ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«ДИНАМИКА МЕХАНИЧЕСКОЙ СИСТЕМЫ»**

**ПО ДИСЦИПЛИНЕ «ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ»**

**ВАРИАНТ ЗАДАНИЯ № 1**

Выполнил(а) студент группы М8О-203Б-23

Арусланов К.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

подпись, дата

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

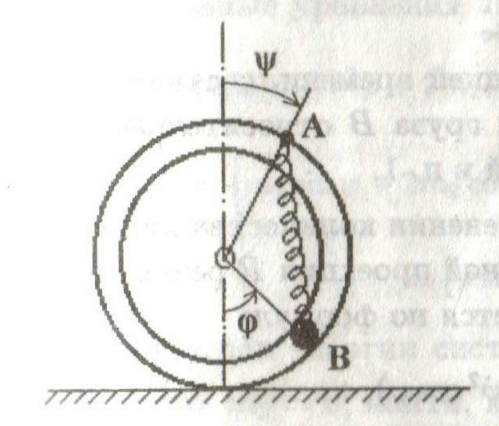
Москва, 2024

**Вариант № 1**

**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

# --- Параметры системы ---

R = 2.0    # Радиус колеса (м)

c = 40.0   # Жесткость пружины (Н/м)

m = 2.0    # Масса шарика (кг)

M = 1.0    # Масса колеса

g = 9.81   # Ускорение свободного падения (м/с^2)

v = 1.0    # Линейная скорость центра колеса (м/с)

phi0 = np.pi / 3   # Начальное отклонение угла фи (рад)

psi0 = 0.0         # Начальный угол пси (рад)

phi\_dot0 = 0.0      # Начальная угловая скорость фи (рад/с)

psi\_dot0 = 0.0      # Начальная угловая скорость пси (рад/с)

ball\_radius = 0.2   # Радиус шарика (м)

# Параметры времени

Steps = 1001

t\_fin = 20

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

dt = t[1]-t[0]

# Функция правых частей уравнений Лагранжа

def odesys(y, t, M, m, c, g, R):

    # y = [phi, psi, phi\_dot, psi\_dot]

    phi = y[0]

    psi = y[1]

    phi\_dot = y[2]

    psi\_dot = y[3]

    alpha = (phi + psi)/2.0

    # Матрица системы:

    # A \* [ddphi; ddpsi] = B

    A11 = 1.0

    A12 = np.cos(phi)

    A21 = np.cos(phi)

    A22 = 1.0 + 2.0\*(M/m)

    det = A11\*A22 - A12\*A21

    # Правая часть

    # B1 и B2 из уравнений Лагранжа:

    # phi'' + psi'' cos phi = -[2c/m(1−cos alpha) sin alpha + (g/R) sin phi]

    # [1+2(M/m)] psi'' + phi'' cos phi - phi\_dot² sin phi = -2c/m(1−cos alpha) sin alpha

    B1 = - (2\*c/m)\*(1 - np.cos(alpha))\*np.sin(alpha) - (g/R)\*np.sin(phi)

    B2 = - (2\*c/m)\*(1 - np.cos(alpha))\*np.sin(alpha) + (phi\_dot\*\*2)\*np.sin(phi)

    ddphi = (B1\*A22 - B2\*A12) / det

    ddpsi = (A11\*B2 - A21\*B1) / det

    return np.array([phi\_dot, psi\_dot, ddphi, ddpsi])

# Начальные условия

y0 = [phi0, psi0, phi\_dot0, psi\_dot0]

# Численное интегрирование

Y = odeint(odesys, y0, t, args=(M, m, c, g, R))

phi = Y[:, 0]

psi = Y[:, 1]

phi\_dot = Y[:, 2]

psi\_dot = Y[:, 3]

# Чтобы найти ddphi и ddpsi на каждом шаге, снова вызовем odesys

ddphi = np.zeros\_like(phi)

ddpsi = np.zeros\_like(phi)

for i in range(len(t)):

    dydt = odesys(Y[i,:], t[i], M, m, c, g, R)

    ddphi[i] = dydt[2]

    ddpsi[i] = dydt[3]

