Московский авиационный институт

(национальный исследовательский университет)

Институт № 8 «Информационные технологии и прикладная математика»

**Лабораторная работа №3**

**по курсу «Теоретическая механика»**

**Динамика системы**

Выполнил студент группы М8О-203Б-20

Сорокин Никита Эдуардович

Преподаватель: Беличенко Михаил Валериевич

Холостова Ольга Владимировна

Оценка:

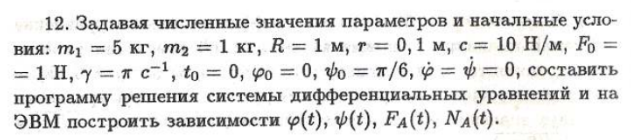
Дата: 10.05.2022

Москва, 2022

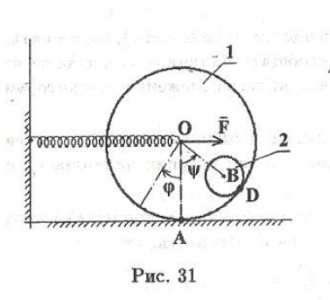
**Вариант №«31»**

**Задание:**

Численно решить дифференциальные уравнения движения механической системы c помощью Python, сделать задание №12 курсовой и построить анимацию движения системы.



**Механическая система:**

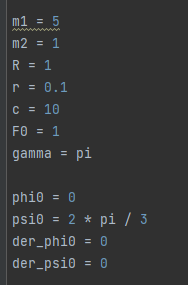
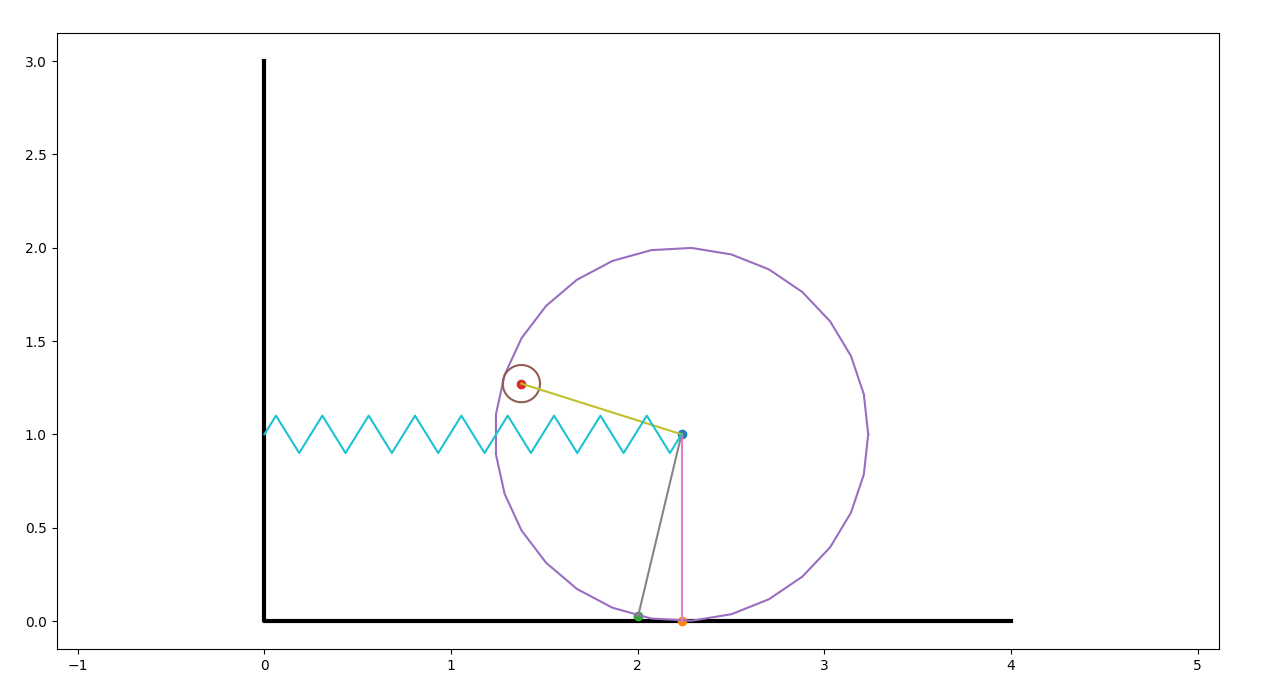
****

