ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«№3»**

**ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА»**

**ВАРИАНТ ЗАДАНИЯ №22**

Выполнил студент группы М80-203Б-23

Салихов Р.Р.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

подпись, дата

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

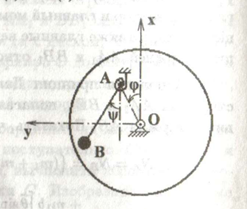
Москва, 2024

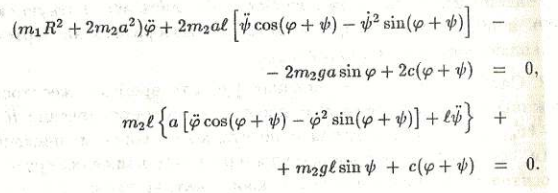
**Вариант №22**

**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:**



Диффуры двжиения: 

**Текст программы**

|  |
| --- |
| import numpy as np  import math  import matplotlib.pyplot as plt  from matplotlib.animation import FuncAnimation  import sympy as sp  from scipy.integrate import odeint  def odesys(y, t, r, a, l, m1, m2, c, g): # Функция создания системы диффуров  dy = np.zeros(4) # инициализация массива для производных  dy[0] = y[2] # угловая скорость для фи  dy[1] = y[3] # угловая скорость для психи  a11 = m1\*r\*\*2 + 2\*m2\*a\*\*2 # вычисление первого элемента матрицы системы  a12 = 2\*m2\*a\*l\*np.cos(y[0] + y[1]) # вычисление второго элемента матрицы  a21 = m2\*l\*a\*np.cos(y[0] + y[1]) # вычисление третьего элемента матрицы  a22 = m2\*l\*\*2 # вычисление четвертого элемента матрицы  b1 = 2\*m2\*a\*l\*np.sin(y[0] + y[1])\*y[3]\*\*2 + 2\*m2\*g\*a\*np.sin(y[0]) - 2\*c\*(y[0] + y[1]) # правая часть для первой координаты  b2 = m2\*l\*a\*np.sin(y[0] + y[1])\*y[2]\*\*2 - m2\*g\*l\*np.sin(y[1]) - c\*(y[0] + y[1]) # правая часть для второй координаты  dy[2] = (b1\*a22 - b2\*a12) / (a11\*a22 - a12\*a21) # угловое ускорение для фи  dy[3] = (b2\*a11 - b1\*a21) / (a11\*a22 - a12\*a21) # угловое ускорение для психи  return dy  ### ИЗМЕНЯЕМЫЕ ПАРАМЕТРЫ СИСТЕМЫ  R = 1 # радиус диска  A = 0.5 # расстояние между шарниром и центром диска  L = 1 # длина стержня, на котором шарнирно прикреплён груз  M1 = 1 # масса диска  M2 = 1 # масса груза  C = 1 # жёсткость спиральной пружины  G = 9.81 # ускорение свободного падения  ### НАЧАЛЬНЫЕ ЗНАЧЕНИЯ  PHI0 = math.pi/6 # начальный угол фи  PSI0 = 0 # начальный угол психи  DPHI0 = 0 # начальная угловая скорость для фи  DPSI0 = 0 # начальная угловая скорость для психи  Y0 = [PHI0, PSI0, DPHI0, DPSI0] # начальные условия  ### СТАТИЧЕСКАЯ ЧАСТЬ  X\_C = R + A + L # координаты центра диска  Y\_C = R + A + L  RM = R / 20 # радиус маленького круга в центре диска  ang = np.linspace(0, 2\*math.pi, 80) # углы для отрисовки кругов  X\_Disk = X\_C + R \* np.cos(ang) # координаты диска  Y\_Disk = Y\_C + R \* np.sin(ang)  X\_Sm = X\_C + RM \* np.cos(ang) # координаты маленького круга в центре диска  