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-215Б-23

Кармишен Егор Сергеевич

Преподаватель: Волков Евгений Валерьевич

Оценка:

Дата:

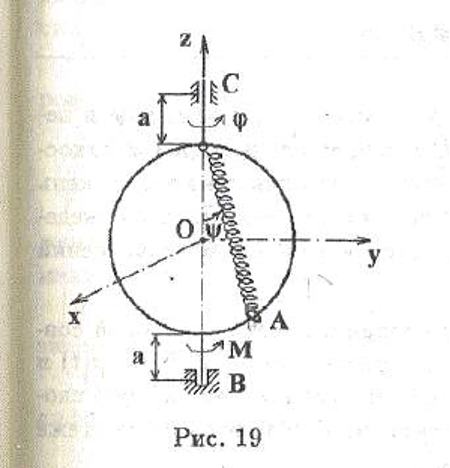
Москва, 2024

**Вариант №«Фантастическая система»**

**Задание:**

Численно решить дифференциальные уравнения движения механической системы в среде Octave (или Matlab), сделать задание №12 курсовой и построить анимацию движения системы.

**Механическая система:**

****

**Текст программы**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from scipy.integrate import odeint

import sympy as sp

import math

*# Функция для формирования системы дифференциальных уравнений*

*def* formY(*y*, *t*, *fV*, *fOm*):

y1, y2, y3, y4 = y

dydt = [y3, y4, fV(y1, y2, y3, y4), fOm(y1, y2, y3, y4)]

return dydt

*# Определение параметров*

alpha = math.pi / 6

M = 1

m = 0.1

R = 0.3

c = 20

l0 = 0.2

g = 9.81

*# Определение t как символа*

t = sp.Symbol('t')

*# Определение функций от 't':*

*# угол*

phi = sp.Function('phi')(t)

psi = sp.Function('psi')(t)

*# угловые скорости*

Vphi = sp.Function('Vphi')(t)

Vpsi = sp.Function('Vpsi')(t)

l = 2 \* R \* sp.cos(phi) *# длина пружины*

*# Построение уравнений Лагранжа*

*# 1 Определение кинетической энергии*

TT1 = M \* R\*\*2 \* Vphi\*\*2 / 4

V1 = 2 \* Vpsi \* R

V2 = Vphi \* R \* sp.sin(2 \* psi)

Vr2 = V1\*\*2 + V2\*\*2

TT2 = m \* Vr2 / 2

TT = TT1 + TT2

*# 2 Определение потенциальной энергии*

Pi1 = 2 \* R \* m \* g \* sp.sin(psi)\*\*2

Pi2 = (c \* (l - l0)\*\*2) / 2

Pi = Pi1 + Pi2

*# 3 Непотенциальная сила*

M = alpha \* phi\*\*2

*# Функция Лагранжа*

L = TT - Pi

*# Уравнения Лагранжа*

ur1 = sp.diff(sp.diff(L, Vphi), t) - sp.diff(L, phi) - M *# для обруча*

ur2 = sp.diff(sp.diff(L, Vpsi), t) - sp.diff(L, psi) *# для бусины*

*# Выделение вторых производных (dV/dt и dom/dt) с использованием метода Крамера*

a11 = ur1.coeff(sp.diff(Vphi, t), 1) *# Коэффициент при dVphi/dt в ur1*

a12 = ur1.coeff(sp.diff(Vpsi, t), 1) *# Коэффициент при dVpsi/dt в ur1*

a21 = ur2.coeff(sp.diff(Vphi, t), 1) *# Коэффициент при dVphi/dt в ur2*

a22 = ur2.coeff(sp.diff(Vpsi, t), 1) *# Коэффициент при dVpsi/dt в ur2*

*# Извлечение свободных членов*

b1 = -(ur1.coeff(sp.diff(Vphi, t), 0)).coeff(sp.diff(Vpsi, t), 0).subs([(sp.diff(phi, t), Vphi), (sp.diff(psi, t), Vpsi)])

