

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ОТЧЕТ
О ВЫПЛЮНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ
«ДИНАМИКА СИСТЕМЫ»
ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И
ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»
ВАРИАНТ ЗАДАНИЯ №12

Выполнил(а) студент группы М8О-207Б-22

Брюханов З. Д._____
подпись, дата

Проверил и принял

Зав. каф. 802, Бардин Б.С._____
подпись, дата

с оценкой _____

Москва, 2023

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.

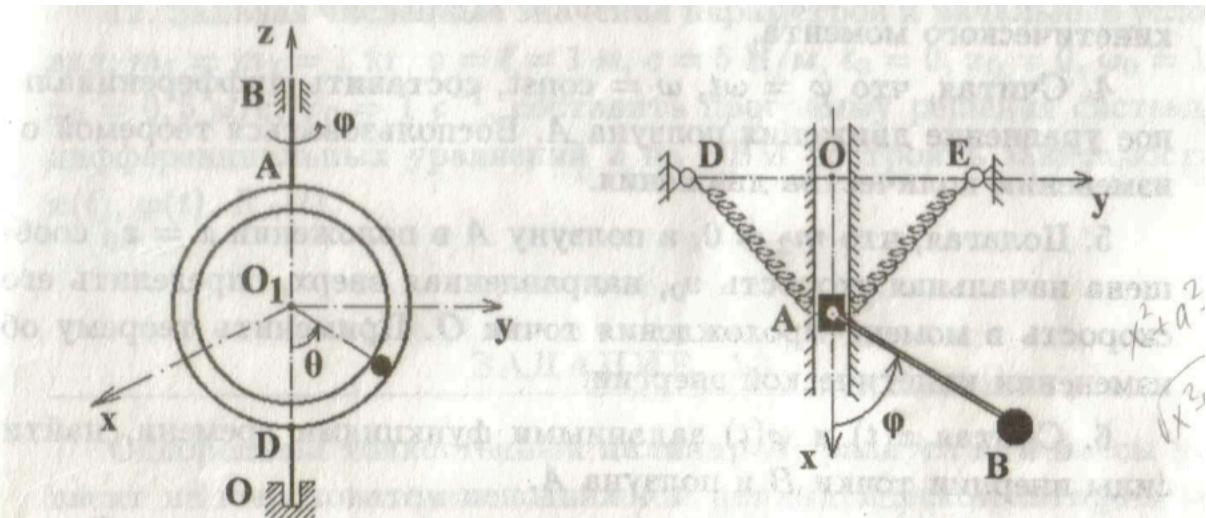


Рис. 11

Рис. 12

ЗАДАНИЕ 12

К ползуну A массы m_1 , помещенному между вертикальными направляющими, прикреплены две пружины жесткости с каждой, расположенные в вертикальной плоскости (рис. 12). Невесомый стержень AB длины ℓ , на конце которого укреплен груз B массы m_2 , соединен шарнирно с ползуном A . Стержень AB и груз B движутся в вертикальной плоскости. Концы пружин D и E расположены на одной горизонтали, причем $DO = OE = a$. Когда ползун A находится на линии DE (в точке O), пружины не напряжены.

10. Используя уравнения Лагранжа второго рода, показать, что дифференциальные уравнения движения системы имеют вид

$$(m_1 + m_2)(\ddot{x} - g) - m_2 \ell (\ddot{\varphi} \sin \varphi + \dot{\varphi}^2 \cos \varphi) = -2cx [1 - (a^2 / \sqrt{x^2 + a^2})], \quad \ddot{x} \sin \varphi - \ell \ddot{\varphi} = g \sin \varphi.$$

12. Задавая численные значения параметров и начальные условия: $m_1 = m_2 = 1$ кг, $a = \ell = 1$ м, $c = 5$ Н/м, $t_0 = 0$, $x_0 = 0$, $\varphi_0 = 1$, $\dot{x}_0 = 0,5$ м/с, $\dot{\varphi}_0 = 1$ с⁻¹, составить программу решения системы дифференциальных уравнений и на ЭВМ построить зависимости $x(t)$, $\varphi(t)$, $R_A(t)$.