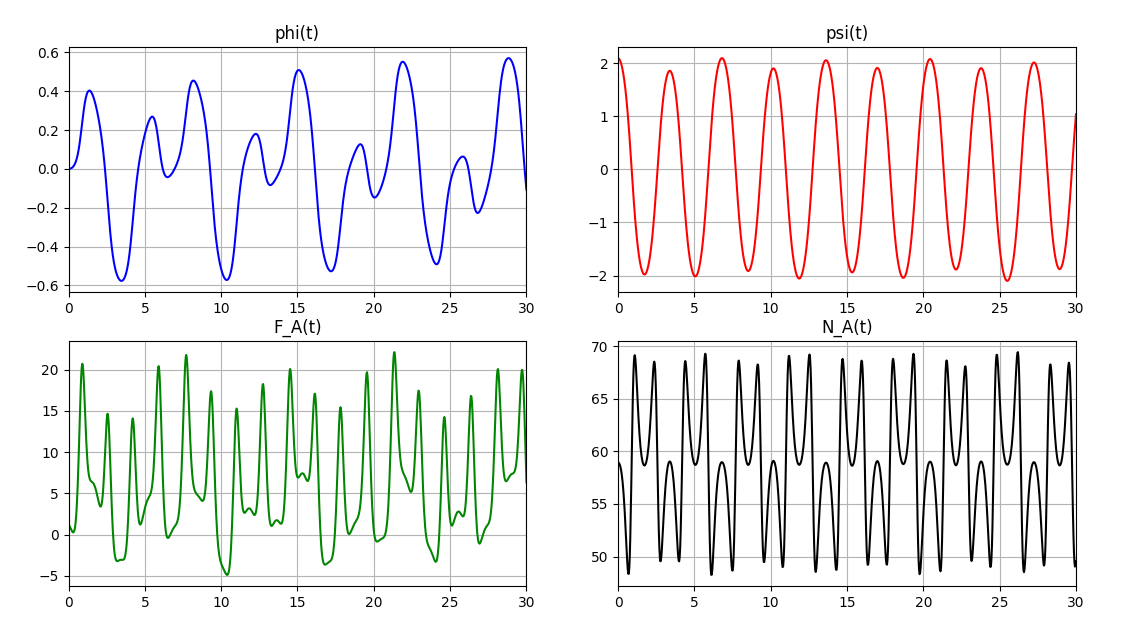
**Текст программы**

import numpy as npimport matplotlib.pyplot as pltfrom matplotlib.animation import FuncAnimationimport sympy as spfrom scipy.integrate import odeintpi = 4 \* np.arctan(1)g = 9.81full\_output = 1# y = (phi, psi, phi', psi')def odesys(y, t, m1, m2, R, r, c, F0, gamma, g): der\_y = np.zeros(4) der\_y[0] = y[2] # phi' = y[2] der\_y[1] = y[3] # psi' = y[3] a11 = 2 \* (m1 + m2) \* R a12 = m2 \* (R - r) \* (1 + np.cos(y[1])) a21 = R \* (1 + np.cos(y[1])) a22 = 2 \* (R - r) \* r b1 = (F0 \* np.sin(gamma \* t)) - (c \* R \* y[0]) + (m2 \* (R - r) \* y[3] \* y[3] \* np.sin(y[1])) b2 = - g \* np.sin(y[1]) der\_y[2] = (b1\*a22 - b2\*a12)/(a11\*a22 - a12\*a21) # phi'' находится по методу Крамера der\_y[3] = (b2\*a11 - b1\*a21)/(a11\*a22 - a12\*a21) # psi'' находится по методу Крамера return der\_y# Заданные числовые значения и начальные условия####################m1 = 5m2 = 0.5R = 1r = 0.1c = 10F0 = 1gamma = piphi0 = 0psi0 = pi / 6der\_phi0 = 0der\_psi0 = 0####################Steps = 1000t\_fin = 20t = np.linspace(0, t\_fin, Steps)y0 = [phi0, psi0, der\_phi0, der\_psi0]Y = odeint(odesys, y0, t, (m1, m2, R, r, c, F0, gamma, g))print(Y)# der - derivativephi = Y[:, 0]psi = -pi/2 + Y[:, 1]der\_phi = Y[:, 2]der\_psi = Y[:, 3]der2\_phi = [odesys(y, t, m1, m2, R, r, c, F0, gamma, g)[2] for y,t in zip(Y,t)]der2\_psi = [odesys(y, t, m1, m2, R, r, c, F0, gamma, g)[2] for y,t in zip(Y,t)]print(der2\_phi[1])F\_A = (m1 + m2)\*R\*der2\_phi[1] + m2\*(R - r)\*(der2\_psi\*np.cos(psi) - der\_psi\*der\_psi\*np.sin(psi)) + c\*R\*phi - F0\*np.sin(gamma\*t)N\_A = (m1 + m2)\*g + m2\*(R - r)\*(der2\_psi\*np.sin(psi) - der\_psi\*der\_psi\*np.cos(psi))fig\_for\_graphs = plt.figure(figsize=[13,7])ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 1)ax\_for\_graphs.plot(t, phi, color='blue')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(2,2,2)ax\_for\_graphs.plot(t,psi + pi/2,color='red')ax\_for\_graphs.set\_title('psi(t)')ax\_for\_graphs.set(xlim=[0,t\_fin])ax\_for\_graphs.grid(True)ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2,2,3)ax\_for\_graphs.plot(t,F\_A,color='green')ax\_for\_graphs.set\_title("F\_A(t)")ax\_for\_graphs.set(xlim=[0,t\_fin])ax\_for\_graphs.grid(True)ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2,2,4)ax\_for\_graphs.plot(t,N\_A,color='black')ax\_for\_graphs.set\_title('N\_A(t)')ax\_for\_graphs.set(xlim=[0,t\_fin])ax\_for\_graphs.grid(True)# Отсюда начинается