Отчет по моделированию.

Выполнили:

1 команда:

Селиховкина Е.И.

Мироненко Егор

2 команда:

Хлучин Г.В.

Гумбатов Влад

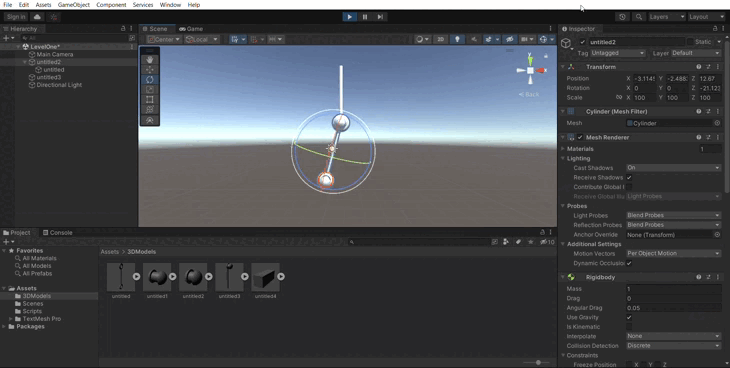
Первая команда выполняла моделирование физического маятника в unity + делала графики к физическому и математическому маятникам.

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

Обе команды работали на UI в unity

Отчет первой команды.

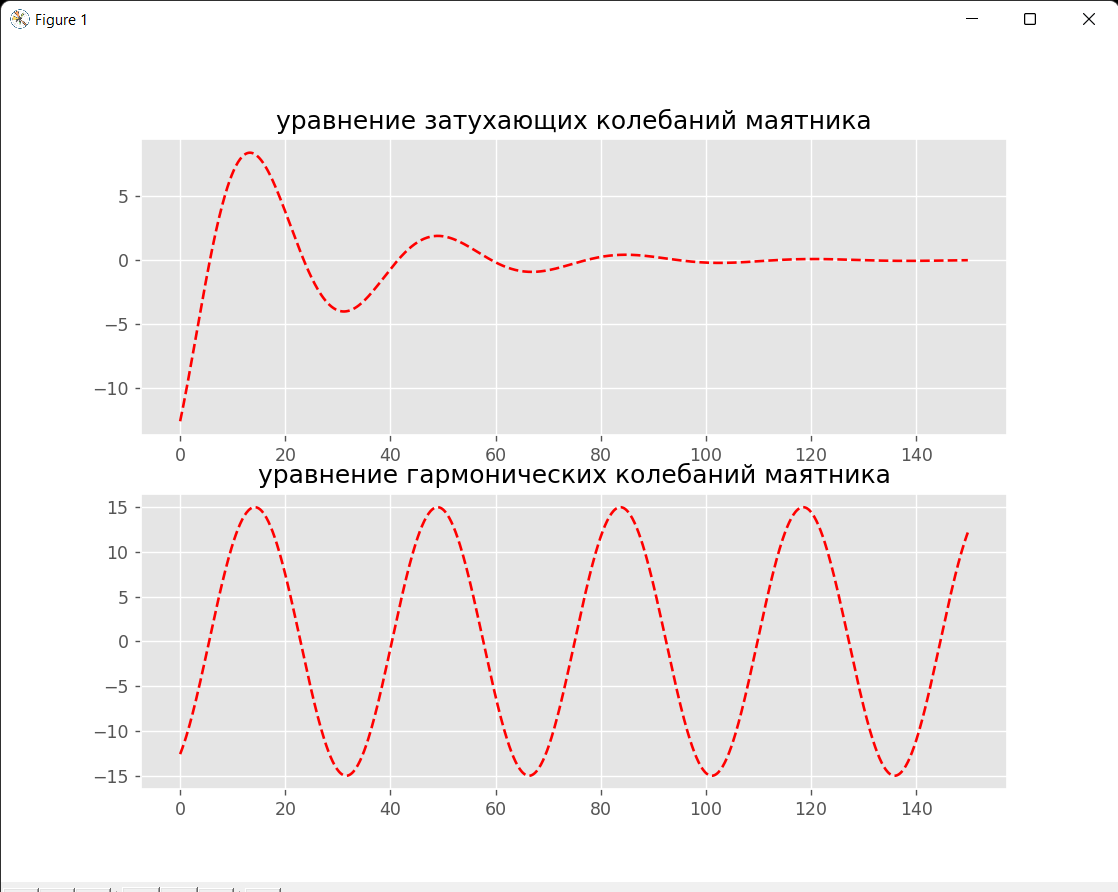
Графическая часть была выполнена в unity с помощью средств которые предоставляет движок. Были использованы такие физические события как rigitbody, joint