Код программы

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from matplotlib.patches import Rectangle
from scipy.integrate import odeint
import sympy as sp
# Функция, описывающая систему 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 системы
a = 1 # Длина АО = ОЕ = a
lAB = 1 # Длина линии АВ
sA = 0.3 # Размер объекта А
mA = 1 # Масса А
mB = 1 # Масса В
g = 9.81 # Ускорение свободного падения
k = 15 # Коэффициент жесткости пружины
# Определение символов и функций SymPy
t = sp.Symbol('t')
xA = sp.Function('x')(t)
phi = sp.Function('phi')(t)
VA = sp.Function('V')(t)
omB = sp.Function('om')(t)
# Построение уравнений Лагранжа
# Выражение для квадрата скорости груза В относительно точки О
VB2 = VA ** 2 + omB ** 2 * lAB ** 2 - 2 * VA * omB * lAB * sp.sin(phi)
# Момент инерции груза В относительно точки О
JB = (mB * lAB ** 2) / 3
# Кинетическая энергия груза В
EkinB = (mB * VB2) / 2 + (JB * omB ** 2) / 2
# Кинетическая энергия массы А
EkinA = (mA * VA ** 2) / 2
# Полная кинетическая энергия системы
Ekin = EkinA + EkinB
# Переменная для выражения деформации пружин
delta_x = sp.sqrt(a ** 2 + xA ** 2) - a
# Потенциальная энергия пружин
EpotStrings = k * delta_x ** 2
# Потенциальная энергия массы А
EpotA = -mA * g * xA
# Потенциальная энергия груза В
```

```

EpotB = -mB * g * (xA + lAB * sp.cos(phi))
# Полная потенциальная энергия системы
Epot = EpotStrings + EpotA + EpotB
# Лагранжиан (разность кинетической и потенциальной энергии)
L = Ekin - Epot
# Уравнение Лагранжа для координаты xA
ur1 = sp.diff(sp.diff(L, VA), t) - sp.diff(L, xA)
# Уравнение Лагранжа для угла поворота omB
ur2 = sp.diff(sp.diff(L, omB), t) - sp.diff(L, phi)
# Коэффициенты уравнения для VA
a11 = ur1.coeff(sp.diff(VA, t), 1)
a12 = ur1.coeff(sp.diff(omB, t), 1)
# Коэффициенты уравнения для omB
a21 = ur2.coeff(sp.diff(VA, t), 1)
a22 = ur2.coeff(sp.diff(omB, t), 1)
# Свободные члены уравнений
b1 = -(ur1.coeff(sp.diff(VA, t), 0)).coeff(sp.diff(omB, t), 0).subs([(sp.diff(xA, t),
VA), (sp.diff(phi, t), omB)])
b2 = -(ur2.coeff(sp.diff(VA, t), 0)).coeff(sp.diff(omB, t), 0).subs([(sp.diff(xA, t),
VA), (sp.diff(phi, t), omB)])
# Определитель матрицы коэффициентов уравнений
det = a11 * a22 - a12 * a21
# Определитель матрицы коэффициентов для VA
det1 = b1 * a22 - b2 * a12
# Определитель матрицы коэффициентов для omB
det2 = a11 * b2 - b1 * a21
# Уравнение для ускорения VA
dVAdt = det1 / det
# Уравнение для ускорения omB
domBdt = det2 / det
# Подготовка данных для численного интегрирования
countOfFrames = 300
y0 = [0, 1, 0.5, 1] # x(0), phi(0), v(0), om(0)
T_start, T_stop = 0, 20
T = np.linspace(T_start, T_stop, countOfFrames)
fVA = sp.lambdify([xA, phi, VA, omB], dVAdt, "numpy")
fOmB = sp.lambdify([xA, phi, VA, omB], domBdt, "numpy")
sol = odeint(formY, y0, T, args=(fVA, fOmB))
XA_def = sp.lambdify(xA, xA)
XB_def = sp.lambdify([xA, phi], xA + lAB * sp.cos(phi))
YB_def = sp.lambdify(phi, lAB * sp.sin(phi))
XA = XA_def(sol[:, 0])
XB = XB_def(sol[:, 0], sol[:, 1])
YB = YB_def(sol[:, 1])

```

