ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«ДИНАМИКА СИСТЕМЫ»**

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

**ВАРИАНТ ЗАДАНИЯ № 24**

Выполнил(а) студент группы М8О-209Б-23

Стрепетов И.О.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Ст. преп. каф. 802, Волков Е.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

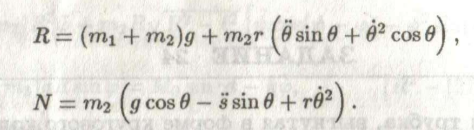
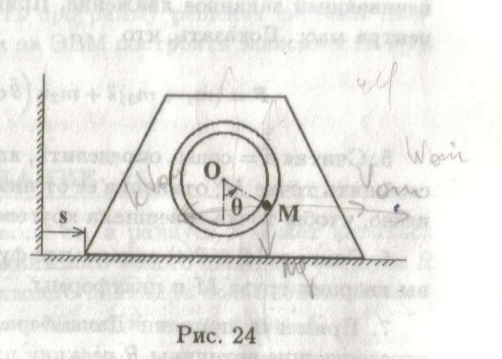
подпись, дата

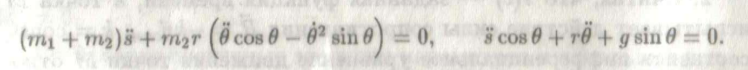
с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

*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.

**Вариант 24:**





**Код программы:**

import numpy as np

import sympy as sp

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

import math

from scipy.integrate import odeint

t\_fin = 10

t = np.linspace(0, t\_fin, 1000)

m1 = 20

m2 = 20

r = 1

g = 9.81

def odesys(y, t, m1, m2, r, g):

dy = np.zeros(4)

dy[0] = y[2]

dy[1] = y[3]

a11 = m1 + m2

a12 = m2 \* r \* np.cos(y[1])

a21 = np.cos(y[1])

a22 = r

b1 = m2 \* r \* np.sin(y[1]) \* (y[3] 2)

b2 = -g \* np.sin(y[1])

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

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

return dy

s0 = 0

phi0 = 1

ds0 = 0

dphi0 = 2

y0 = [s0, phi0, ds0, dphi0]

Y = odeint(odesys, y0, t, (m1, m2, r, g))

s = Y[:, 0]

phi = Y[:, 1]

ds = Y[:, 2]

dphi = Y[:, 3]

dds = [odesys(y, t\_i, m1, m2, r, g)[2] for y, t\_i in zip(Y, t)]

ddphi = [odesys(y, t\_i, m1, m2, r, g)[3] for y, t\_i in zip(Y, t)]

R\_2 = (m1 + m2) \* g + m2 \* r \* (np.array(ddphi) \* np.sin(phi) + dphi 2 \* np.cos(phi))

N = m2 \* (g \* np.cos(phi) - np.array(dds) \* np.sin(phi) + r \* dphi \*\* 2)

O = s

XA = 2.5 + s

YA = 1.5

XB = XA + r \* np.sin(phi)

YB = YA - r \* np.cos(phi)

Xtr = np.array([1, 1.5, 3.5, 4, 1])

Ytr = np.array([0, 3, 3, 0, 0])

# Анимация

fig = plt.figure(figsize=[1, 1])

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

ax.set(xlim=[-2, 10], ylim=[-1, 5])

ax.set\_aspect('equal')

X\_Ground = [-1, -1, 9]

Y\_Ground = [4, 0, 0]

ax.plot(X\_Ground, Y\_Ground, color='Black')

Trap = ax.plot(O[0] + Xtr, Ytr)[0]

Rad = ax.plot([XA[0], XB[0]], [YA, YB[0]])[0]

Point\_A = ax.plot(XA[0], YA, marker='o')[0]

angles = np.linspace(0, 2 \* math.pi, 1000)

R = 0.1

Point\_B = ax.plot(XB[0] + R \* np.cos(angles), YB[0] + R \* np.sin(angles))[0]

X\_Circle = (r - R) \* np.cos(angles)

Y\_Circle = (r - R) \* np.sin(angles)

Drawed\_Circle = ax.plot(XA[0] + X\_Circle, YA + Y\_Circle, 'green')[0]

X\_Circle2 = (R + r) \* np.cos(angles)

Y\_Circle2 = (R + r) \* np.sin(angles)