Формулы:

Дифференциальные уравнения затухающих колебаний

r – коэффициент трения

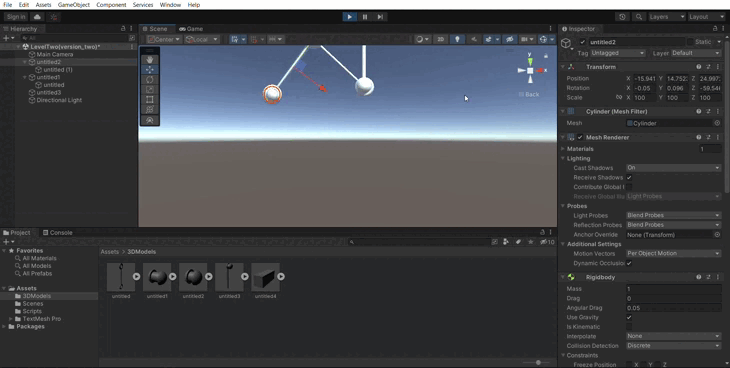
**

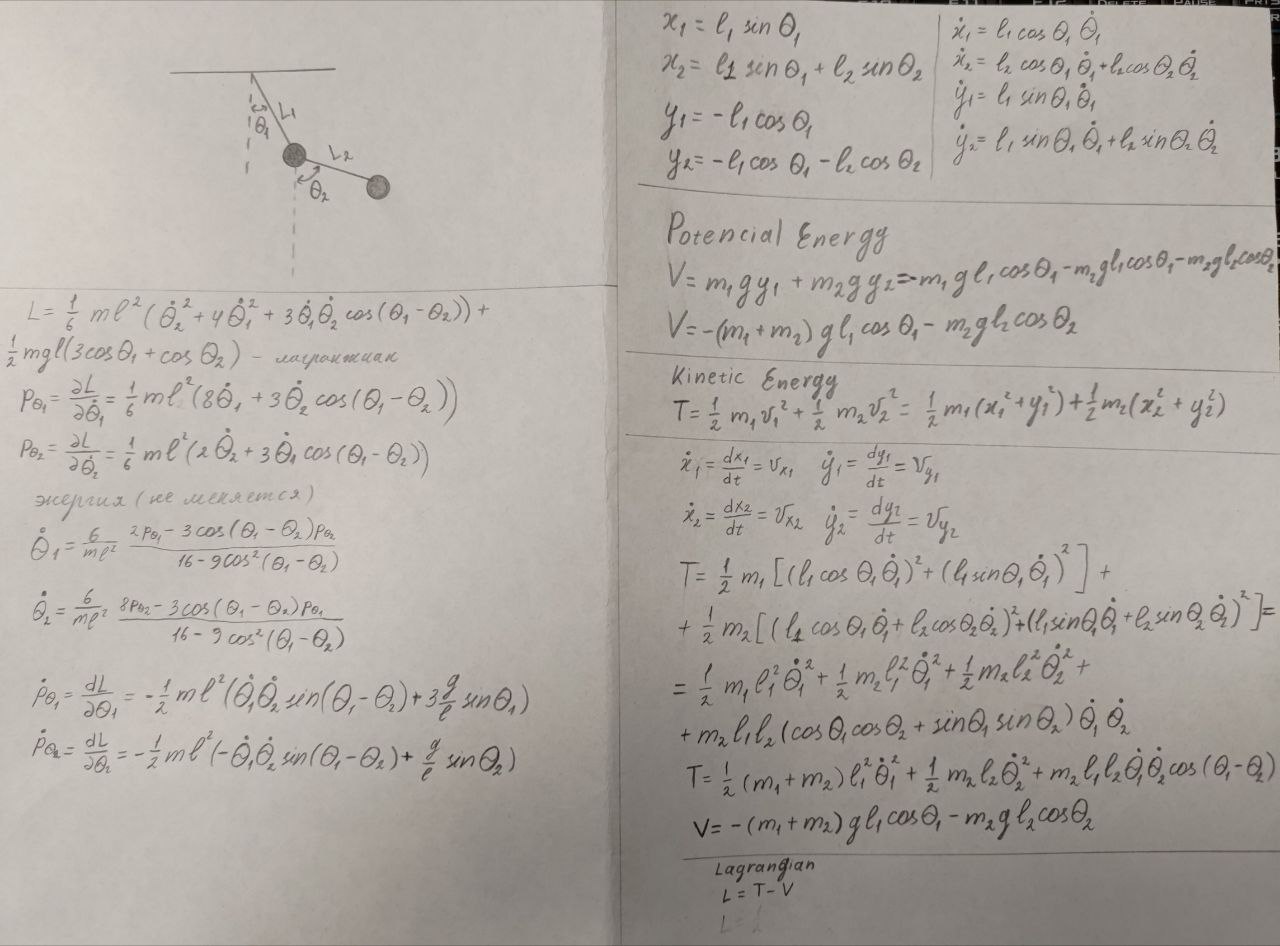