лр2R1\_array = np.full(Steps, R, dtype=int)r\_0 = 2Point\_O = R \* phi + r\_0theta = np.linspace(0, 2 \* pi, 30)Circle1\_X = R \* np.cos(theta)Circle1\_Y = R \* np.sin(theta)Ground\_X = [0, 0, 4]Ground\_Y = [3, 0, 0]Line\_OH\_X = [0, 0]Line\_OH\_Y = [0, R]Point\_A\_X = Point\_O + R \* np.cos(- phi - pi / 2)Point\_A\_Y = R1\_array + R \* np.sin(- phi - pi / 2)Point\_B\_X = Point\_O + (R - r) \* np.cos(psi)Point\_B\_Y = R1\_array + (R - r) \* np.sin(psi)Circle2\_X = r \* np.cos(theta)Circle2\_Y = r \* np.sin(theta)AnglesCount = 20MaxWidth = 0.1Spring\_X = np.zeros(AnglesCount)Spring\_Y = np.zeros(AnglesCount)Spring\_X[AnglesCount - 1] = 1k = AnglesCount - 2for i in range(AnglesCount - 2): Spring\_X[i + 1] = (i + 1) / k - 1 / (2 \* k) Spring\_Y[i + 1] = ((-1) \*\* i) \* MaxWidthSpring\_Length = Point\_OFigure = plt.figure(figsize=[15,8])ax = Figure.add\_subplot(1, 1, 1)ax.axis("equal")Drawed\_Ground = ax.plot(Ground\_X, Ground\_Y, color="black", linewidth=3)Drawed\_Point\_O = ax.plot(Point\_O[0], R, marker="o")[0]Drawed\_Point\_H = ax.plot(Point\_O[0], 0, marker="o")[0]Drawed\_Point\_A = ax.plot(Point\_A\_X[0], Point\_A\_Y[0], marker="o")[0]Drawed\_Point\_B = ax.plot(Point\_B\_X[0], Point\_B\_Y[0], marker="o")[0]Drawed\_Circle1 = ax.plot(Point\_O[0] + Circle1\_X, R + Circle1\_Y)[0]Drawed\_Circle2 = ax.plot(Point\_B\_X[0] + Circle2\_X, Point\_B\_Y[0] + Circle2\_Y)[0]Drawed\_Line\_OH = ax.plot(Line\_OH\_X, Line\_OH\_Y)[0]Drawed\_Line\_OA = ax.plot([Point\_O[0], Point\_A\_X[0]], [R1\_array[0], Point\_A\_Y[0]])[0]Drawed\_Line\_OB = ax.plot([Point\_O[0], Point\_B\_X[0]], [R1\_array[0], Point\_B\_Y[0]])[0]Drawed\_Spring = ax.plot(Spring\_X \* Spring\_Length[0], Spring\_Y + R)[0]def Movement(i) : Drawed\_Point\_O.set\_data(Point\_O[i], R) Drawed\_Point\_H.set\_data(Point\_O[i], 0) Drawed\_Point\_A.set\_data(Point\_A\_X[i], Point\_A\_Y[i]) Drawed\_Point\_B.set\_data(Point\_B\_X[i], Point\_B\_Y[i]) Drawed\_Circle1.set\_data(Point\_O[i] + Circle1\_X, R + Circle1\_Y) Drawed\_Circle2.set\_data(Point\_B\_X[i] + Circle2\_X, Point\_B\_Y[i] + Circle2\_Y) Drawed\_Line\_OH.set\_data(Line\_OH\_X + Point\_O[i], Line\_OH\_Y) Drawed\_Line\_OA.set\_data([Point\_O[i], Point\_A\_X[i]], [R1\_array[i], Point\_A\_Y[i]]) Drawed\_Line\_OB.set\_data([Point\_O[i], Point\_B\_X[i]], [R1\_array[i], Point\_B\_Y[i]]) Drawed\_Spring.set\_data(Spring\_X \* Spring\_Length[i], Spring\_Y + R)Animation = FuncAnimation(Figure, Movement, frames=Steps, interval=10)plt.show()