b2 = -(ur2.coeff(sp.diff(Vphi, t), 0)).coeff(sp.diff(Vpsi, t), 0).subs([(sp.diff(phi, t), Vphi), (sp.diff(psi, t), Vpsi)])

detA = a11 \* a22 - a12 \* a21 *# определитель матрицы коэффициентов системы линейных уравнений.*

detA1 = b1 \* a22 - b2 \* a21 *# это определитель матрицы, полученной заменой первого столбца матрицы коэффициентов на столбец свободных членов.*

detA2 = a11 \* b2 - b1 \* a21 *# это определитель матрицы, полученной заменой второго столбца матрицы коэффициентов на столбец свободных членов.*

dVdt = detA1 / detA *# вторая производная phi*

domdt = detA2 / detA *# вторая производная psi*

countOfFrames = 2500

*# Построение системы дифференциальных уравнений*

T = np.linspace(0, 25, countOfFrames)

fVphi = sp.lambdify([phi, psi, Vphi, Vpsi], dVdt, "numpy") *# функция вычисления второй производной phi*

fVpsi = sp.lambdify([phi, psi, Vphi, Vpsi], domdt, "numpy") *# функция вычисления второй производной psi*

y0 = [0, np.pi / 6, -0.5, 0] *# начальные условия для системы дифференциальных уравнений.*

sol = odeint(formY, y0, T, *args*=(fVphi, fVpsi)) *# это массив, содержащий решения системы дифференциальных*

*# уравнений для каждого временного шага. Каждая строка массива sol соответствует одному временному шагу и содержит*

*# значения для углов phi и psi, а также их угловых скоростей Vphi и Vpsi.*

*# Извлечение решений*

phi = sol[:, 0] *# все значения из первого столбца*

psi = sol[:, 1] *# все значения из второго столбца*

Vphi = sol[:, 2] *# ...*

Vpsi = sol[:, 3] *# ...*

*# Функция для вычисления N1*

*def* compute\_N1(*phi*, *psi*, *Vphi*, *Vpsi*, *m*, *R*, *g*, *c*, *l0*):

return m \* R \* (4 \* Vpsi\*\*2 + Vphi\*\*2 \* np.sin(2 \* psi)\*\*2) + m \* g \* np.cos(2 \* psi) - c \* (2 \* R \* np.cos(psi) - l0) \* np.cos(psi)

*# Функция для вычисления N2*

*def* compute\_N2(*phi*, *psi*, *Vphi*, *Vpsi*, *m*, *R*, *g*, *c*, *l0*):

return m \* R \* (4 \* Vpsi\*\*2 + Vphi\*\*2 \* np.sin(2 \* psi)\*\*2) - m \* g \* np.cos(2 \* psi) + c \* (2 \* R \* np.cos(psi) - l0) \* np.cos(psi)

*# Вычисление N1 и N2 для каждого шага времени*

*# Проекции силы давления бусинки на обруч, лежащие соответственно в плоскости обруча и перпедикулярно ей*

N1 = np.array([compute\_N1(phi[i], psi[i], Vphi[i], Vpsi[i], m, R, g, c, l0) for i in range(len(T))])

N2 = np.array([compute\_N2(phi[i], psi[i], Vphi[i], Vpsi[i], m, R, g, c, l0) for i in range(len(T))])

*# Построение окна и графика с выравниванием осей*

fig = plt.figure(*figsize*=(17, 8))

ax1 = fig.add\_subplot(1, 2, 1)

ax1.axis('equal')

w = np.linspace(0, 2 \* math.pi, countOfFrames) *# массив углов, используемый для создания окружности.*

conline, = ax1.plot([np.sin(2 \* psi[0]) \* R \* np.cos(phi[0]), 0], [-np.cos(2 \* psi[0]) \* R, R], 'black') *# спираль*

