Отчет по моделированию.

Выполнили:

1 команда:

Селиховкина Е.И.

Мироненко Егор

2 команда:

Хлучин Г.В.

Гумбатов Влад

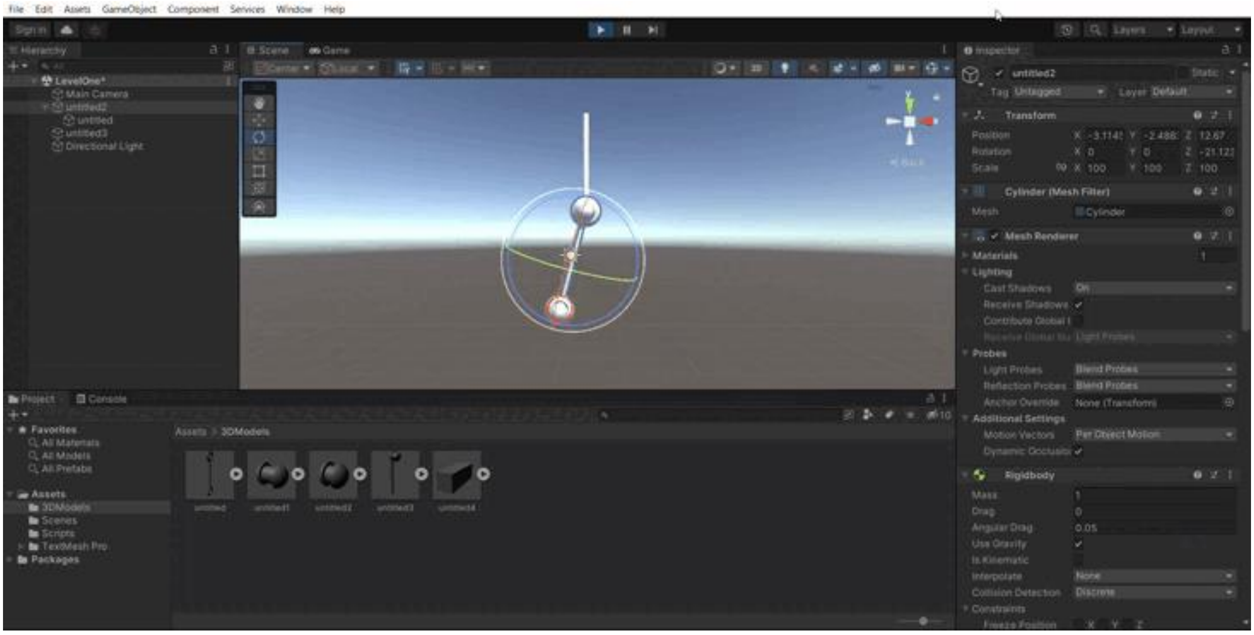
Первая команда выполняла моделирование физического маятника в unity + делала графики к физическому и математическому маятникам.

Вторая команда выполняла моделирование двойного маятника в unity + делал графики движения двойного физического маятника.

Обе команды работали на UI в unity

Отчет первой команды.

Графическая часть была выполнена в unity с помощью средств которые предоставляет движок. Были использованы такие физические события как rigitbody, joint



Формулы:

$\varphi(t) = \varphi_0 * \cos(\omega_0 t + \alpha)$ – уравнение гармонических колебаний, α – начальная фаза колебаний, φ_0 – амплитуда колебаний, ω_0 – собственнная циклическая частота

