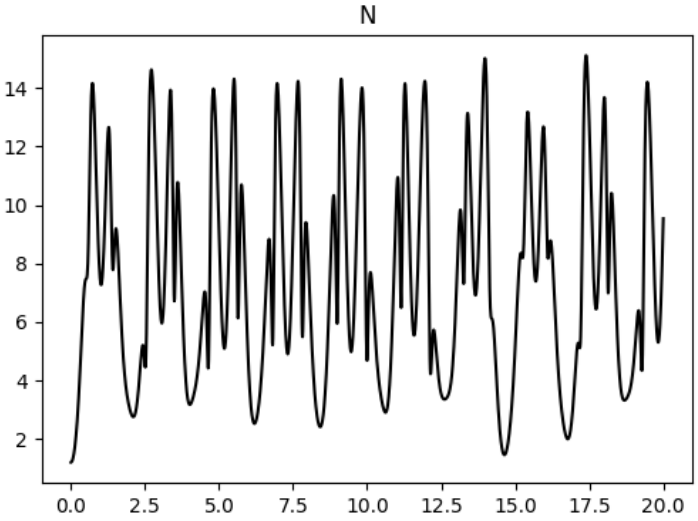


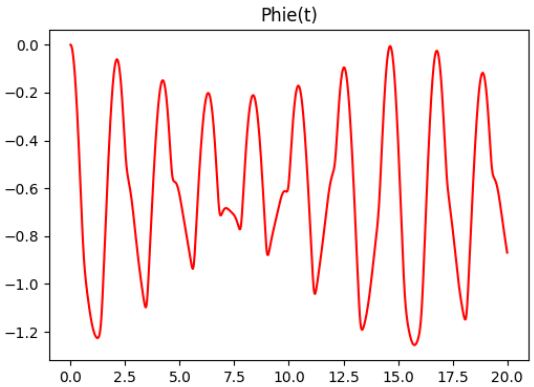


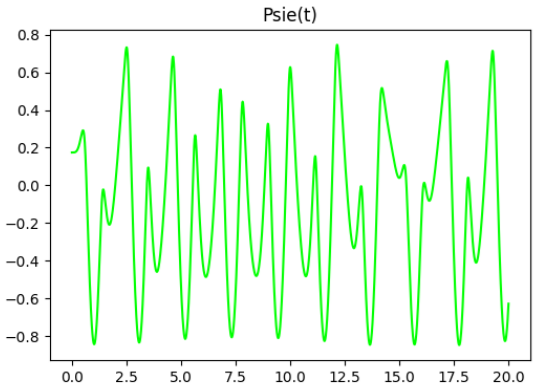


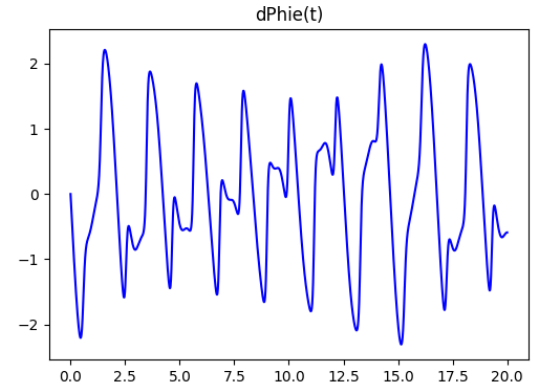
3) параметры:
m1 = 1 кг, m2 = 50 кг, a = 1.5, b = 0.5, lo = 1 м, c = 100 Н/м.
y0 = [0,(np.pi)/18,0,0]
Результат: балка *DE(красная линия)* колеблется под синей горизонтальной осью. Груз *B(черная точка на конце черного отрезка AB)*, прикрепленный к центру балки шарниром с невесомым стержнем также совершает колебания с отклонением *psie* менее 90 градусов, но из-за большего веса он либо тянет балку вверх за собой, либо вниз, поэтому балка не достигает ни нижнего вертикального, ни горизонтального положения. Снизу представлена работа системы и графики зависимостей

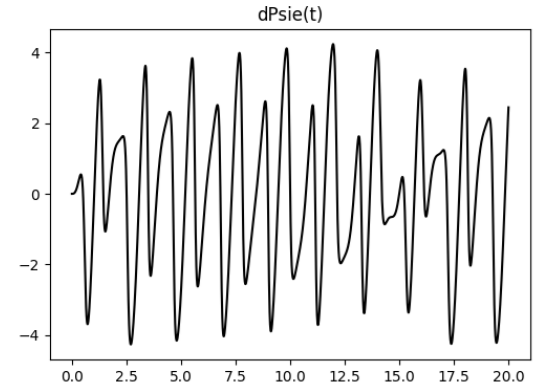












4) параметры:
m1 = 5 кг, m2 = 6 кг, a = b = lo = 1 м, c = 300 Н/м.
y0 = [0,(np.pi)/18,0,0]
Результат: балка *DE(красная линия)* колеблется под синей горизонтальной осью, не достигая ее, с малой амплитудой. Груз *B(черная точка на конце черного отрезка AB)*, прикрепленный к центру балки шарниром с невесомым стержнем также совершает малые колебания около нижнего положения равновесия. Снизу представлена работа системы и графики зависимостей