```

dphi = sol[:,3]
RA = mB * g * np.cos(YB) - mB * lAB * dphi ** 2 + k * (XA - a)
fig_for_graphs = plt.figure(figsize=[13, 7])
ax_for_graphs1 = fig_for_graphs.add_subplot(2, 2, 1)
ax_for_graphs1.plot(T, XA, color='Blue')
ax_for_graphs1.set_title("x(t)")
ax_for_graphs1.set_xlim=[T_start, T_stop])
ax_for_graphs1.grid(True)
ax_for_graphs2 = fig_for_graphs.add_subplot(2, 2, 3)
ax_for_graphs2.plot(T, YB, color='Red')
ax_for_graphs2.set_title("phi(t)")
ax_for_graphs2.set_xlim=[T_start, T_stop])
ax_for_graphs2.grid(True)
ax_for_graphs3 = fig_for_graphs.add_subplot(2, 2, 2)
ax_for_graphs3.plot(T, RA, color='Black')
ax_for_graphs3.set_title("RA(t)")
ax_for_graphs3.set_xlim=[T_start, T_stop])
ax_for_graphs3.grid(True)
# Построение графиков
fig = plt.figure(figsize=(10, 7))
# График 1: Положение объектов в пространстве
ax1 = fig.add_subplot(1, 1, 1)
ax1.axis('equal')
ax1.set(ylim=[-a, XA.max() + a], xlim=[min(-lAB, -a), max(lAB, a)])
ax1.set_xlabel('ось y')
ax1.set_ylabel('ось x')
ax1.invert_yaxis()
# Рисование точек D и E
ax1.plot(-a, 0, marker='o', color='black')
ax1.plot(a, 0, marker='o', color='black')
# Рисование линий, между которыми находится A
ax1.plot([-sA / 2, -sA / 2], [XA.min(), XA.max()], linestyle='-.', color='black')
ax1.plot([sA / 2, sA / 2], [XA.min(), XA.max()], linestyle='-.', color='black')
# Рисование начальных положений
# Рисование объекта A
PA = ax1.add_patch(Rectangle(xy=[-sA / 2, XA[0] - sA / 2], width=sA, height=sA,
color='g'))
# Рисование объекта B
PB, = ax1.plot(YB[0], XB[0], marker='o', color='r')
# Рисование линии AB
PAB, = ax1.plot([0, YB[0]], [XA[0], XB[0]], 'black')
# Рисование линий DA и EA
PDA, = ax1.plot([-a, 0], [0, XA[0]], linestyle='--', color='m')
PEA, = ax1.plot([a, 0], [0, XA[0]], linestyle='--', color='m')
# Функция для пересчета положений

```