$\omega_0 = \sqrt{\frac{g}{l}}$ – циклическая частота

$T = \frac{2\pi}{\omega_0}$ – период колебаний

$\nu = \frac{1}{2\pi}\sqrt{\frac{g}{l}}$ – частота колебаний

$\mathrm{I} = ml^2$ – момент инерции

$$mL\ddot{\theta} = -mg\sin\theta$$

$$\ddot{\theta} + \frac{g}{L}\sin\theta = 0$$

Дифференциальные уравнения затухающих колебаний

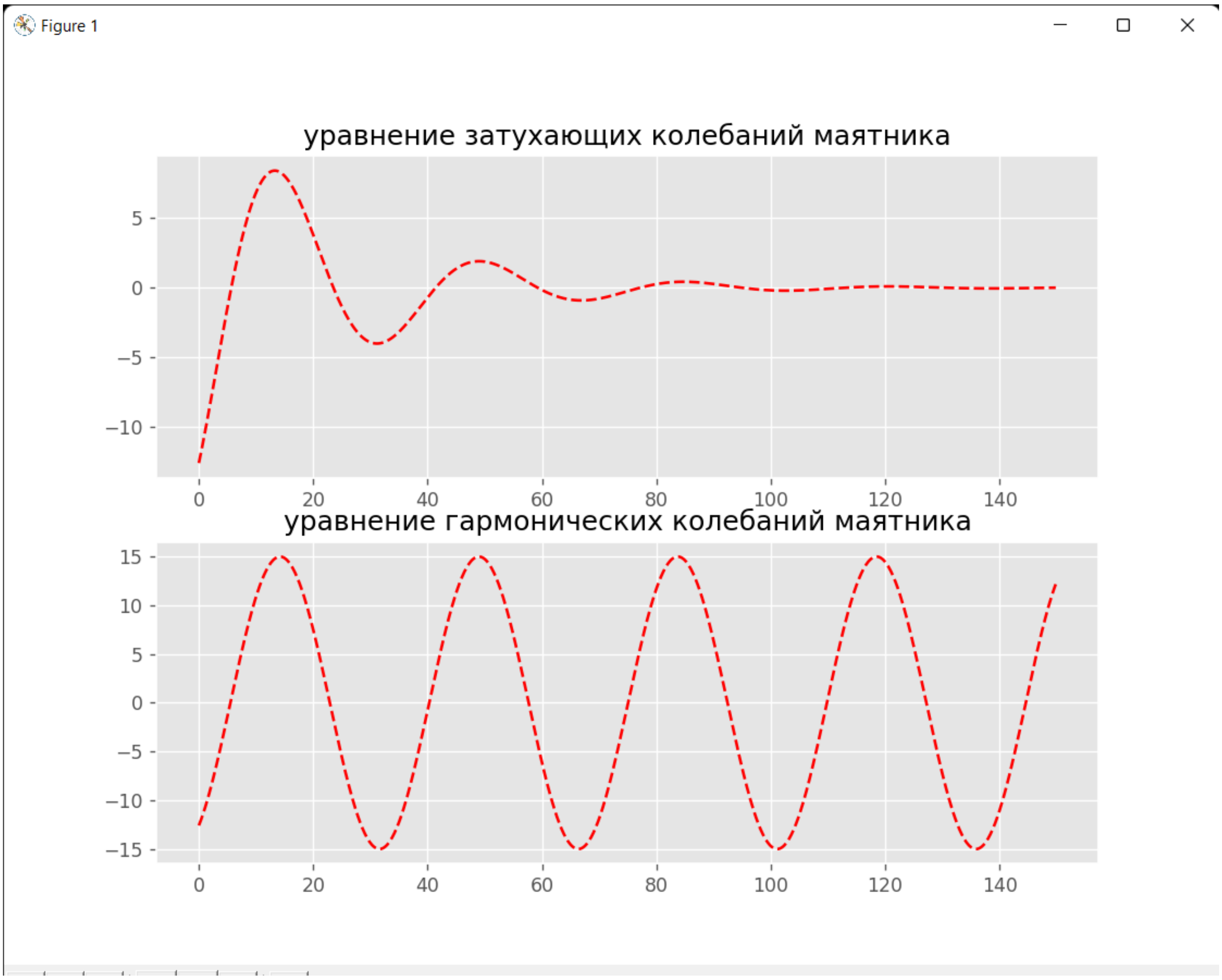
$$\frac{d^2x}{dt^2} + 2a\frac{dx}{dt} + \omega_0^2 x = 0$$