# Теперь вычисляем силу N

# N = m[gcos phi + R( phi'^2- psi''sin phi )] + 2 R c (1−cos alpha) cos alpha

alpha = (phi + psi)/2.0

N = (m\*(g\*np.cos(phi) + R\*(phi\_dot\*\*2 - ddpsi\*np.sin(phi))) +

     2\*R\*c\*(1 - np.cos(alpha))\*np.cos(alpha))

# Координаты центра колеса

X\_O = v \* t

Y\_O = R

# Координаты точки A

X\_A = X\_O + R \* np.sin(psi)

Y\_A = Y\_O - R \* np.cos(psi)

# Положение шарика

center\_radius = R - ball\_radius

X\_B = X\_O + center\_radius \* np.sin(phi)

Y\_B = Y\_O - center\_radius \* np.cos(phi)

# Параметры пружины

spring\_segments = 20

# Координаты внутренней трубки

inner\_tube\_radius = R - ball\_radius\*2

X\_Tube = inner\_tube\_radius \* np.cos(np.linspace(0, 2\*np.pi, 100))

Y\_Tube = inner\_tube\_radius \* np.sin(np.linspace(0, 2\*np.pi, 100))

def create\_spring\_segments(x\_start, y\_start, x\_end, y\_end, segments):

    X\_spring = np.zeros(segments)

    Y\_spring = np.zeros(segments)

    for i in range(segments):

        fraction = i/(segments - 1)

        X\_spring[i] = x\_start + fraction\*(x\_end - x\_start)

        Y\_spring[i] = y\_start + fraction\*(y\_end - y\_start)

        if 0 < i < segments - 1:

            if i % 2 == 0:

                X\_spring[i] += 0.1\*(y\_end - y\_start)

            else:

                X\_spring[i] -= 0.1\*(y\_end - y\_start)

    return X\_spring, Y\_spring

# --- Анимация ---

fig, ax = plt.subplots(figsize=(10, 6))

ax.set\_xlim(X\_O.min() - R - 1, X\_O.max() + R + 1)

ax.set\_ylim(-R - 1, Y\_O + R + 1)

ax.set\_aspect('equal')

ax.set\_xlabel('Горизонтальная позиция (м)')

ax.set\_ylabel('Вертикальная позиция (м)')

ax.set\_title('Анимация катящегося колеса с шариком и пружиной')

# Горизонтальная направляющая

ax.plot([X\_O.min() - R - 1, X\_O.max() + R + 1], [0, 0], 'k-', linewidth=2)

wheel\_outline = 100

psi\_circle = np.linspace(0, 2\*np.pi, wheel\_outline)

X\_Wheel = R\*np.cos(psi\_circle)

Y\_Wheel = R\*np.sin(psi\_circle)

wheel, = ax.plot([], [], 'b-', linewidth=2)

tube, = ax.plot([], [], 'gray', linestyle='--', linewidth=1)

spring, = ax.plot([], [], 'r-', linewidth=2)

ball = plt.Circle((X\_B[0], Y\_B[0]), ball\_radius, color='g')

ax.add\_patch(ball)

radius\_vector\_A, = ax.plot([], [], 'g--', linewidth=1)

radius\_vector\_B, = ax.plot([], [], 'b--', linewidth=1)

def init():

    wheel.set\_data([], [])

    tube.set\_data([], [])

    spring.set\_data([], [])

    ball.center = (X\_B[0], Y\_B[0])

    radius\_vector\_A.set\_data([], [])

    radius\_vector\_B.set\_data([], [])

    return wheel, tube, spring, ball, radius\_vector\_A, radius\_vector\_B

def anima(i):

    current\_X\_Wheel = X\_Wheel + X\_O[i]

    current\_Y\_Wheel = Y\_Wheel + Y\_O

    wheel.set\_data(current\_X\_Wheel, current\_Y\_Wheel)

    current\_X\_Tube = X\_Tube + X\_O[i]

    current\_Y\_Tube = Y\_Tube + Y\_O

    tube.set\_data(current\_X\_Tube, current\_Y\_Tube)