Drawed\_Circle2 = ax.plot(XA[0] + X\_Circle2, YA + Y\_Circle2, 'green')[0]

ax\_for\_graphs = fig.add\_subplot(2, 2, 4)

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

ax\_for\_graphs.plot(t, s)

ax\_for\_graphs.grid(True)

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

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

ax\_for\_graphs.plot(t, phi)

ax\_for\_graphs.grid(True)

def anima(i):

Point\_A.set\_data(XA[i], YA)

Rad.set\_data([XA[i], XB[i]], [YA, YB[i]])

Trap.set\_data(O[i] + Xtr, Ytr)

Point\_B.set\_data(XB[i] + R \* np.cos(angles), YB[i] + R \* np.sin(angles))

Drawed\_Circle.set\_data(XA[i] + X\_Circle, YA + Y\_Circle)

Drawed\_Circle2.set\_data(XA[i] + X\_Circle2, YA + Y\_Circle2)

return Point\_A, Rad, Point\_B, Trap, Drawed\_Circle, Drawed\_Circle2

anim = FuncAnimation(fig, anima, frames=1000, interval=0.01, blit=True)

# Второе окно с графиками

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

ax2 = fig\_for\_graphs.add\_subplot(2, 2, 1)

ax2.set\_title("N(t)")

ax2.plot(t, N)

ax2.grid(True)

ax2 = fig\_for\_graphs.add\_subplot(2, 2, 3)

ax2.set\_title("R(t)")

ax2.plot(t, R\_2)

ax2.grid(True)

ax2 = fig\_for\_graphs.add\_subplot(2, 2, 2)

ax2.set\_title("dds(t)")

ax2.plot(t, dds)

ax2.grid(True)

ax2 = fig\_for\_graphs.add\_subplot(2, 2, 4)

ax2.set\_title("ddphi(t)")

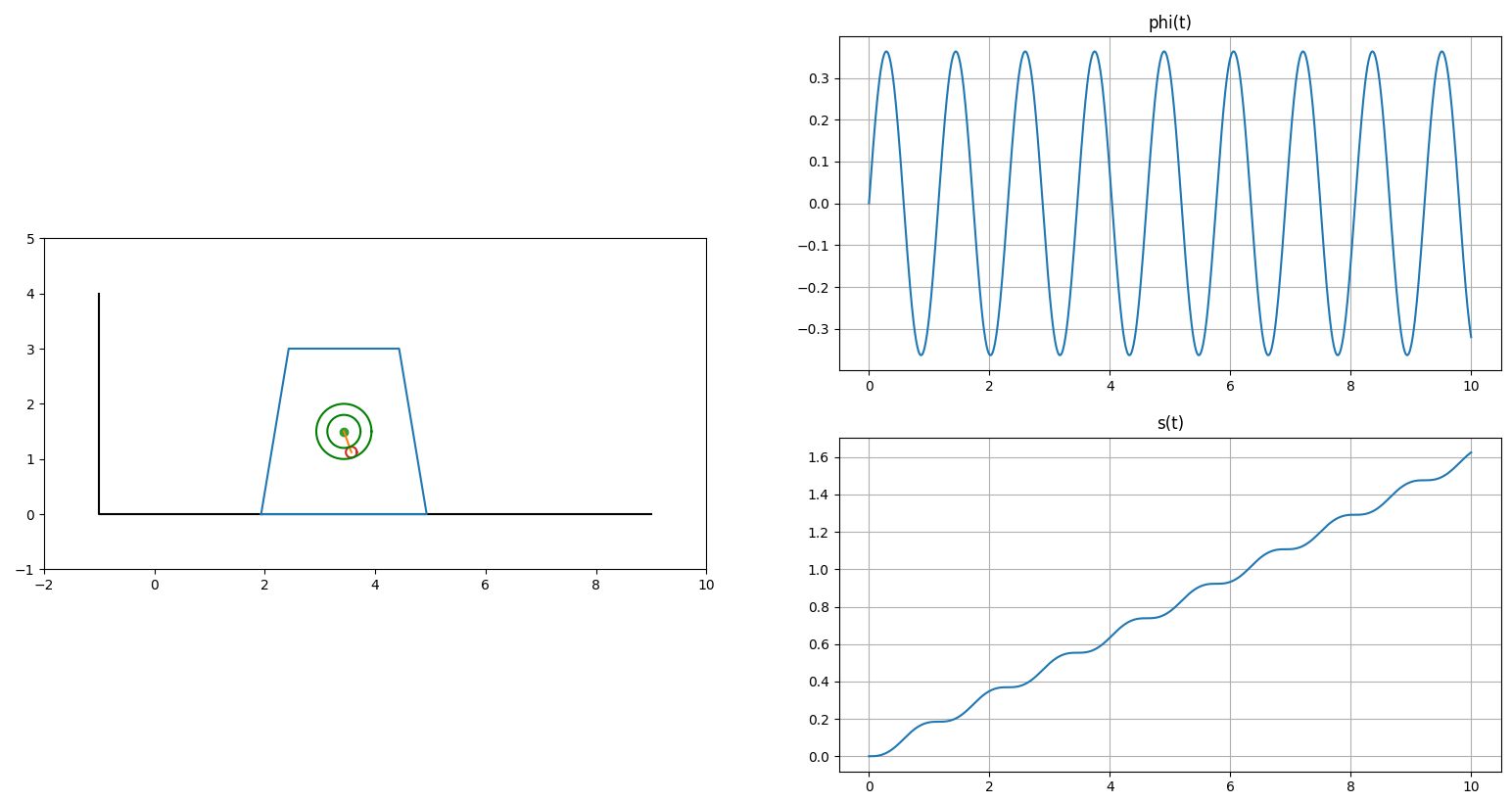
ax2.plot(t, ddphi)