*Код:*

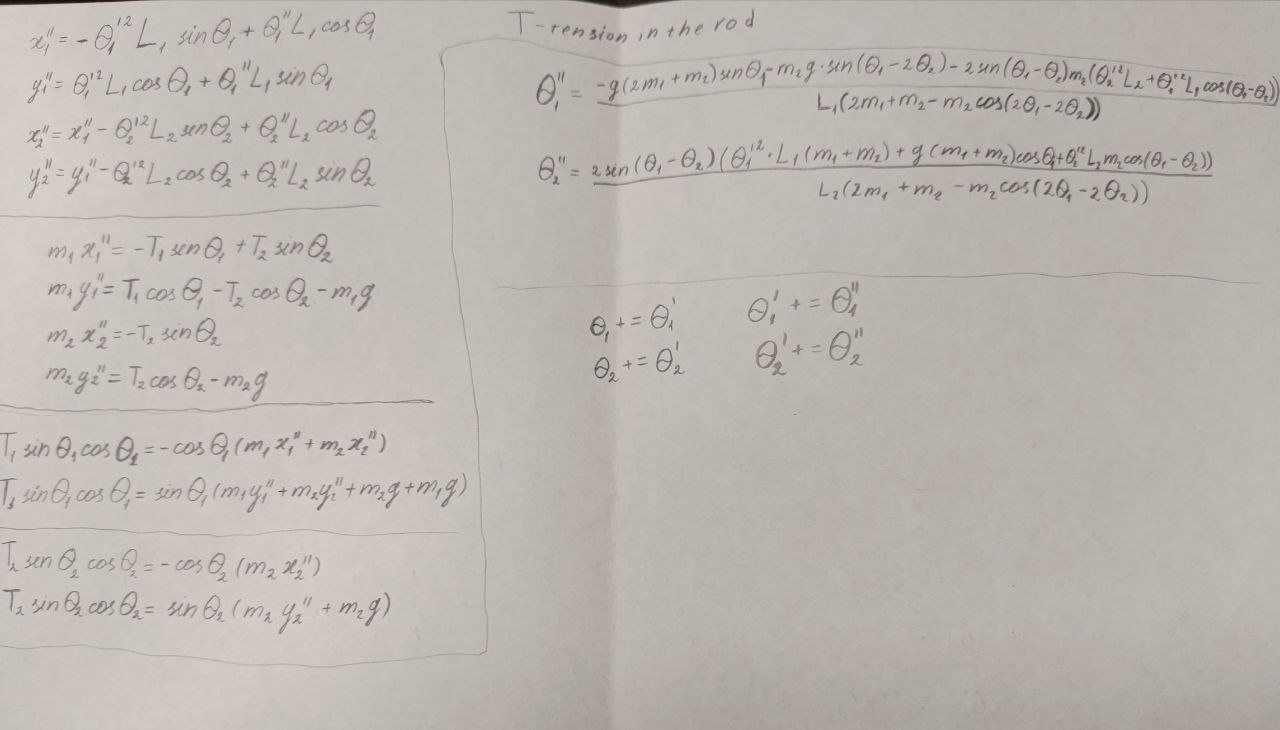
import matplotlib.pyplot as plt  
import numpy as np  
import math  
  
plt.style.use('ggplot')  
  