P, = ax1.plot(np.sin(2 \* psi[0]) \* R \* np.cos(phi[0]), -np.cos(2 \* psi[0]) \* R, *marker*='o', *color*='black') *# точка, представляющая текущее положение на окружности.*

Circ, = ax1.plot(R \* np.cos(phi[0]) \* np.cos(w), R \* np.sin(w), 'black') *# окружность*

*# Дополнительные подграфики*

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

ax2.plot(T, Vphi)

ax2.set\_xlabel('T')

ax2.set\_ylabel('Vphi')

ax3 = fig.add\_subplot(4, 2, 4)

ax3.plot(T, Vpsi)

ax3.set\_xlabel('T')

ax3.set\_ylabel('Vpsi')

ax4 = fig.add\_subplot(4, 2, 6)

ax4.plot(T, N1)

ax4.set\_xlabel('T')

ax4.set\_ylabel('N1')

ax5 = fig.add\_subplot(4, 2, 8)

ax5.plot(T, N2)

ax5.set\_xlabel('T')

ax5.set\_ylabel('N2')

*def* anima(*i*): *# Функция для обновления данных анимации*

P.set\_data(np.sin(2 \* psi[i]) \* R \* np.cos(phi[i]), -np.cos(2 \* psi[i]) \* R)

conline.set\_data([np.sin(2 \* psi[i]) \* R \* np.cos(phi[i]), 0], [-np.cos(2 \* psi[i]) \* R, R])

Circ.set\_data(R \* np.cos(phi[i]) \* np.cos(w), R \* np.sin(w))

return Circ, P, conline

anim = FuncAnimation(fig, anima, *frames*=countOfFrames, *interval*=10) *# Создаем анимацию*

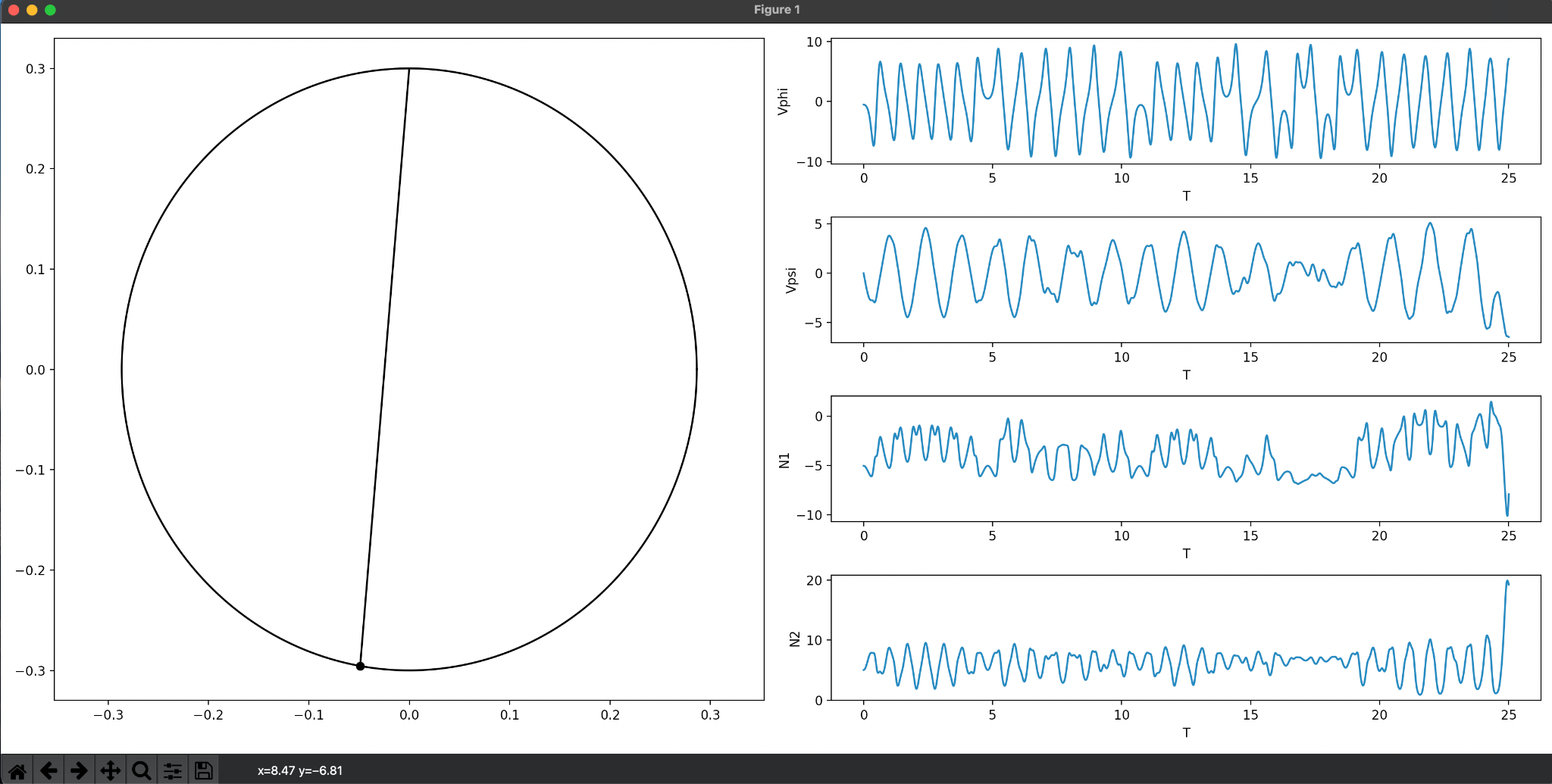
plt.tight\_layout() *# настройка параметров макета фигуры, чтобы избежать перекрытия элементов*