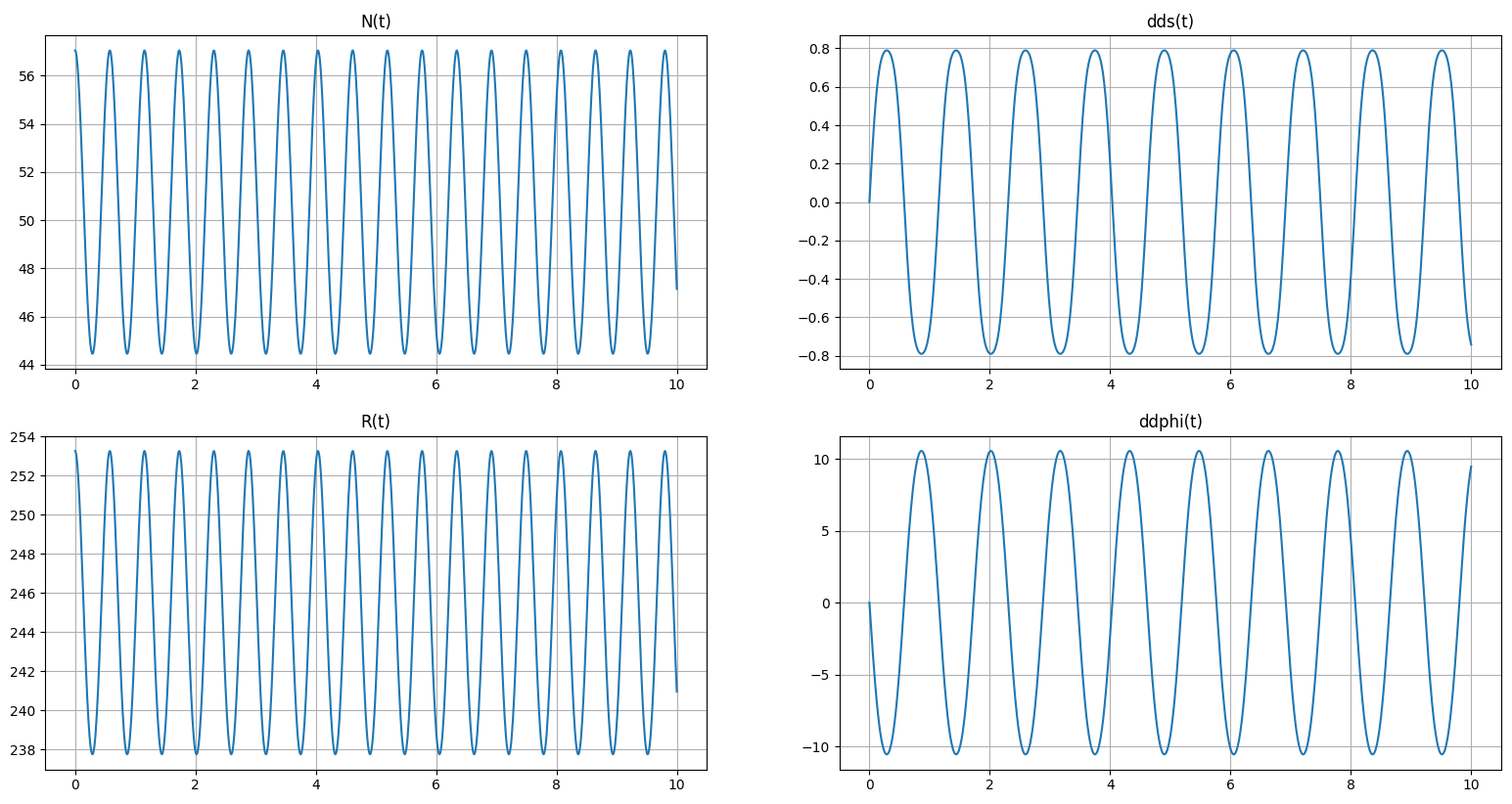
ax2.grid(True)