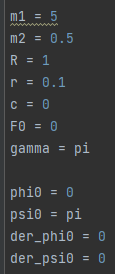
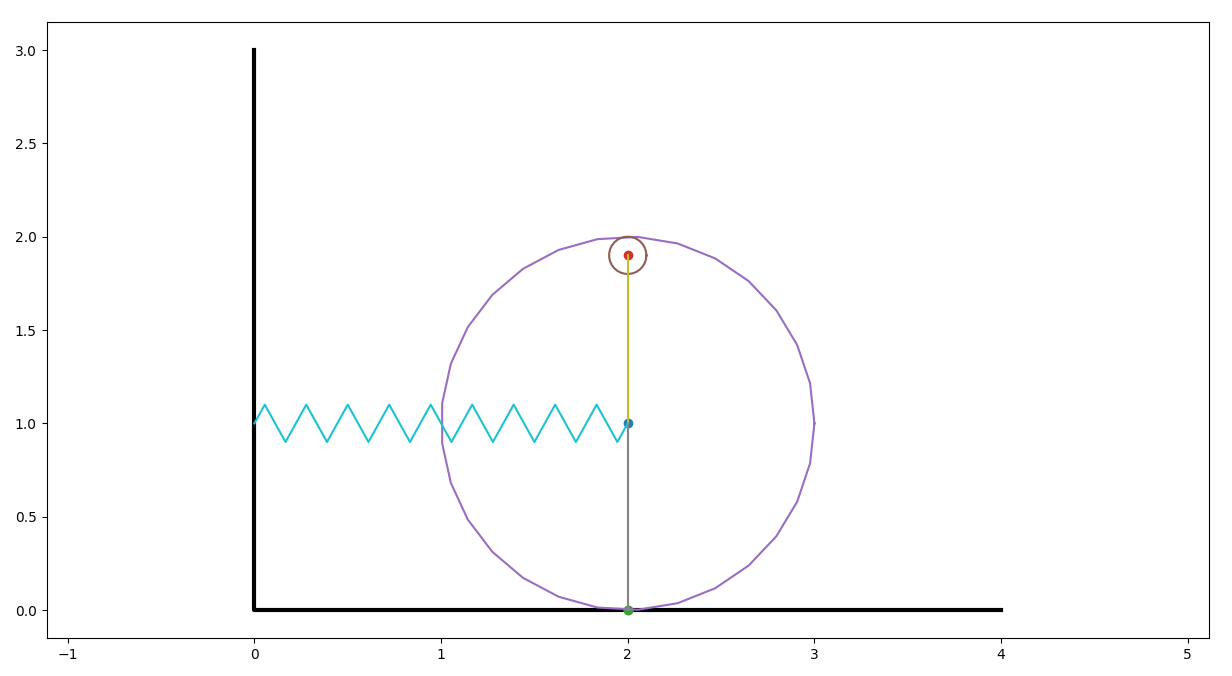
**Результат работы:**