$x = A\, e^{-Bt} \cos(\omega t + \varphi_0)$

r – коэффициент трения

$\beta = \dfrac{r}{2m}$ – коэффициент затухания

$\omega = \sqrt{\omega_0^2 - \beta^2}$



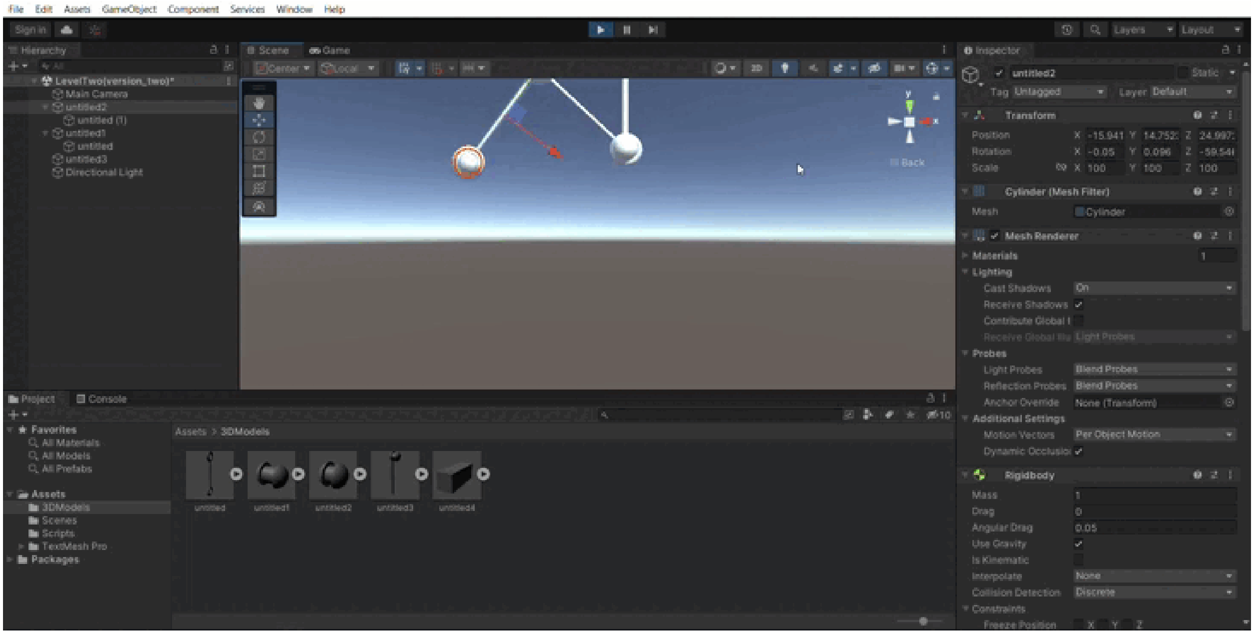
Код:

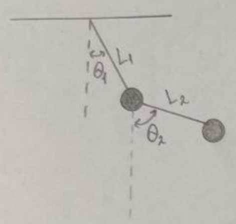
```
import matplotlib.pyplot as plt
import numpy as np
import math

plt.style.use('ggplot')


G = 9.8
e = 2.71

a = 10 #= int(input('введите начальную фазу колебаний: '))
```

```
l = 300 #= int(input('введите длину стержня: '))

A = 15 #= int(input('введите аммплитуду колебаний: '))

w0 = math.sqrt(G/l)

# массив времени t от 0 секунд до 5
t = np.arange(0., 150., 0.2)

f = A*np.cos(w0*t + a)

r = 1#= int(input('введите коэффициент трения: '))
m = 12 #= int(input('введите массу: '))

#коэффициент затухания
B = r / (2*m)

#циклическая частота
w = math.sqrt((w0**2 - B**2))

#уравнение затухающих колебаний
x = A * (e**(-B*t))*np.cos(w*t + a)

print('Гармонические колебания:')
print('собственная частота = ', w0)
print('период = ', 2 * 3.14 / w0)
print('частота колебаний = ', w0 / (6.28))
print('момент инерции = ', m * l**2)

print()
print('Затухающие колебания:')
print('собственная частота = ', w0)
print('период = ', 6.28 / w)
print('частота колебаний = ', w)
print('коэфициент затухания = ', B)

plt.subplot(2, 1, 1)
# отображение графиков
plt.plot(t, x, 'r--')
plt.title('уравнение затухающих колебаний маятника')

plt.subplot(2, 1, 2)
# отображение графиков  math.cos(w0*t + a)
plt.plot(t, f, 'r--')
plt.title('уравнение гармонических колебаний маятника')

plt.show()
```

Отчет второй команды.

Графическая часть была выполнена в unity с помощью средств которые предоставляет движок. Были использованы такие физические события как rigitbody, joint





$$L = \frac{1}{6} m\ell^2(\dot\theta_2^2 + 4\dot\theta_1^2 + 3\dot\theta_1\dot\theta_2\cos(\theta_1-\theta_2)) + \frac{1}{2} mg\ell(3\cos\theta_1 + \cos\theta_2)$$ – лагранжиан

$$p_{\theta_1} = \frac{\partial L}{\partial\dot\theta_1} = \frac{1}{6} m\ell^2(8\dot\theta_1 + 3\dot\theta_2\cos(\theta_1-\theta_2)))$$

$$p_{\theta_2} = \frac{\partial L}{\partial\dot\theta_2} = \frac{1}{6} m\ell^2(2\dot\theta_2 + 3\dot\theta_1\cos(\theta_1-\theta_2)))$$