  
G = 9.8  
e = 2.71  
  
a = 10 #= int(input('введите начальную фазу колебаний: '))  
l = 300 #= int(input('введите длину стержня: '))  
  
A = 15 #= int(input('введите аммплитуду колебаний: '))  
  
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()

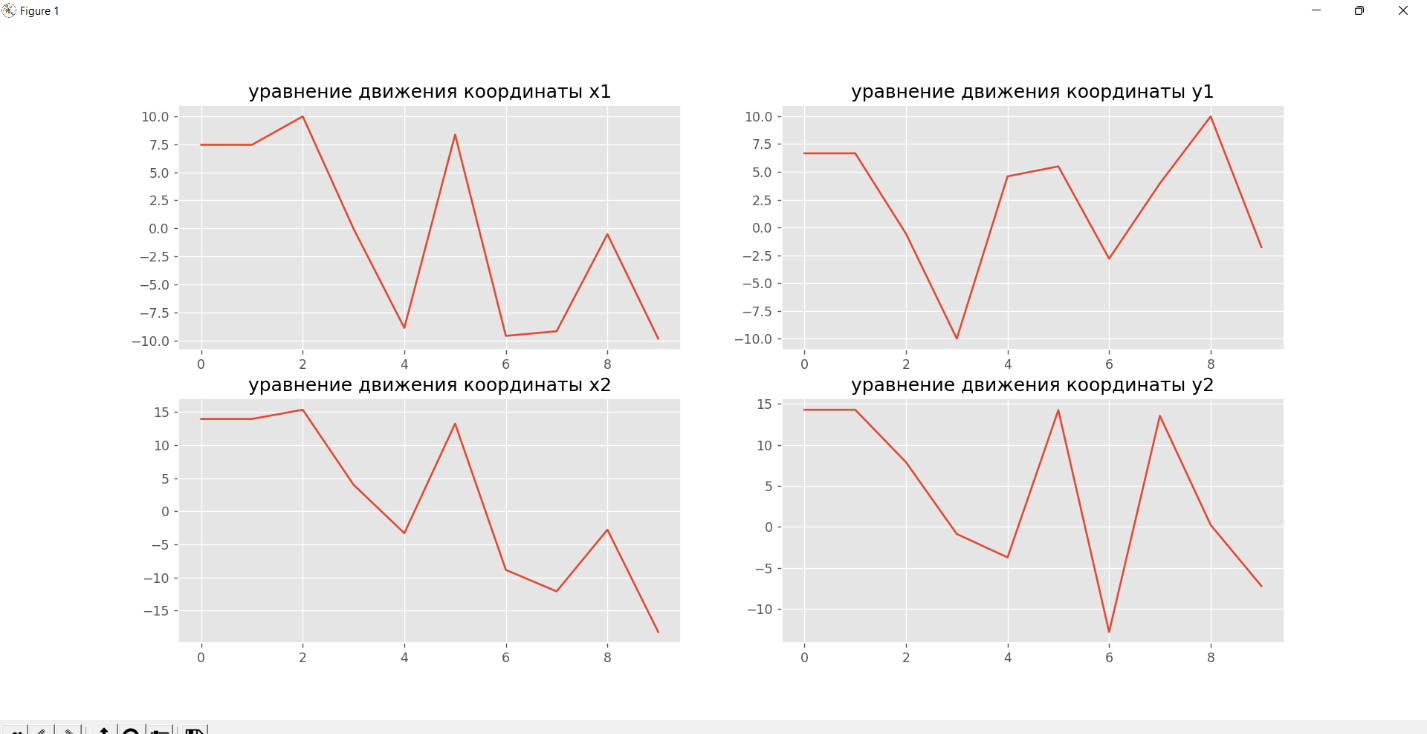
Отчет второй команды.