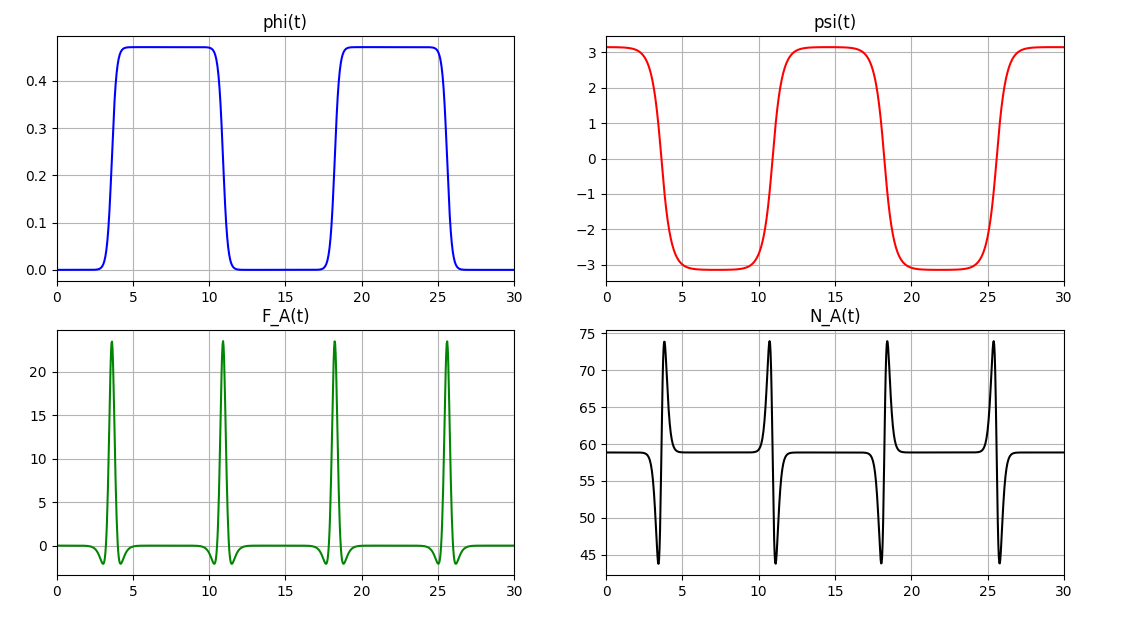
Тест 1. Набор данных похожий на набор из №12





Тест 2. Положение неустойчивого равновесия





Тест 3. Сильная пружина