энергия (не меняется)

$$\dot\theta_1 = \frac{6}{m\ell^2}\frac{2p_{\theta_1} - 3\cos(\theta_1-\theta_2)p_{\theta_2}}{16 - 9\cos^2(\theta_1-\theta_2)}$$

$$\dot\theta_2 = \frac{6}{m\ell^2}\frac{8p_{\theta_2} - 3\cos(\theta_1-\theta_2)p_{\theta_1}}{16 - 9\cos^2(\theta_1-\theta_2)}$$

$$\dot p_{\theta_1} = \frac{dL}{\partial\theta_1} = -\frac{1}{2} m\ell^2(\dot\theta_1\dot\theta_2\sin(\theta_1-\theta_2) + 3\frac{g}{\ell}\sin\theta_1)$$

$$\dot p_{\theta_2} = \frac{dL}{\partial\theta_2} = -\frac{1}{2} m\ell^2(-\dot\theta_1\dot\theta_2\sin(\theta_1-\theta_2) + \frac{g}{\ell}\sin\theta_2)$$

$$x_1 = \ell_1\sin\theta_1$$
$$x_2 = \ell_1\sin\theta_1 + \ell_2\sin\theta_2$$
$$y_1 = -\ell_1\cos\theta_1$$
$$y_2 = -\ell_1\cos\theta_1 - \ell_2\cos\theta_2$$

$$\dot x_1 = \ell_1\cos\theta_1\dot\theta_1$$
$$\dot x_2 = \ell_2\cos\theta_1\dot\theta_1 + \ell_2\cos\theta_2\dot\theta_2$$
$$\dot y_1 = \ell_1\sin\theta_1\dot\theta_1$$
$$\dot y_2 = \ell_1\sin\theta_1\dot\theta_1 + \ell_2\sin\theta_2\dot\theta_2$$

Potencial Energy

$$V = m_1 g y_1 + m_2 g y_2 = -m_1 g\ell_1\cos\theta_1 - m_2 g\ell_1\cos\theta_1 - m_2 g\ell_2\cos\theta_2$$

$$V = -(m_1+m_2)g\ell_1\cos\theta_1 - m_2 g\ell_2\cos\theta_2$$

Kinetic Energy

$$T = \frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 = \frac{1}{2} m_1(x_1^2 + y_1^2) + \frac{1}{2} m_2(x_2^2 + y_2^2)$$

$$\dot x_1 = \frac{dx_1}{dt} = v_{x_1} \quad \dot y_1 = \frac{dy_1}{dt} = v_{y_1}$$

$$\dot x_2 = \frac{dx_2}{dt} = v_{x_2} \quad \dot y_2 = \frac{dy_2}{dt} = v_{y_2}$$

$$T = \frac{1}{2} m_1[(\ell_1\cos\theta_1\dot\theta_1)^2 + (\ell_1\sin\theta_1\dot\theta_1)^2] +$$
$$+ \frac{1}{2} m_2[(\ell_1\cos\theta_1\dot\theta_1 + \ell_2\cos\theta_2\dot\theta_2)^2 + (\ell_1\sin\theta_1\dot\theta_1 + \ell_2\sin\theta_2\dot\theta_2)^2] =$$
$$= \frac{1}{2} m_1\ell_1^2\dot\theta_1^2 + \frac{1}{2} m_2\ell_1^2\dot\theta_1^2 + \frac{1}{2} m_2\ell_2^2\dot\theta_2^2 +$$
$$+ m_2\ell_1\ell_2(\cos\theta_1\cos\theta_2 + \sin\theta_1\sin\theta_2)\dot\theta_1\dot\theta_2$$

$$T = \frac{1}{2}(m_1+m_2)\ell_1^2\dot\theta_1^2 + \frac{1}{2} m_2\ell_2\dot\theta_2^2 + m_2\ell_1\ell_2\dot\theta_1\dot\theta_2\cos(\theta_1-\theta_2)$$

$$V = -(m_1+m_2)g\ell_1\cos\theta_1 - m_2 g\ell_2\cos\theta_2$$

Lagrangian

$$L = T - V$$

$$x_1'' = -\theta_1'^2 L_1 \sin\theta_1 + \theta_1'' L_1 \cos\theta_1$$
$$y_1'' = \theta_1'^2 L_1 \cos\theta_1 + \theta_1'' L_1 \sin\theta_1$$
$$x_2'' = x_1'' - \theta_2'^2 L_2 \sin\theta_2 + \theta_2'' L_2 \cos\theta_2$$
$$y_2'' = y_1'' - \theta_2'^2 L_2 \cos\theta_2 + \theta_2'' L_2 \sin\theta_2$$

$$m_1 x_1'' = -T_1 \sin\theta_1 + T_2 \sin\theta_2$$
$$m_1 y_1'' = T_1 \cos\theta_1 - T_2 \cos\theta_2 - m_1 g$$
$$m_2 x_2'' = -T_2 \sin\theta_2$$
$$m_2 y_2'' = T_2 \cos\theta_2 - m_2 g$$