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



**

**

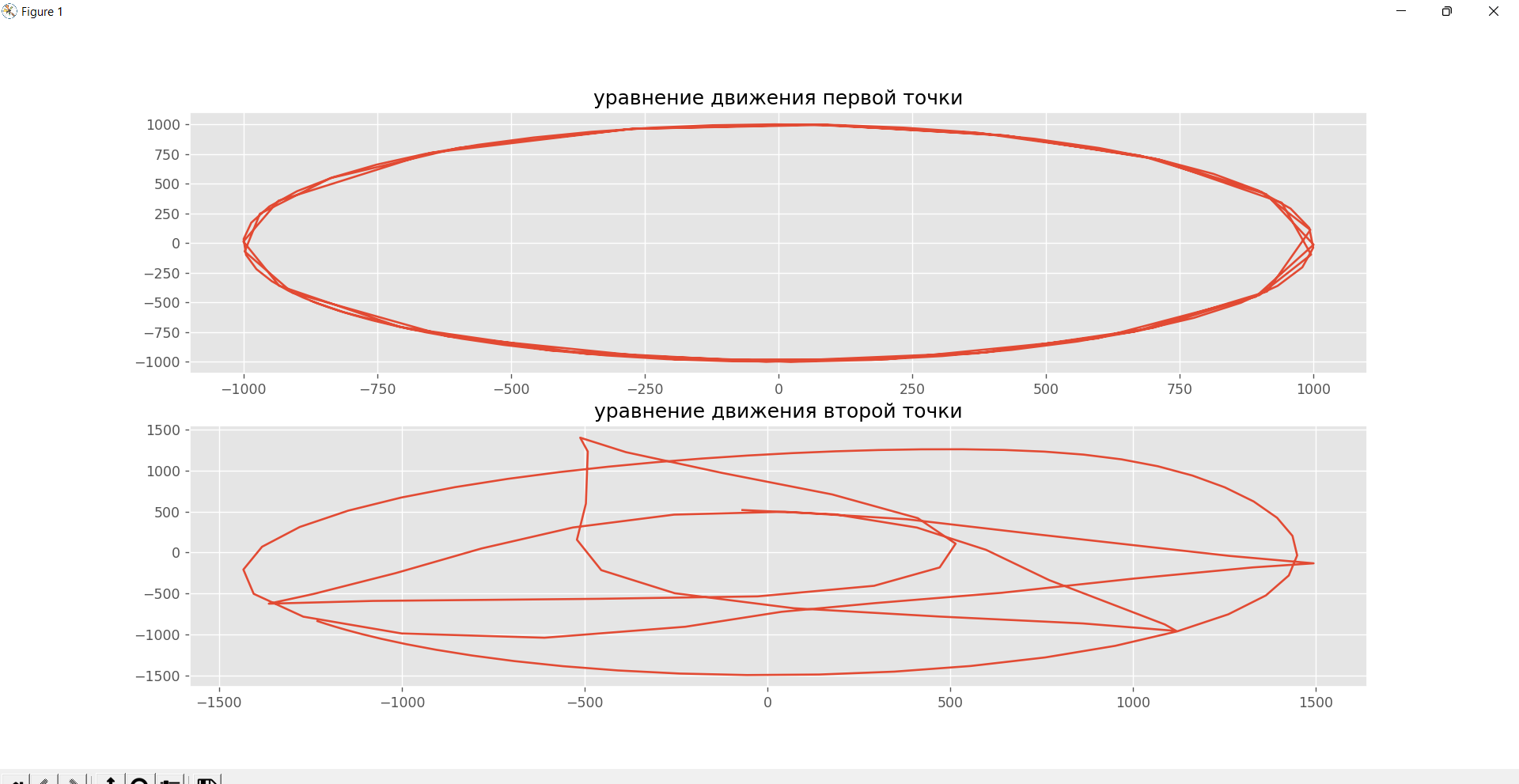


Код:

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\*math.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+self.m2)\*math.cos(self.theta1)+(self.theta2\_\*\*2)\*self.l2\*self.m2\*math.cos(self.theta1-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 х1')  
  
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 х2')  
  
plt.subplot(2, 2, 4)  
plt.plot(t, np.array(arr\_y2))  
plt.title('уравнение движения координаты y2')  
  
plt.show()

код после внесения правок:

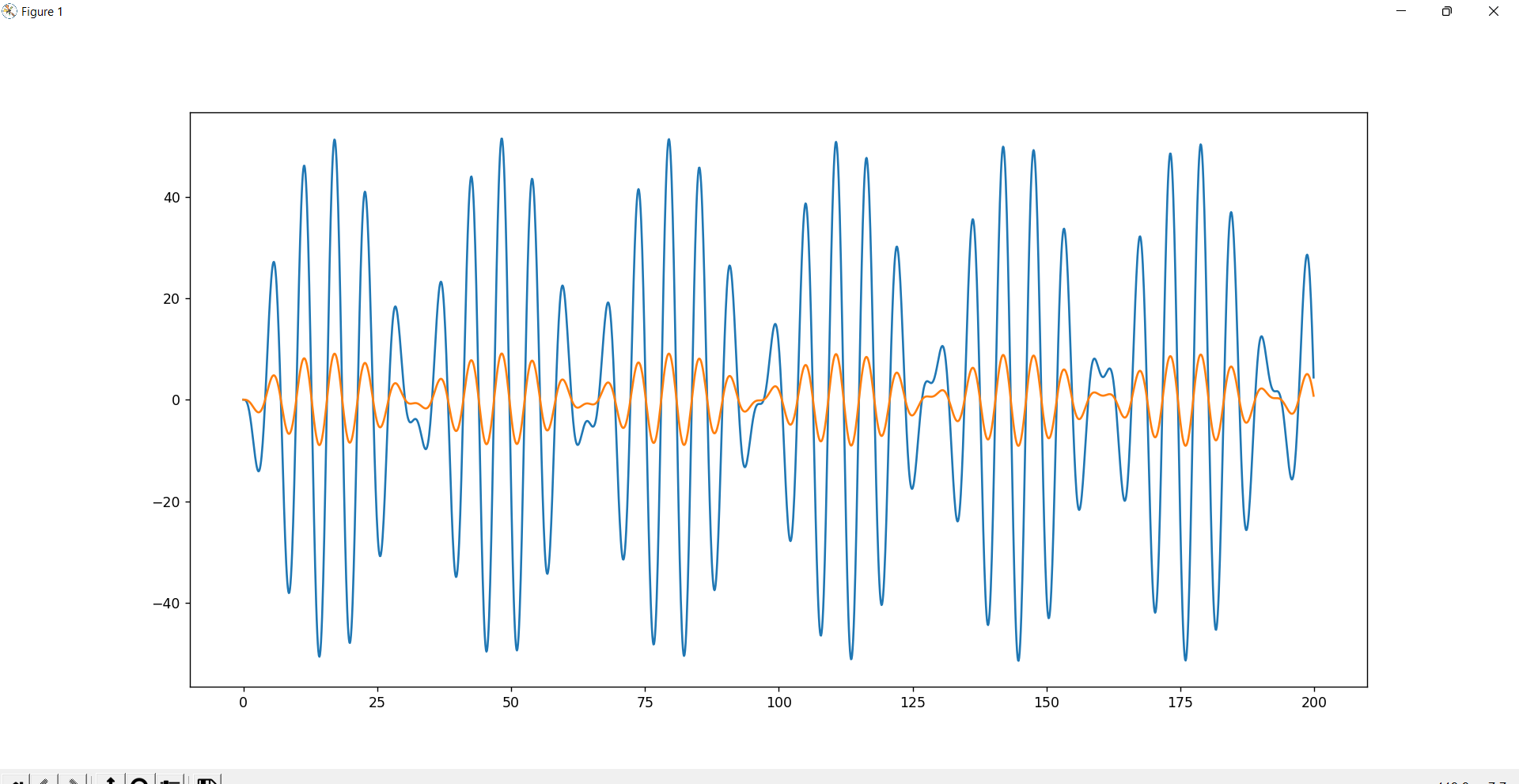
import numpy as np  
import sympy as smp  
from scipy.integrate import odeint  
import matplotlib.pyplot as plt  
from matplotlib import animation  
from mpl\_toolkits.mplot3d import Axes3D  
from matplotlib.animation import PillowWriter  
import decimal  
  
  
class DoublePendulum:  
  
 def \_\_init\_\_(self):  
 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 х для первого маятника  
  
x2 = (C / 2 \* ((np.sin(w\*t)/w) - (np.sin(w\_\*t)/w\_))) #расчет координаты х 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).