plt.show()

**Результат работы программы:**

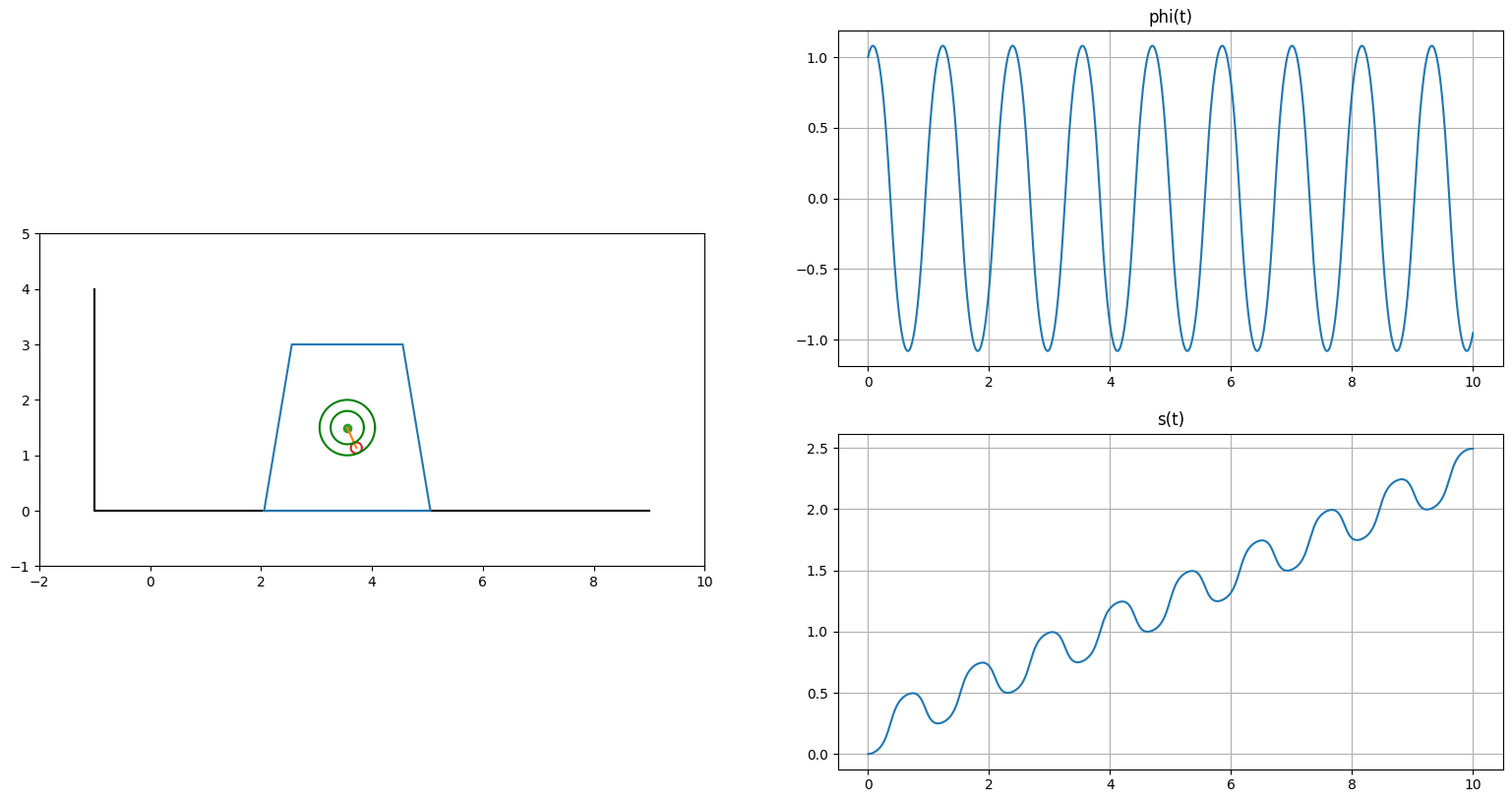
При , где — масса трапеции, — масса точки, — радиус вращения точки, — начальный угол отклонения.