$$T_1 \sin\theta_1 \cos\theta_1 = -\cos\theta_1 (m_1 x_1'' + m_2 x_2'')$$
$$T_1 \sin\theta_1 \cos\theta_1 = \sin\theta_1 (m_1 y_1'' + m_2 y_2'' + m_2 g + m_1 g)$$

$$T_2 \sin\theta_2 \cos\theta_2 = -\cos\theta_2 (m_2 x_2'')$$
$$T_2 \sin\theta_2 \cos\theta_2 = \sin\theta_2 (m_2 y_2'' + m_2 g)$$

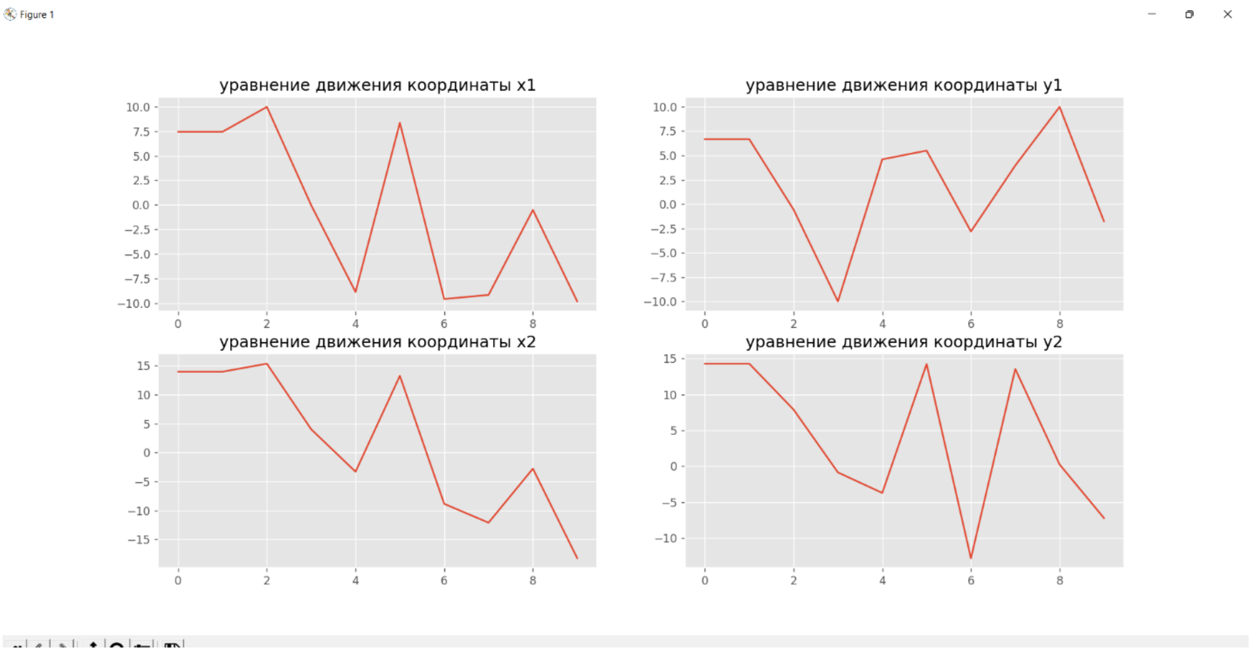
T - tension in the rod

$$\theta_1'' = \frac{-g(2m_1+m_2)\sin\theta_1 - m_2 g \sin(\theta_1 - 2\theta_2) - 2\sin(\theta_1-\theta_2) m_2 (\theta_2'^2 L_2 + \theta_1'^2 L_1 \cos(\theta_1-\theta_2))}{L_1(2m_1+m_2-m_2\cos(2\theta_1-2\theta_2))}$$

$$\theta_2'' = \frac{2\sin(\theta_1-\theta_2)(\theta_1'^2 \cdot L_1(m_1+m_2) + g(m_1+m_2)\cos\theta_1 + \theta_2'^2 L_2 m_2 \cos(\theta_1-\theta_2))}{L_2(2m_1+m_2-m_2\cos(2\theta_1-2\theta_2))}$$

$\theta_1 \mathrel{+}= \theta_1'$ $\quad$ $\theta_1' \mathrel{+}= \theta_1''$

$\theta_2 \mathrel{+}= \theta_2'$ $\quad$ $\theta_2' \mathrel{+}= \theta_2''$



Код:

```
import numpy as np
import sympy as smp
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from matplotlib import animation
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import PillowWriter
import decimal


class DoublePendulum:

    def __init__(self):
        self.l1 = 10
        self.l2 = 10
```