Y\_Sm = Y\_C + RM \* np.sin(ang)  X\_Side\_1 = [X\_C + RM \* np.cos(math.pi \* 5 / 4), X\_C + RM \* 3 \* np.cos(math.pi \* 5 / 4)] # боковые линии (центр)  Y\_Side\_1 = [X\_C + RM \* np.sin(math.pi \* 5 / 4), Y\_C + RM \* 3 \* np.sin(math.pi \* 5 / 4)]  X\_Side\_2 = [X\_C + RM \* np.cos(math.pi / -4), X\_C + RM \* 3 \* np.cos(math.pi / -4)]  Y\_Side\_2 = [X\_C + RM \* np.sin(math.pi / -4), Y\_C + RM \* 3 \* np.sin(math.pi / -4)]  X\_Bottom = [X\_Side\_1[1] - R / 40, X\_Side\_2[1] + R / 40] # линия-закреп центра  Y\_Bottom = [Y\_Side\_1[1], Y\_Side\_2[1]]  X\_Lines\_1 = np.linspace(float(X\_Bottom[0]) + R / 50, float(X\_Bottom[1]) - R / 50, 5) # полоски на линии-закрепа центра диска  X\_Lines\_2 = X\_Lines\_1 + R / 20 \* np.cos(math.pi \* 9 / 8)  Y\_Lines\_1 = np.full(5, Y\_Bottom[0])  Y\_Lines\_2 = Y\_Lines\_1 + R / 20 \* np.sin(math.pi \* 9 / 8)  ### ДИНАМИЧЕСКАЯ ЧАСТЬ  Steps = 1000 # количество шагов  t\_fin = 20 # конечное время  t = np.linspace(0, t\_fin, Steps) # время  X\_Sh = np.zeros\_like(t) # координаты шарнира  Y\_Sh = np.zeros\_like(t)  X\_Gr = np.zeros\_like(t) # координаты груза  Y\_Gr = np.zeros\_like(t)  Sol = odeint(odesys, Y0, t, (R, A, L, M1, M2, C, G)) # решение диффура  phi = Sol[:, 0] # угол между вертикальной осью и радиус-вектором к шарниру  psi = Sol[:, 1] # угол между вертикальной осью и стержнем  dphi = Sol[:, 2] # угловые скорости  dpsi = Sol[:, 3]  ddphi = [odesys(y, t, R, A, L, M1, M2, C, G)[2] for y, t in zip(Sol, t)] # угловые ускорения для фи  ddpsi = [odesys(y, t, R, A, L, M1, M2, C, G)[3] for y, t in zip(Sol, t)] # угловые ускорения для психи  for i in np.arange(len(t)): # просчёт основных величин  X\_Sh[i] = X\_C + A \* np.cos(phi[i] + math.pi / 2) # вычисление координат шарнира  Y\_Sh[i] = Y\_C + A \* np.sin(phi[i] + math.pi / 2)  X\_Gr[i] = X\_Sh[i] + L \* np.cos(-psi[i] - math.pi / 2) # вычисление координат груза  Y\_Gr[i] = Y\_Sh[i] + L \* np.sin(-psi[i] - math.pi / 2)  ### ПЕРЕХОД К ОТРИСОВКЕ  fig = plt.figure() # задаём пространство для отрисовки  ax = fig.add\_subplot(1, 1, 1)  ax.axis('equal')  ax.set(xlim=[0, X\_C \* 2], ylim=[0, Y\_C \* 2])  ax.set(xlabel="x", ylabel="y")  ### СТАТИЧЕСКАЯ ОТРИСОВКА  ax.plot(X\_C, Y\_C, marker='o', markersize=1, color='blue') # отрисовка центра диска  ax.plot(X\_Disk, Y\_Disk, color='blue') # отрисовка диска  ax.plot(X\_Sm, Y\_Sm, color='blue') # отрисовка кружка вокруг центра диска  ax.plot(X\_Side\_1, Y\_Side\_1, color='blue') # отрисовка боковых линий от центра диска  ax.plot(X\_Side\_2, Y\_Side\_2, color='blue')  ax.plot(X\_Bottom, Y\_Bottom, color='blue') # отрисовка