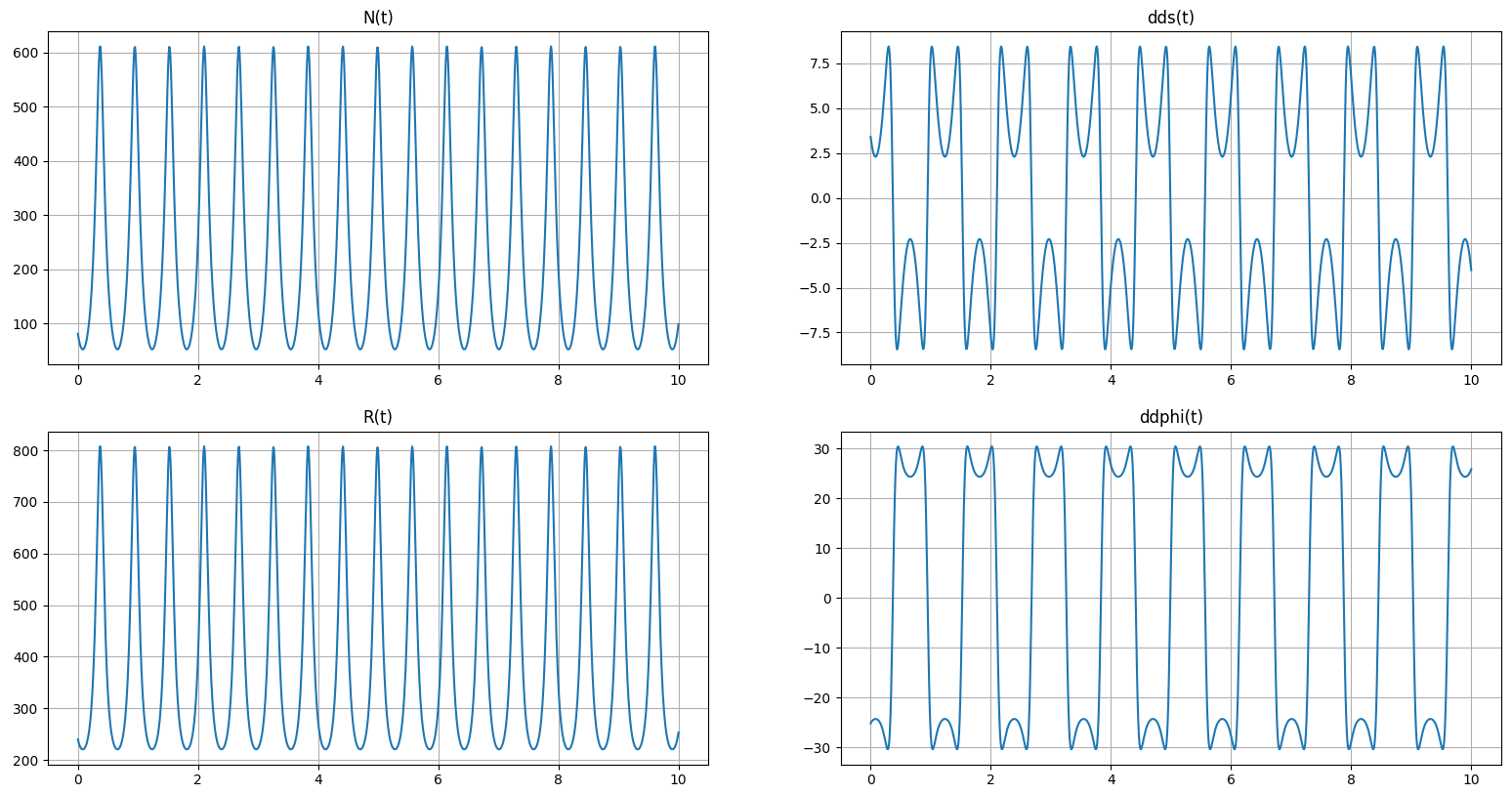




**Результат:** колебания слабо заметны, движение только вправо.

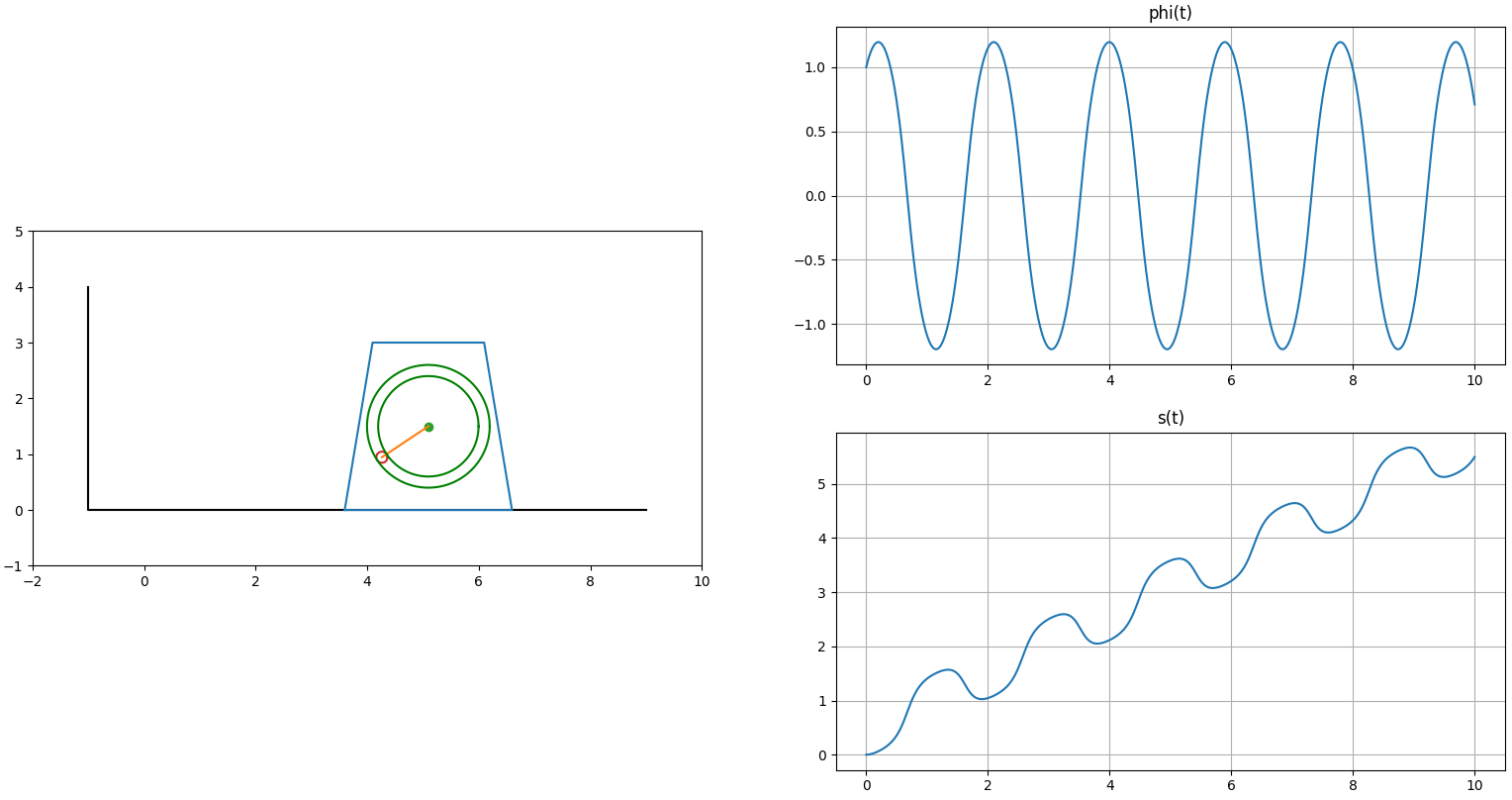
При