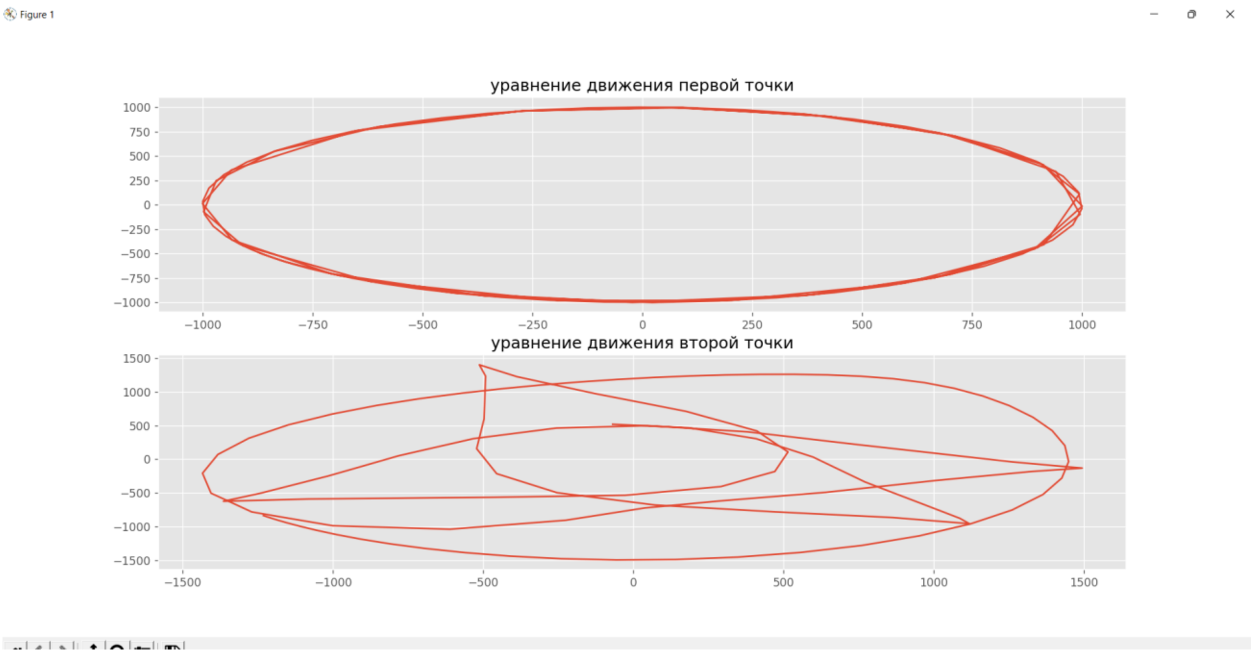
```
        self.m1 = 12
        self.m2 = 8
        self.theta1 = 40
        self.theta2 = 15
        self.theta1_ = 0
        self.theta2_ = 0
        self.g = 9.8


    def get_coordinates(self, t):
        numerator = (-self.g*(2*self.m1+self.m2)*math.sin(self.theta1) - 
self.m2*self.g*math.sin(self.theta1-2*self.theta2) - 2*math.sin(self.theta1-
self.theta2)*self.m2*((self.theta2_**2)*self.l2+(self.theta1_**2)*self.l1*mat
h.cos(self.theta1-self.theta2)))
        denominator = (self.l1*(2*self.m1+self.m2-
self.m2*math.cos(2*self.theta1-2*self.theta2)))

        theta1__ = numerator / denominator

        numerator = (2*math.sin(self.theta1-
self.theta2)*((self.theta1_**2)*self.l1*(self.m1+self.m2)+self.g*(self.m1+sel
f.m2)*math.cos(self.theta1)+(self.theta2_**2)*self.l2*self.m2*math.cos(self.t
heta1-self.theta2)))
        denominator = (self.l2*(2*self.m1+self.m2-
self.m2*math.cos(2*self.theta1-2*self.theta2)))

        theta2__ = numerator / denominator

        x1 = self.l1 * math.sin(self.theta1)
        y1 = -self.l1 * math.cos(self.theta1)

        x2 = x1 + self.l2 * math.sin(self.theta2)
        y2 = y1 - self.l2 * math.cos(self.theta2)

        self.theta1 += self.theta1_
        self.theta2 += self.theta2_

        self.theta1_ += theta1__
        self.theta2_ += theta2__

        return [
            x1,
            y1,
            x2,
            y2,
        ]



plt.style.use('ggplot')

t = np.arange(0., 10, 1)

double_pendulum = DoublePendulum()

x = np.sin(t)
arr_x1 = []
arr_y1 = []
arr_x2 = []
arr_y2 = []



for i in t:
```

```
    mass = double_pendulum.get_coordinates(t)
    arr_x1.append(mass[0])
    arr_y1.append(mass[1])
    arr_x2.append(mass[2])
    arr_y2.append(mass[3])


plt.subplot(2, 2, 1)
plt.plot(t, np.array(arr_x1))
plt.title('уравнение движения координаты x1')

plt.subplot(2, 2, 2)
plt.plot(t, np.array(arr_y1))
plt.title('уравнение движения координаты y1')

plt.subplot(2, 2, 3)
plt.plot(t, np.array(arr_x2))
plt.title('уравнение движения координаты x2')

plt.subplot(2, 2, 4)
plt.plot(t, np.array(arr_y2))
plt.title('уравнение движения координаты y2')

plt.show()
```