линии-закрепа центра диска  for i in np.arange(len(X\_Lines\_1)): # отрисовка штрихов на линии-закрепе центра диска  ax.plot([X\_Lines\_1[i], X\_Lines\_2[i]], [Y\_Lines\_1[i], Y\_Lines\_2[i]], color='darkblue')  ### ДИНАМИЧЕСКАЯ ОТРИСОВКА  LEN = R / 6 # длина линии-закрепа пружинки  WIDE = R / 8 # ширина линии-закрепа пружинки  X\_DSHT = R / 20 \* np.cos(math.pi / 4) # сдвиги штрихов по координатам  Y\_DSHT = R / 20 \* np.cos(math.pi / 4)  R1 = R / 8 # радиусы спиральной пружины  R2 = R / 64  thetta = np.linspace(0, 3 / 2 \* math.pi + psi[0], 100) # угол проворота спиральной пружины  X\_SpiralSpr = (R1 + thetta \* (R2 - R1) / thetta[-1]) \* np.cos(thetta) # координаты точек спиральной пружины  Y\_SpiralSpr = -(R1 + thetta \* (R2 - R1) / thetta[-1]) \* np.sin(thetta)  spr, = ax.plot(X\_SpiralSpr + X\_Sh[0], Y\_SpiralSpr + Y\_Sh[0], color='green') # отрисовка спиральной пружины  pl1, = ax.plot([X\_Sh[0] + R1 - WIDE / 2, X\_Sh[0] + R1 - WIDE / 2 + X\_DSHT], [Y\_Sh[0] + LEN, Y\_Sh[0] + LEN + Y\_DSHT], color='darkgreen') # штрихи на линии-закрепе спиральки  pl2, = ax.plot([X\_Sh[0] + R1, X\_Sh[0] + R1 + X\_DSHT], [Y\_Sh[0] + LEN, Y\_Sh[0] + LEN + Y\_DSHT], color='darkgreen')  pl3, = ax.plot([X\_Sh[0] + R1 + WIDE / 2, X\_Sh[0] + R1 + WIDE / 2 + X\_DSHT], [Y\_Sh[0] + LEN, Y\_Sh[0] + LEN + Y\_DSHT], color='darkgreen')  hl, = ax.plot([X\_Sh[0] + R1 - WIDE / 2 - R / 10, X\_Sh[0] + R1 + WIDE / 2 + R / 10], [Y\_Sh[0] + LEN, Y\_Sh[0] + LEN], color='green') # отрисовка вертикальной линии от спиральки  upl, = ax.plot([X\_Sh[0] + R1, X\_Sh[0] + R1], [Y\_Sh[0], Y\_Sh[0] + LEN], color='green') # отрисовка линии-закрепа спирали  sh, = ax.plot(X\_Sh[0], Y\_Sh[0], marker='o', markersize=5, color='orange') # отрисовка шарнира  st, = ax.plot([X\_Sh[0], X\_Gr[0]], [Y\_Sh[0], Y\_Gr[0]], color='orange') # отрисовка стержня  gr, = ax.plot(X\_Gr[0], Y\_Gr[0], marker='o', markersize=20, color='orange') # отрисовка грузика  def anima(i): # функция анимации  thetta = np.linspace(0, 3 / 2 \* math.pi + psi[i], 100)  X\_SpiralSpr = (R1 + thetta \* (R2 - R1) / thetta[-1]) \* np.cos(thetta)  Y\_SpiralSpr = -(R1 + thetta \* (R2 - R1) / thetta[-1]) \* np.sin(thetta)  spr.set\_data(X\_SpiralSpr + X\_Sh[i], Y\_SpiralSpr + Y\_Sh[i]) # обновление координат спиральной пружины  pl1.set\_data([X\_Sh[i] + R1 - WIDE / 2, X\_Sh[i] + R1 - WIDE / 2 + X\_DSHT], [Y\_Sh[i] + LEN, Y\_Sh[i] + LEN + Y\_DSHT])  pl2.set\_data([X\_Sh[i] + R1, X\_Sh[i] + R1 + X\_DSHT], [Y\_Sh[i] + LEN, Y\_Sh[i] + LEN + Y\_DSHT])  pl3.set\_data([X\_Sh[i] + R1 + WIDE / 2, X\_Sh[i] + R1 + WIDE / 2 + X\_DSHT], [Y\_Sh[i] + LEN, Y\_Sh[i] + LEN + Y\_DSHT])  hl.set\_data([X\_Sh[i] + R1 - WIDE / 2, X\_Sh[i] + R1 + WIDE / 2], [Y\_Sh[i] + LEN, Y\_Sh[i] + LEN])  upl.set\_data([X\_Sh[i] + R1, X\_Sh[i] + R1], [Y\_Sh[i], Y\_Sh[i] + LEN])  sh.set\_data(X\_Sh[i], Y\_Sh[i]) # обновление позиции шарнира  st.set\_data([X\_Sh[i], X\_Gr[i]], [Y\_Sh[i], Y\_Gr[i]]) # обновление стержня  gr.set\_data(X\_Gr[i], Y\_Gr[i]) # обновление позиции груза  return spr, pl1, pl2, pl3, hl, upl, sh, st, gr  anim = FuncAnimation(fig, anima, frames=Steps, interval=40, repeat=False) # создаём разовую анимацию  ### ГРАФИКИ ЗАВИСИМОСТЕЙ ВЕЛИЧИН ОТ ВРЕМЕНИ  Nox = (M1 + M2) \* G - M2 \* (A \* (ddphi \* np.sin(phi) + dphi\*\*2 \* np.cos(phi)) - L \* (ddpsi \* np.sin(psi) + dpsi\*\*2 \* np.cos(psi))) # проекция реакции оси диска  k1 = 1 # коэффициенты для нахождения силы сопротивления R  k2 = 10  pls = plt.figure()  p1 = pls.add\_subplot(3, 2, 1) # строим графики величин  p1.set(xlim=[0, t\_fin])  p1.plot(t, phi)  p1.grid()  plt.title('Phi(t)')  p2 = pls.add\_subplot(3, 2, 3)  p2.plot(t, psi)  p2.grid()  p2.set(xlim=[0, t\_fin])  plt.title('Psi(t)')  p3 = pls.add\_subplot(3, 2, 2)  p3.plot(t, Nox)  p3.grid()  p3.set(xlim=[0, t\_fin])  plt.title('Nox(t)')  p4 = pls.add\_subplot(3, 2, 4)  p4.plot(t, -k1 \* dpsi)  p4.grid()  p4.set(xlim=[0, t\_fin])  plt.title('R1(t)')  p5 = pls.add\_subplot(3, 2, 6)  p5.plot(t, -k2 \* dpsi)  p5.grid()  p5.set(xlim=[0, t\_fin])  plt.title('R2(t)')  plt.tight\_layout() # чтобы не накладывались названия  plt.show()plt.close() |

**Результат работы программы:**

|  |  |
| --- | --- |
|  |  |
|  |  |

**Вывод:** успешно выполнил лабораторную работу по теоретической механике. С помощью языка программирования Python и библиотек matplotlib, numpy и scipy я смоделировал динамическое поведение системы, состоящей из диска с закреплённым на нем стержнем и грузом. Для этого была решена система дифференциальных уравнений, описывающих движение этой механической системы. В программе учтены основные законы динамики и силы, действующие на систему, такие как сила тяжести и упругие силы пружины.

Визуализация движения осуществляется через анимацию, отображающую поведение системы во времени, а также графики зависимостей углов, угловых скоростей и сил. Модель позволяет наблюдать, как будет изменяться положение стержня и груза, а также как на систему воздействуют силы, такие как реакция оси диска и сопротивление пружины.

Программа учитывает реальный физический процесс (без трения), что позволяет получить наглядное представление о поведении системы в реальной жизни.