    X\_spring\_, Y\_spring\_ = create\_spring\_segments(X\_A[i], Y\_A[i], X\_B[i], Y\_B[i], spring\_segments)

    spring.set\_data(X\_spring\_, Y\_spring\_)

    ball.center = (X\_B[i], Y\_B[i])

    radius\_vector\_A.set\_data([X\_O[i], X\_A[i]], [Y\_O, Y\_A[i]])

    radius\_vector\_B.set\_data([X\_O[i], X\_B[i]], [Y\_O, Y\_B[i]])

    return wheel, tube, spring, ball, radius\_vector\_A, radius\_vector\_B

anim = FuncAnimation(fig, anima, init\_func=init,

                     frames=Steps, interval=40, blit=True, repeat=False)

plt.show()

# Построение трех графиков: x(t), phi(t), N(t)

fig2, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(8, 10))

# x(t) = X\_O(t)

ax1.plot(t, X\_O, label='x(t)')

ax1.set\_ylabel('x (м)')

ax1.set\_title('x(t)')

ax1.grid(True)

ax1.legend()

# phi(t)

ax2.plot(t, phi, label='phi(t)', color='r')

ax2.set\_ylabel('phi (рад)')

ax2.set\_title('phi(t)')

ax2.grid(True)

ax2.legend()

# N(t)

ax3.plot(t, N, label='N(t)', color='g')

ax3.set\_ylabel('N (H)')

ax3.set\_title('N(t)')

ax3.grid(True)

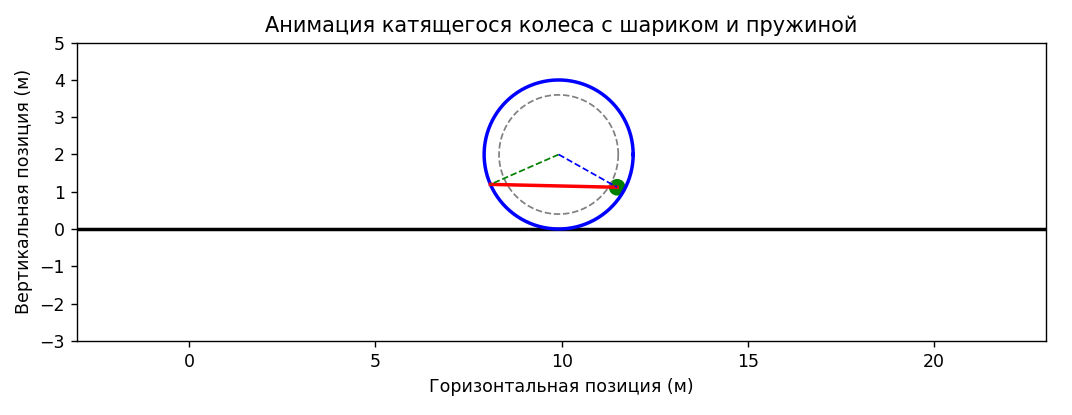
ax3.legend()