```

def anima(i):
    PA.set(xy=[-sA / 2, XA[i] - sA / 2])
    PB.set_data(YB[i], XB[i])
    PAB.set_data([0, YB[i]], [XA[i], XB[i]])
    PDA.set_data([-a, 0], [0, XA[i]])
    PEA.set_data([a, 0], [0, XA[i]])
    return PAB, PDA, PEA, PA, PB
# Функция для анимации
anim = FuncAnimation(fig, anima, frames=countOfFrames, interval=100,
blit=True)
plt.show()

```

Описание

Программа начинается с определения функции `formY`, которая представляет систему дифференциальных уравнений. Эта функция описывает движение маятника и пружины, принимая входные параметры и возвращая значения производных координат и скоростей.

Далее идет блок определения параметров системы, таких как длины маятника и пружины, размеры объектов, массы, ускорение свободного падения, и коэффициент жесткости пружины. Символьные переменные `t`, `xA`, `phi`, `VA`, `omB` используются для создания уравнений Лагранжа и определения динамики системы.

Построение уравнений Лагранжа представляет собой описание кинетической и потенциальной энергии системы. В этом блоке создаются уравнения, описывающие движение объектов А и В, а также учитывается энергия пружины.

Следующий блок подготавливает данные для численного интегрирования. Задаются начальные условия и численно решаются уравнения движения для получения временных рядов координат и скоростей.

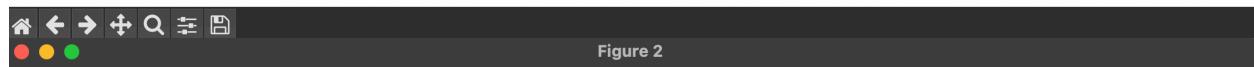
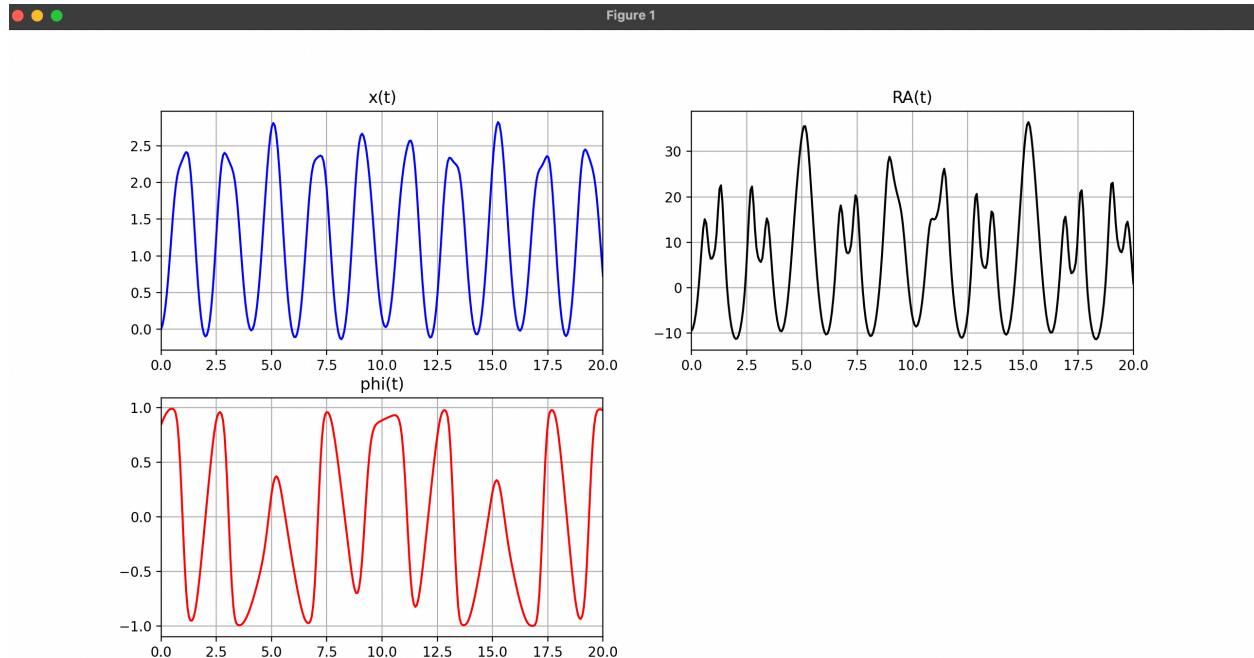
Затем программа использует полученные численные результаты для построения графиков, отражающих изменение координат, угла поворота и реакции пружины во времени. Два графика отображают изменения координаты `xA` и угла поворота `phi`, а также реакцию пружины `RA`.