код после внесения правок:

```
import numpy as np
import sympy as smp
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from matplotlib import animation
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import PillowWriter
import decimal


class DoublePendulum:

    def __init__(self):
        self.l1 = 1000
        self.l2 = 500
        self.m1 = 12
        self.m2 = 8
        self.theta1 = 200
        self.theta2 = 150
        self.theta1_ = 0
        self.theta2_ = 0
        self.g = 10


    def get_coordinates(self, t):
        num1 = (-self.g*(2*self.m1+self.m2)*(float(math.sin(self.theta1))))
        num2 = (self.m2*self.g*(float(math.sin(self.theta1-2*self.theta2))))
        num3 = 2*(float(math.sin(self.theta1-self.theta2)))*self.m2
        num4 = (self.theta2_**2)*self.l2
        num5 = (self.theta1_**2)*self.l1*(float(math.cos(self.theta1-
self.theta2)))
        numerator = (num1 - num2 - num3*(num4+num5))
        denominator = (self.l1*(2*self.m1+self.m2-
self.m2*(float(math.cos(2*self.theta1-2*self.theta2)))))
```

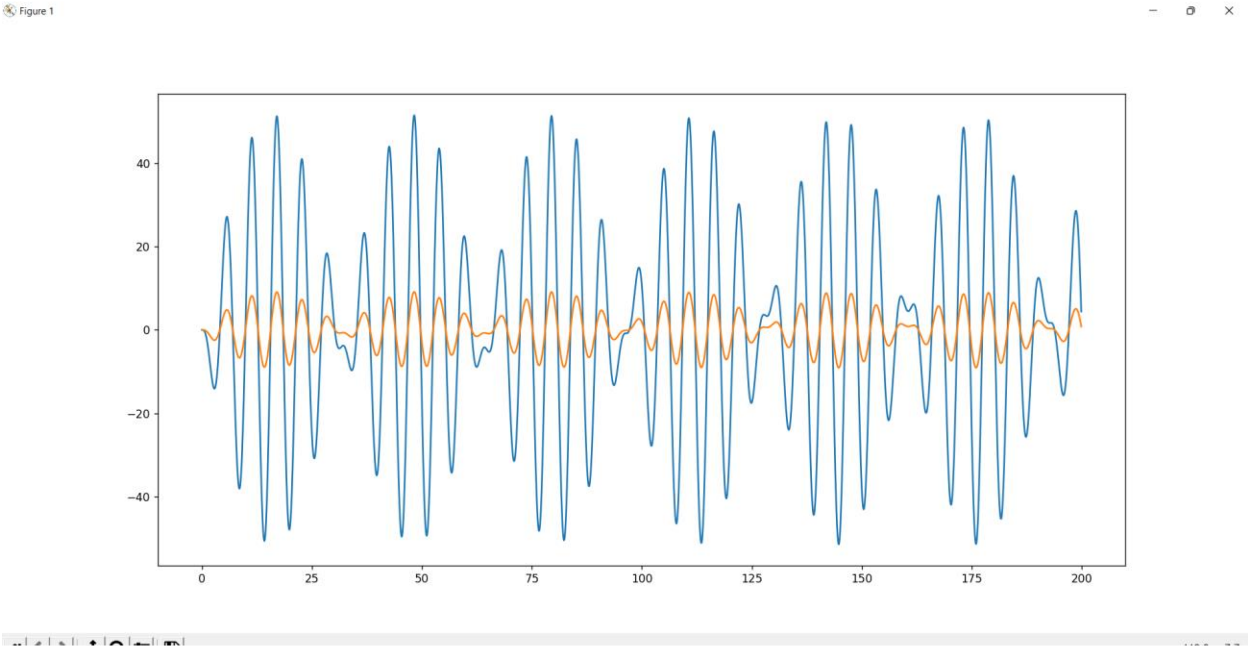
```
        theta1__ = (float(numerator / denominator))

        num1 = (2*(float(math.sin(self.theta1-
self.theta2)))*(self.theta1_**2)*self.l1*(self.m1+self.m2))
        num2 = (self.g*(self.m1+self.m2)*(float(math.cos(self.theta1))))
        num3 = (self.theta2_**2)*self.l2*self.m2*(float(math.cos(self.theta1-
self.theta2)))
        numerator = (float(num1+num2+num3))
        denominator = (float(self.l2*(2*self.m1+self.m2-
self.m2*(float(math.cos(2*self.theta1-2*self.theta2))))))

        theta2__ = (float(numerator / denominator))

        x1 = (self.l1 * (float(math.sin(self.theta1))))
        y1 = (-self.l1 * (float(math.cos(self.theta1))))

        x2 = (x1 + self.l2 * (float(math.sin(self.theta2))))
        y2 = (y1 - self.l2 * (float(math.cos(self.theta2))))

        self.theta1 += self.theta1_
        self.theta2 += self.theta2_

        self.theta1_ += theta1__
        self.theta2_ += theta2__

        return [
            x1,
            y1,
            x2,
            y2,
        ]



plt.style.use('ggplot')

t = np.arange(0., 10, 0.1)

double_pendulum = DoublePendulum()

x = np.sin(t)
arr_x1 = []
arr_y1 = []
arr_x2 = []
arr_y2 = []


for i in t:
    mass = double_pendulum.get_coordinates(t)
    arr_x1.append(mass[0])
    arr_y1.append(mass[1])
    arr_x2.append(mass[2])
    arr_y2.append(mass[3])

plt.subplot(2, 1, 1)
plt.plot(np.array(arr_x1), np.array(arr_y1))
plt.title('уравнение движения первой точки')

plt.subplot(2, 1, 2)
plt.plot(np.array(arr_x2), np.array(arr_y2))
plt.title('уравнение движения второй точки')

plt.show()
```