plt.show() *# Отображаем графики*

**Результат работы:**

1. m = 0,1 кг; M = 1 кг; R = 0,3 м; c = 20 Н/м; ; g = 9.81; phi = 0, psi = pi/6

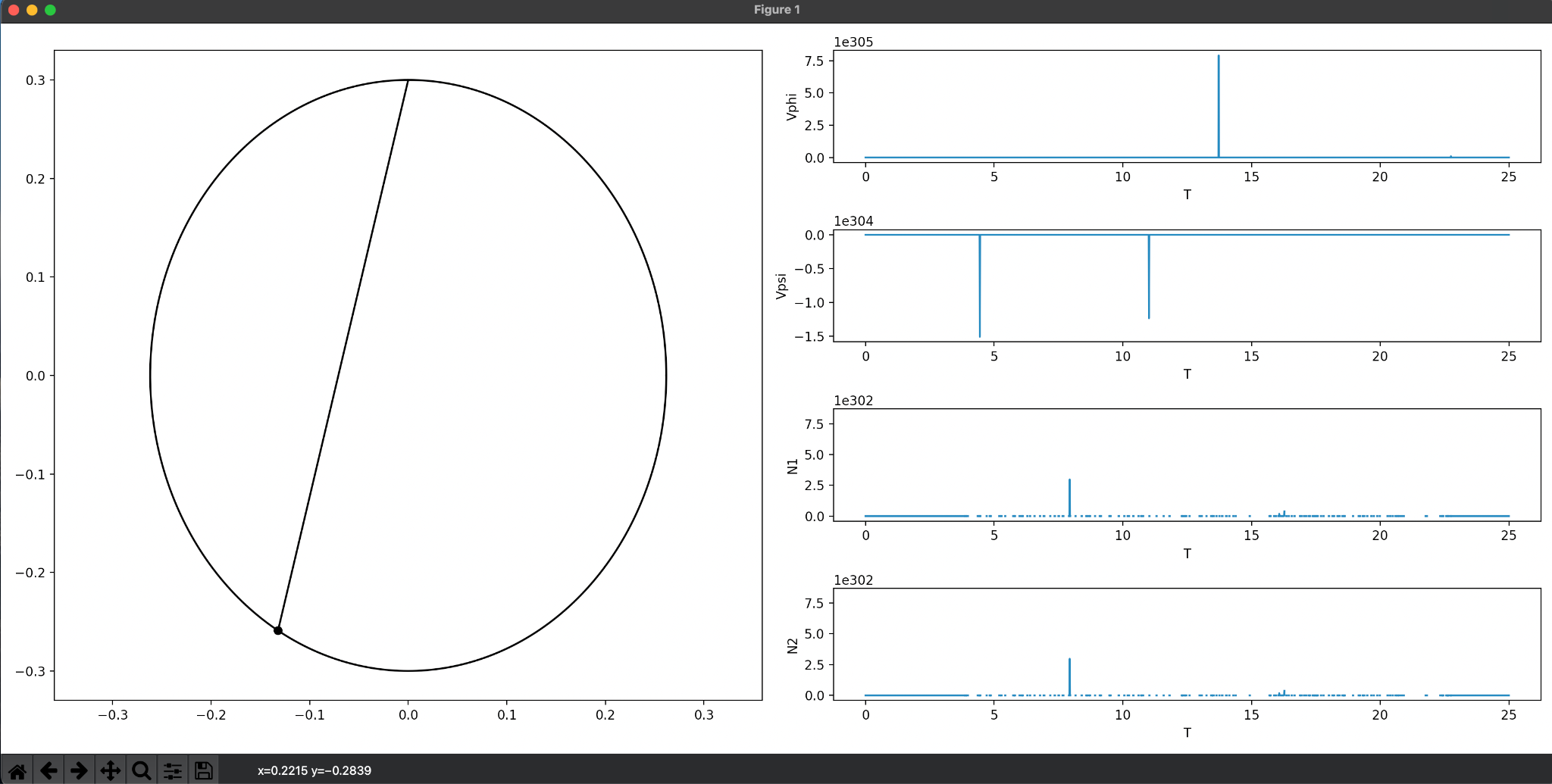
y0 = [0, pi/6, -0,5, 0]; -:

****

Результат: Когда бусина движется по обручу, она будет создавать силу, которая заставит обруч вращаться вокруг своей оси. Это происходит из-за сохранения углового момента.