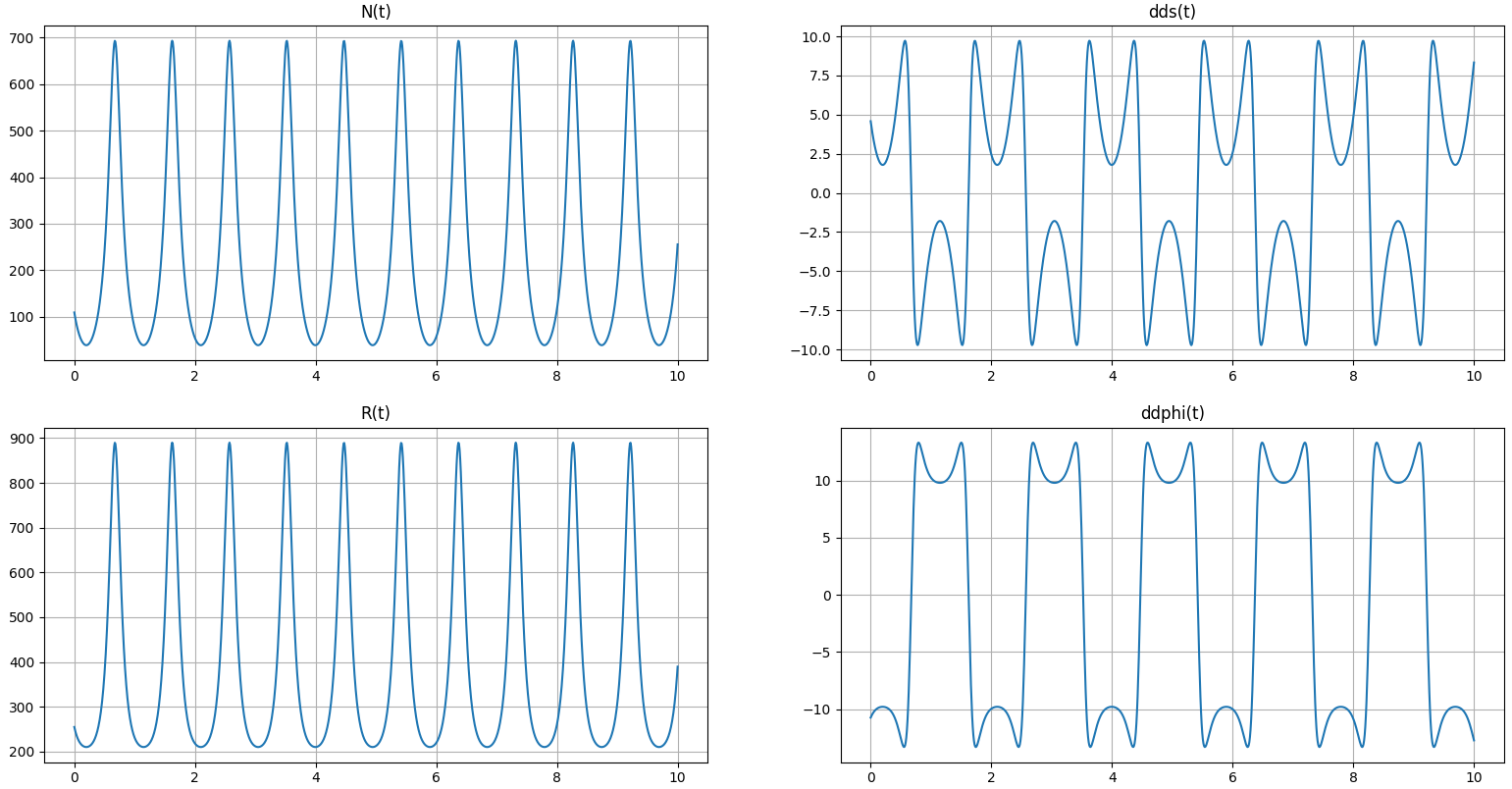




**Результат:** ярко выраженный колебательный характер, трапеция немного отодвигается влево, затем следует рывок вправо.

При





**Результат:** колебания усилились.

**Вывод:** проделав лабораторную работу, я построил анимацию движения системы, построил графики законов её движения.