Код:

```
import math
import matplotlib.pyplot as plt
import numpy as np
import math

f1 = 15 #угол поворота первого маятника
f2 = 30 #угол поворота второго маятника

m1 = 10 #масса первого маятника
m2 = 320 #масса второго маятника

g = 9.8

l = 8 #длины стержней маятников

w0 = math.sqrt(g / l) #собственная циклическая частота

M = m1 / m2 #отношение масс маятников

w = w0*(1 + math.sqrt(M)/2) #циклическая частота

w_ = w0*(1 - math.sqrt(M)/2) #производная циклической частоты

t = np.arange(0., 200., 0.1) #массив времени

C = 10 #константа, получаемая при решении дифференциального уравнения, может
быть любой, по сути от нее зависят интегральные кривые

x1 = (C / (2 * math.sqrt(M)))*((np.sin(w*t)/w) - (np.sin(w_*t)/w_)) #расчет
координаты x для первого маятника

x2 = (C / 2 * ((np.sin(w*t)/w) - (np.sin(w_*t)/w_))) #расчет координаты x для
второго маятника

# отображение графиков
plt.plot(t, x1, t, x2)

plt.show()
```

график:



Отличие физического и математического маятников:

Математи́ческий ма́ятник — механическая система, состоящая из материальной точки, подвешенной на невесомой нерастяжимой нити или на невесомом стержне в поле тяжести. Период малых колебаний математического маятника длины l в поле тяжести с ускорением свободного падения g равен и не зависит от амплитуды и массы маятника. Плоский математический маятник со стержнем — система с одной степенью свободы.

Физический маятник - это реалистичная модель маятника; он имеет конечное тело и форму. Подвеска Простому маятнику нужна ступенька или веревка, чтобы подвешиваться к жесткой опоре. Физическому маятнику не нужна веревка для подвески. напряжение На струну действует сила натяжения, которая помогает объекту подвешиваться. Поскольку физическому маятнику не нужна веревка для подвески, натяжения не будет.

Амплитуда постоянная и там, и там. Так что разница в другом совсем в другом. В физическом маятнике груз уже нельзя считать точечным телом. Поэтому при составлении уравнения колебаний нужно брать не массу груза, а его момент инерции. То есть "длина" маятника - это не длина нити подвеса, а расстояние от центра тяжести тела до точки подвеса (приведённая длина).