1. m = 0,1 кг; M = 1 кг; R = 0,3 м; c = 1 Н/м; ; g = 9.81; phi = 0, psi = pi/6

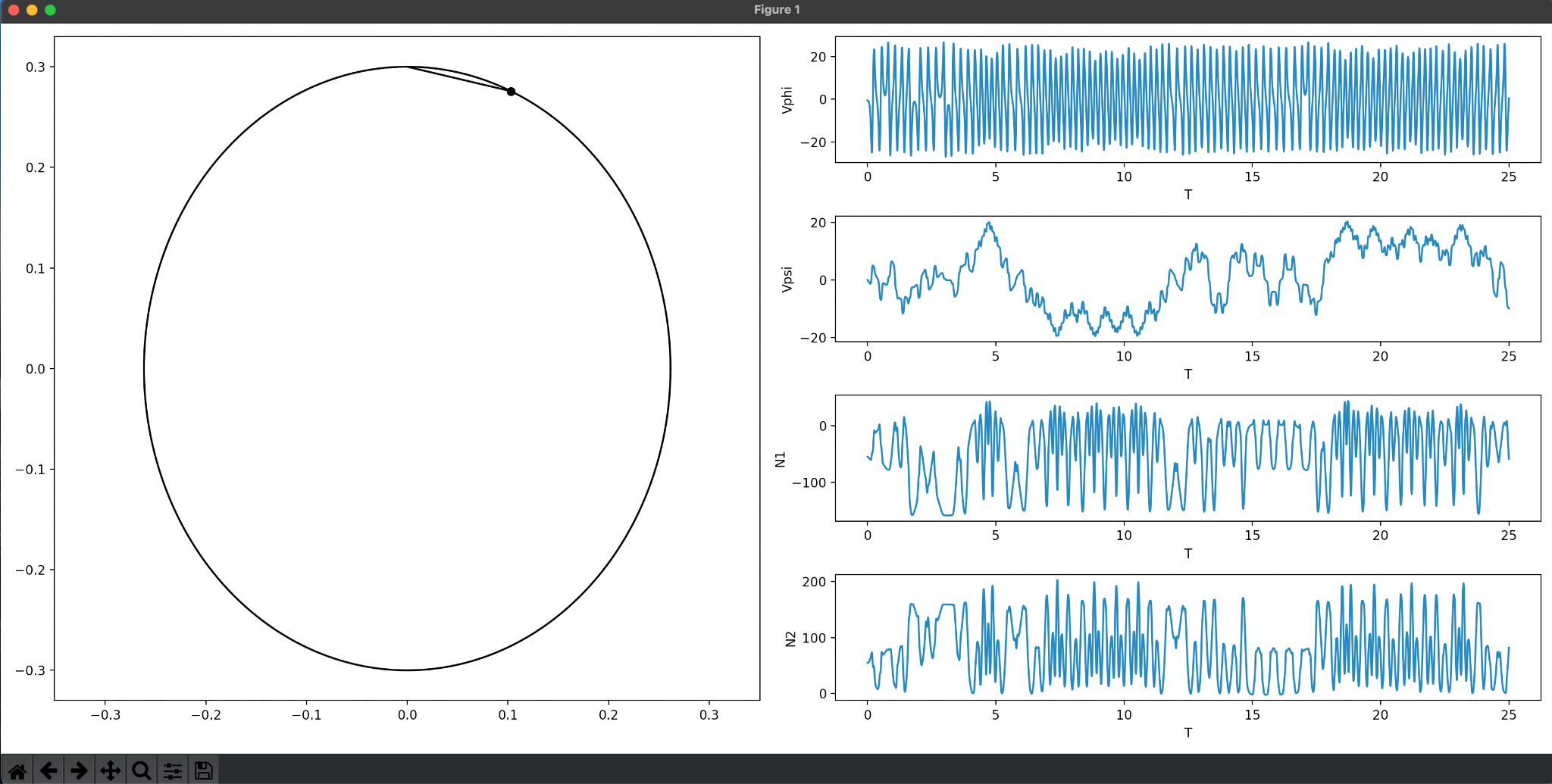
y0 = [0, pi/6, -0,5, 0]:

****

Результат: Когда бусина движется по обручу, она будет создавать силу, которая заставит обруч вращаться вокруг своей оси. Это происходит из-за сохранения углового момента. При уменьшении жесткости пружины - обруч будет вращаться медленнее, так как возвратная сила пружины будет меньше.

1. m = 0,1 кг; M = 1 кг; R = 0,3 м; c = 200 Н/м; ; g = 9.81; phi = 0, psi = pi/6

y0 = [0, pi/6, -0,5, 0]:

****

Результат: Когда бусина движется по обручу, она будет создавать силу, которая заставит обруч вращаться вокруг своей оси. Это происходит из-за сохранения углового момента. При увеличении жесткости пружины - обруч будет вращаться быстрее, так как возвратная сила пружины будет больше.

Вывод: В процессе выполнения данной лабораторной работы я познакомился с уравнениями Лагранжа 2 рода и использовал полученные знания для написания программы анимации ‘системы из своего варианта задания. Данные уравнения - наиболее удобный и совершенный способ составления уравнений движения механических систем. Число таких уравнений минимально и равно числу степеней свободы механической системы, что позволяет легко представлять их компьютеру.