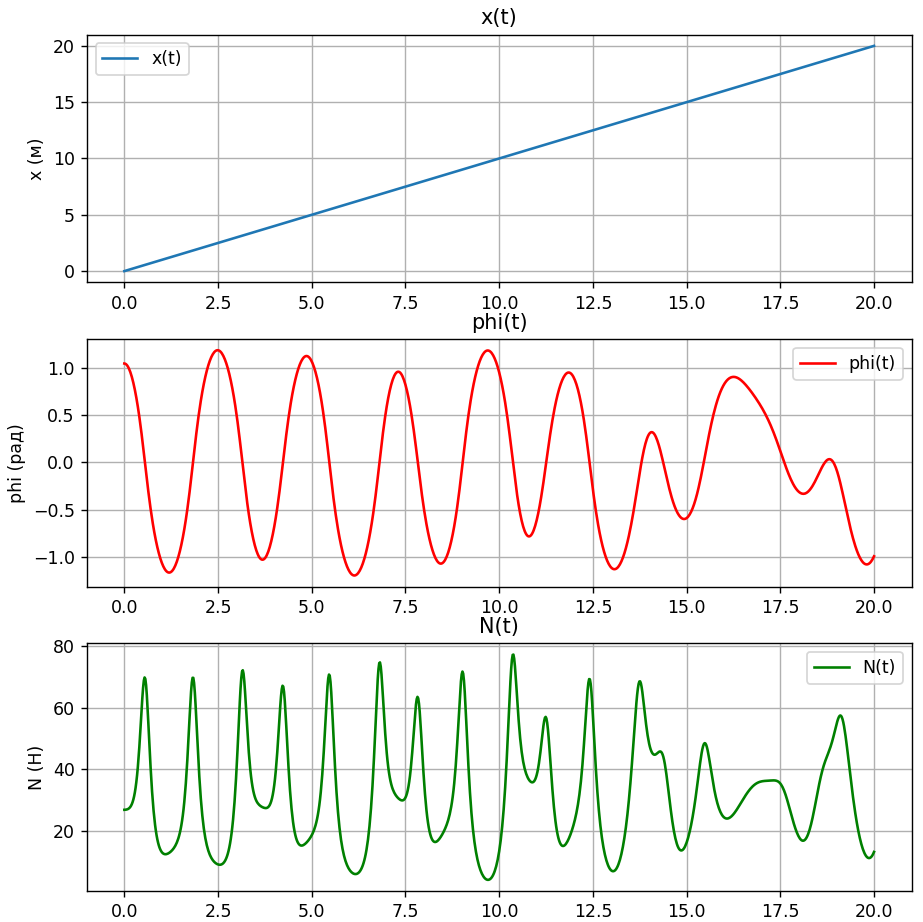
plt.tight\_layout()

plt.show()

**Результат работы программы:**

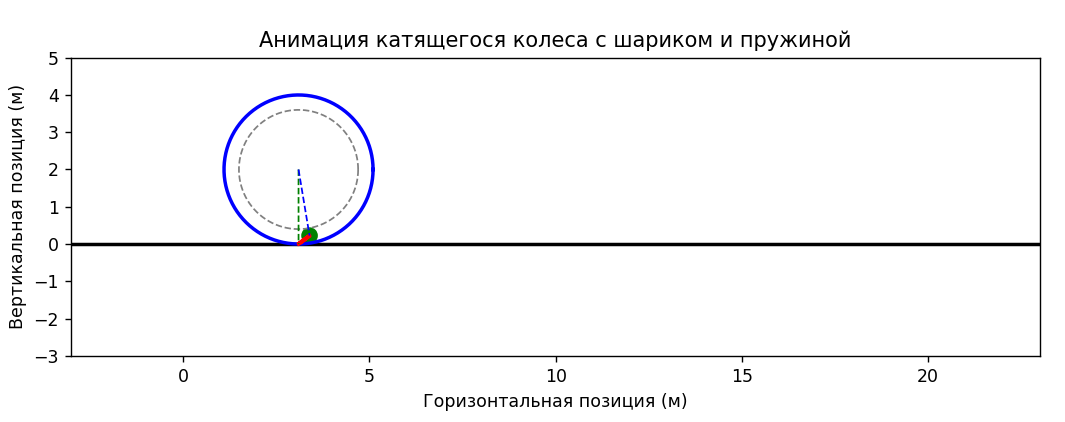
1. m = 2; M = 1; R = 2; c = 40; g = 9.81; ; – масса шарика больше массы колеса, довольно жесткая пружина, умеренное начальное отклонение:

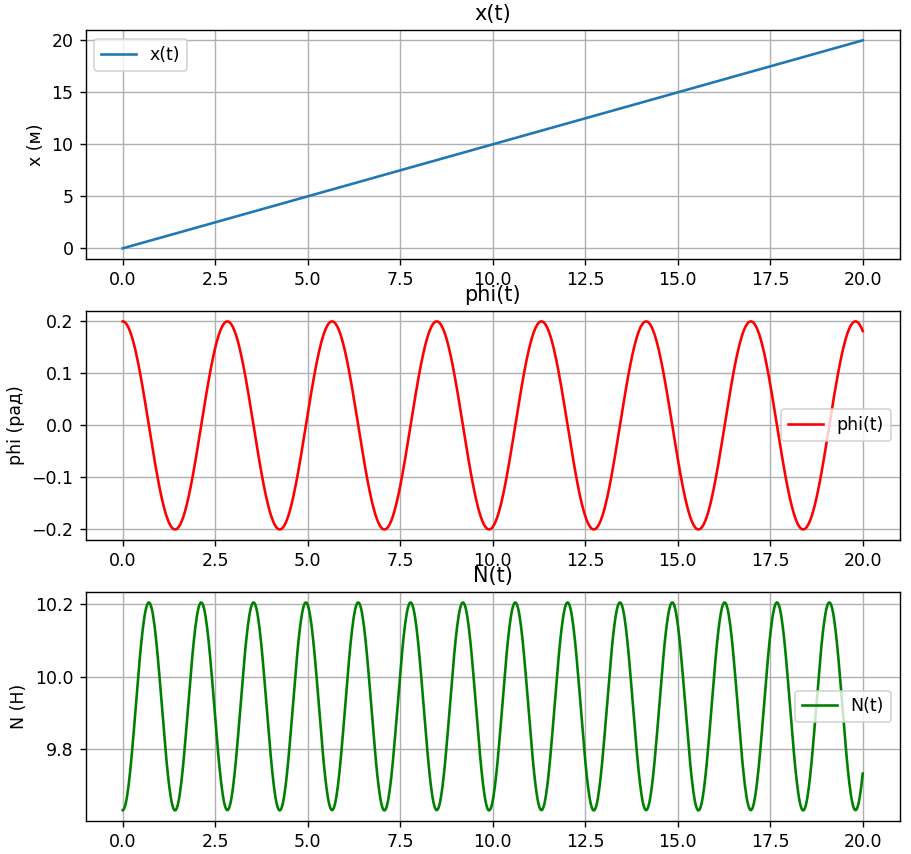
****

****

Результат: шарик с пружиной образуют подобие малораскачивающихся качелей, колебания происходят без значительных оборотов или хаотичных движений.

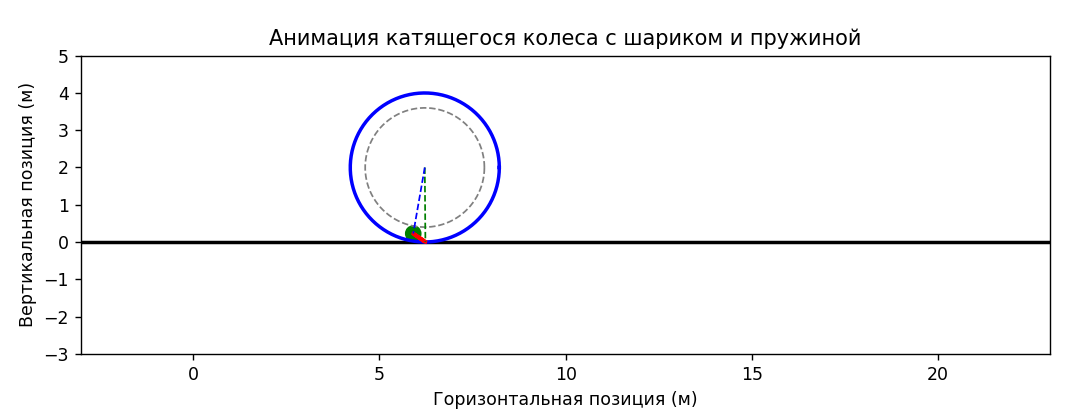
1. m = 1; M = 50; R = 2; c = 1; g = 9.81; ; – Слабая пружина, большая масса колеса и небольшое начальное отклонение:

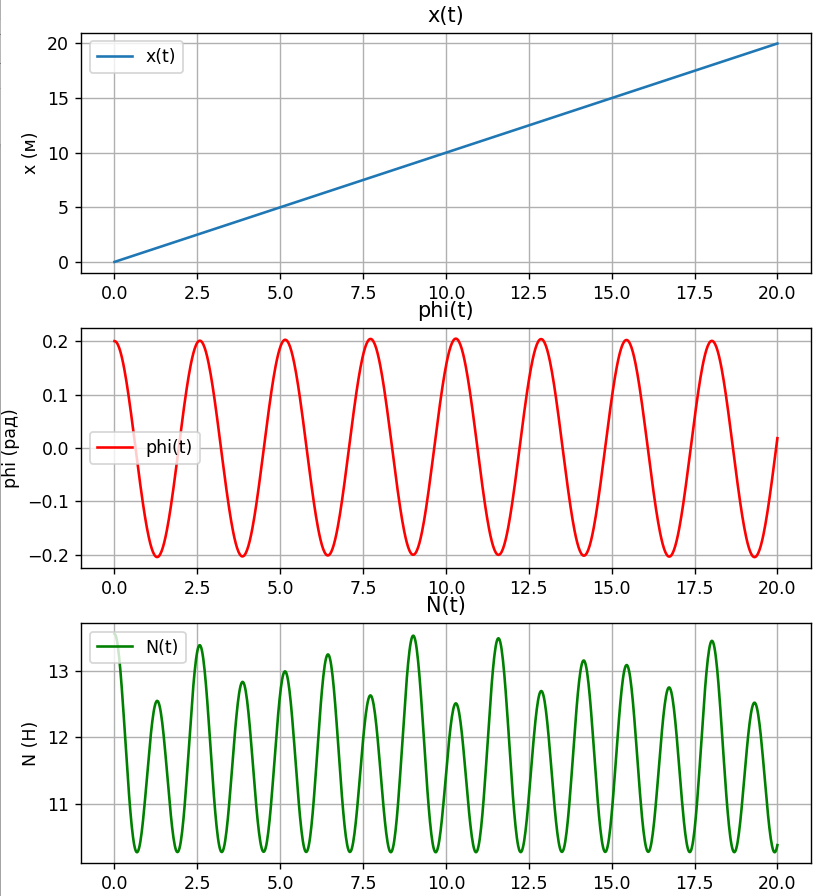
****

****

Результат: шарик совершает почти незаметные колебания около равновесия внутри колеса.

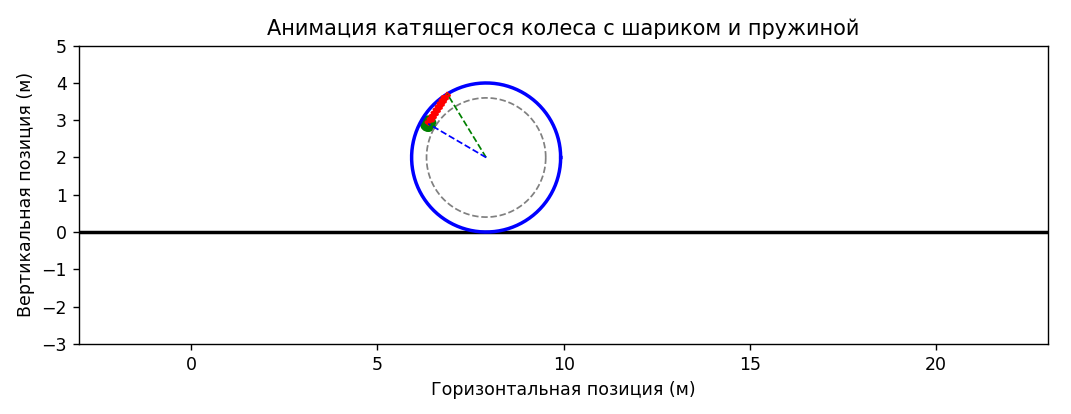
1. m = 1; M = 5; R = 2; c = 200; g = 9.81; ; – сильная пружина, небольшая масса колеса и небольшое начальное отклонение:

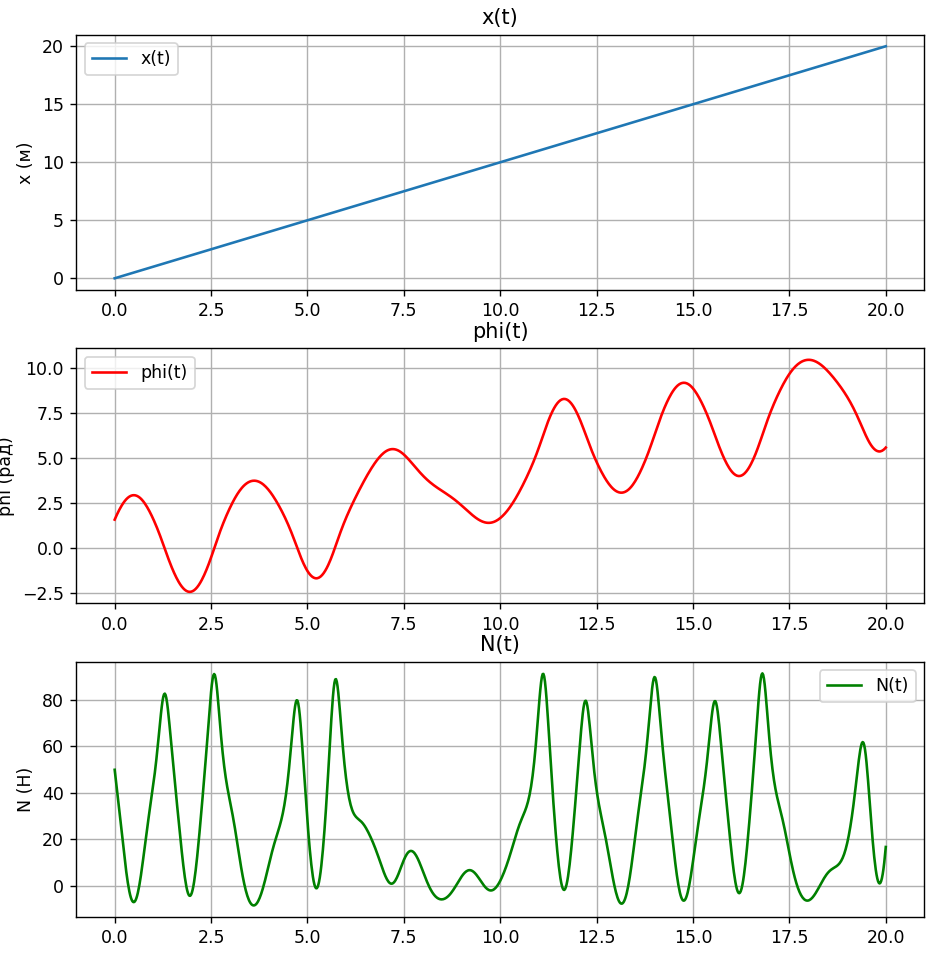
****

****

Результат: шарик выполняет малые колебания вокруг стабильного положения.

1. m = 1; M = 5; R = 2; c = 5; g = 9.81; ; – слабая пружина, небольшая масса колеса и сильное начальное возмущение:

****

****

Результат: шарик может облететь вокруг трубки, так как упругая сила не достаточно сильна, чтобы вернуть его в окрестность равновесия.1

**Вывод:**

Были добавлены уравнения движения, описанные через уравнения Лагранжа второго рода, что позволило численно моделировать поведение системы. Проведённые эксперименты с разными начальными условиями и параметрами показали разнообразие поведения системы. Мои навыки программирования на Python улучшились.