Последний блок строит анимацию движения системы. График содержит объекты А и В, линию, соединяющую их, и линии, представляющие начальные положения объектов и направление пружины. Анимация обновляет положение этих объектов с течением времени, визуализируя динамику системы.

Эта программа позволяет исследовать взаимодействие между маятником, пружиной и грузом, а также влияние начальных условий на движение системы.

Результат работы программы

Параметры: $a = 1$; $lab = 1$; $mA = 1$; $mB = 1$; $g = 9.81$; $k = 15$; $x0 = 0$; $\phi i0 = 1$; $v0 = 0.5$; $om0 = 1$.



Параметры: $a = 2$; $lab = 1$; $mA = 1$; $mB = 50$; $g = 9.81$; $k = 30$; $x0 = 3$; $\phi_0 = 0$; $v_0 = 0$; $om_0 = 1$.

Figure 1

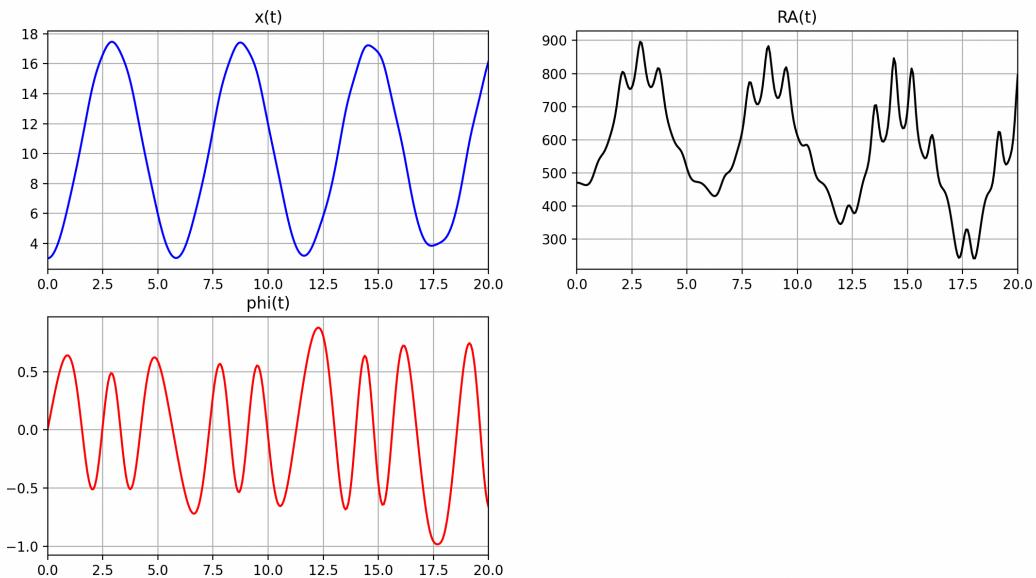


Figure 2

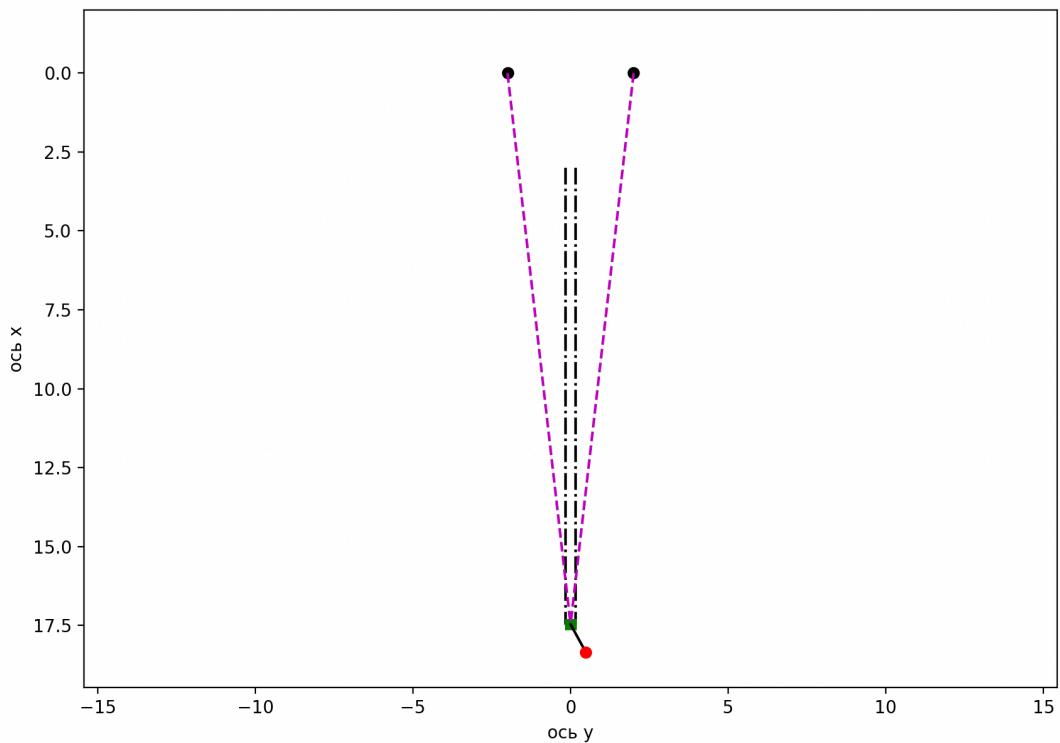
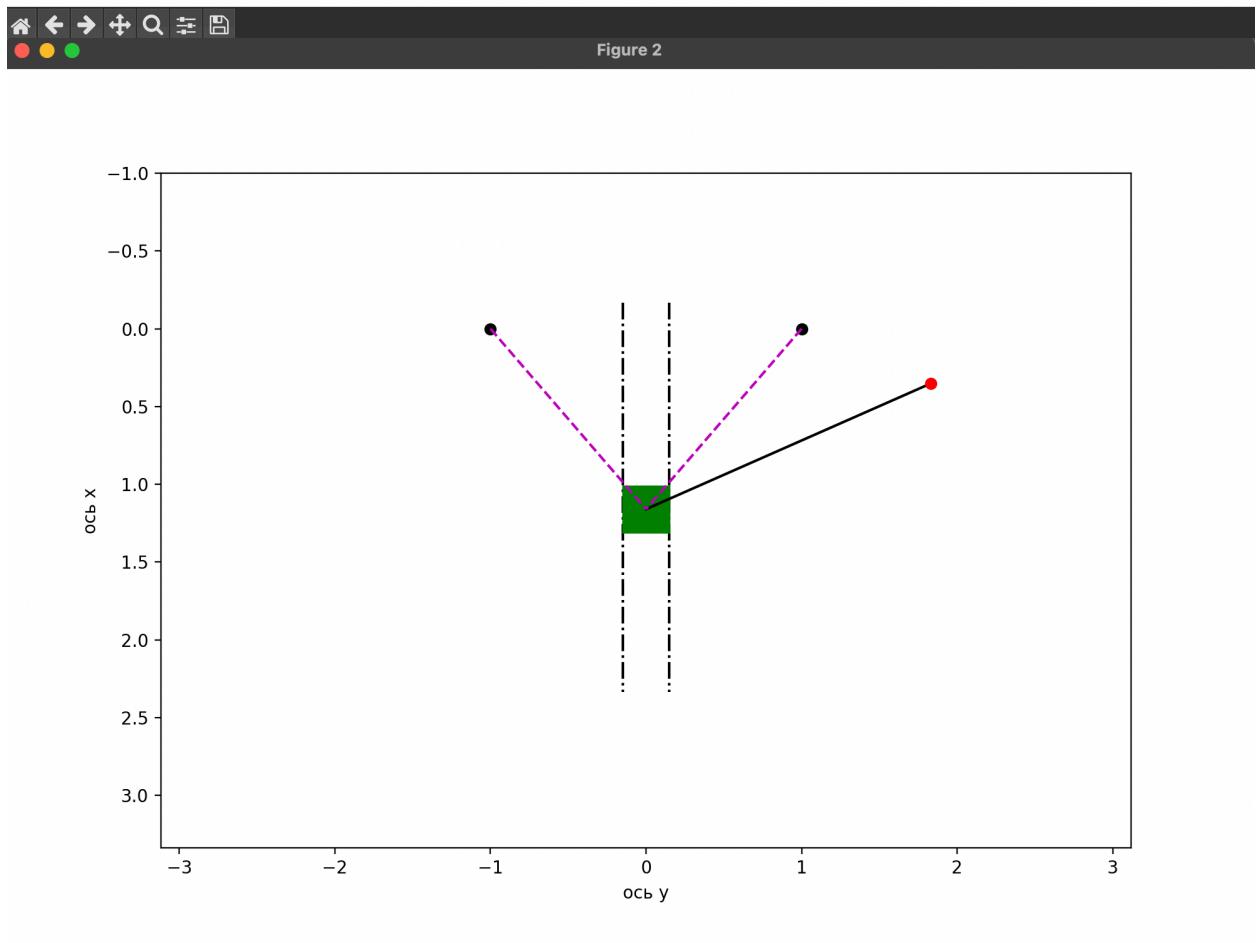
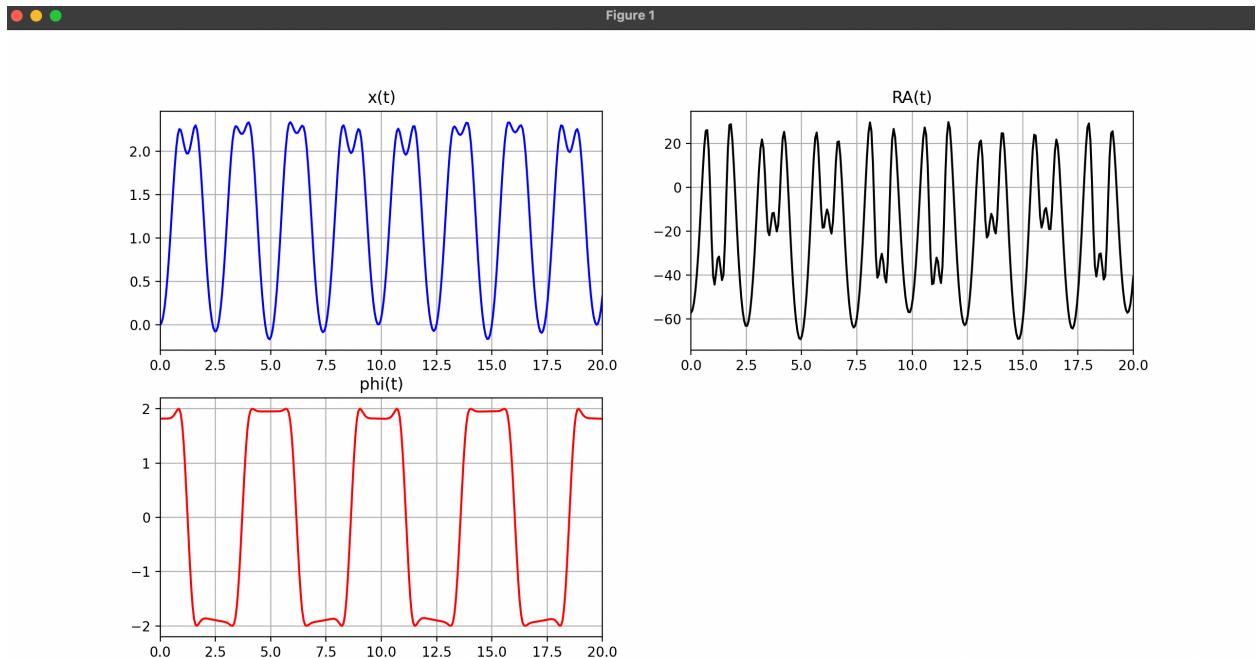


Figure 2

Параметры: $a = 1$; $lab = 2$; $mA = 4$; $mB = 3$; $g = 9.81$; $k = 50$; $x0 = 0$; $\phi_0 = 2$; $v0 = 0$; $om0 = 0$.



Вывод

В ходе выполнения данной работы мною был рассмотрен многокомпонентный маятник, состоящий из груза А, подвешенного на пружине, и груза В, подвешенного на маятнике. Для анализа динамики системы были построены уравнения Лагранжа, описывающие движение маятника.

Программа визуализирует движение грузов А и В, а также строит графики изменения координаты и угла между маятниками во времени и реакции пружины от времени.

В процессе выполнения лабораторной работы были применены знания динамики систем твердых тел, уравнений Лагранжа, численных методов решения дифференциальных уравнений, а также навыки визуализации движения многокомпонентных систем.

Кроме того, использование библиотеки SymPy позволило провести аналитический анализ системы, вывести уравнения движения и получить численное решение. Визуализация результатов помогла лучше понять и проиллюстрировать динамику системы во времени.