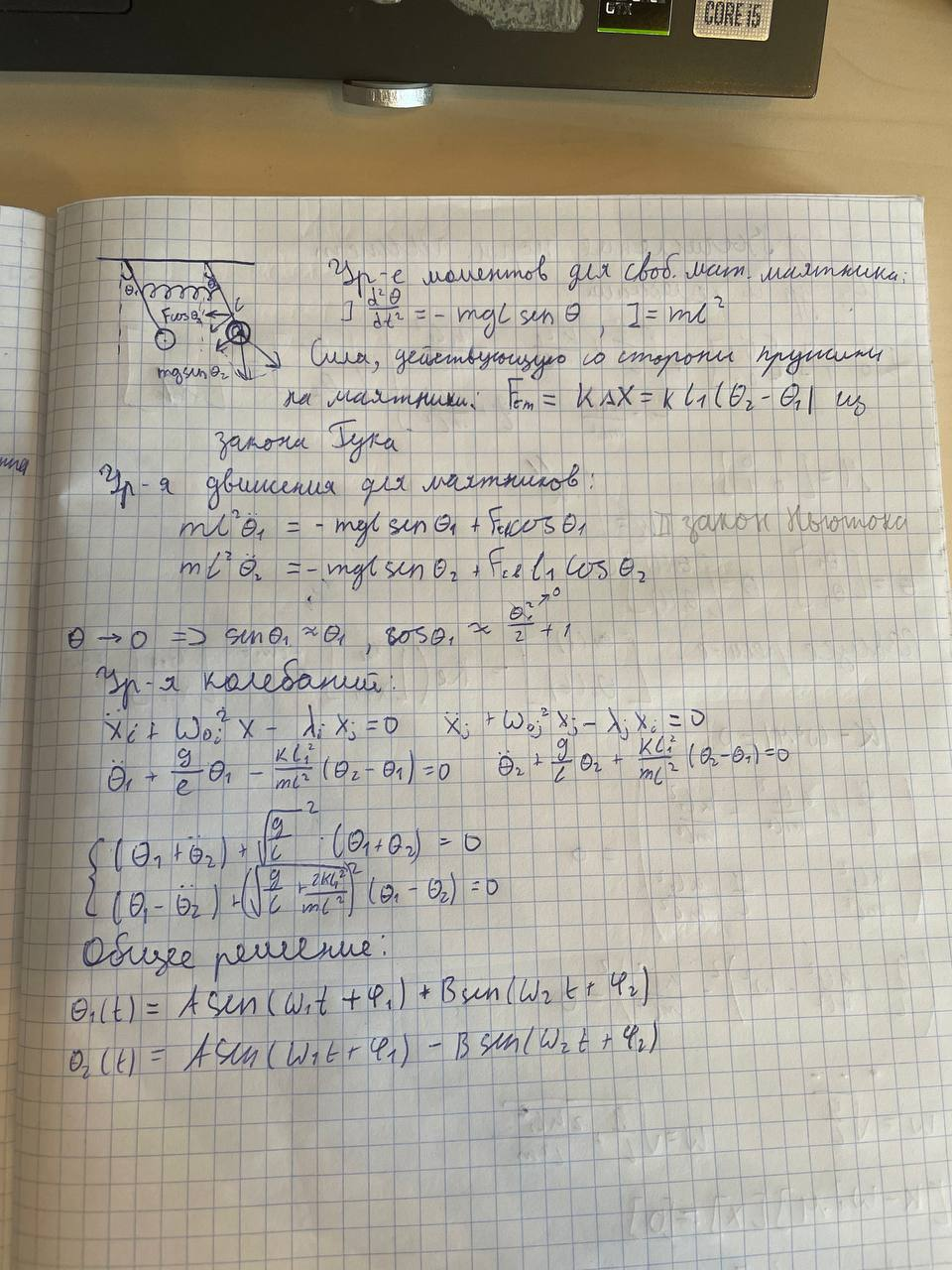
|  |  |
| --- | --- |
| Группа М3219 | К работе допущен |
| Студент Сливкин Артём | Работа выполнена |
| Преподаватель | Отчет принят |

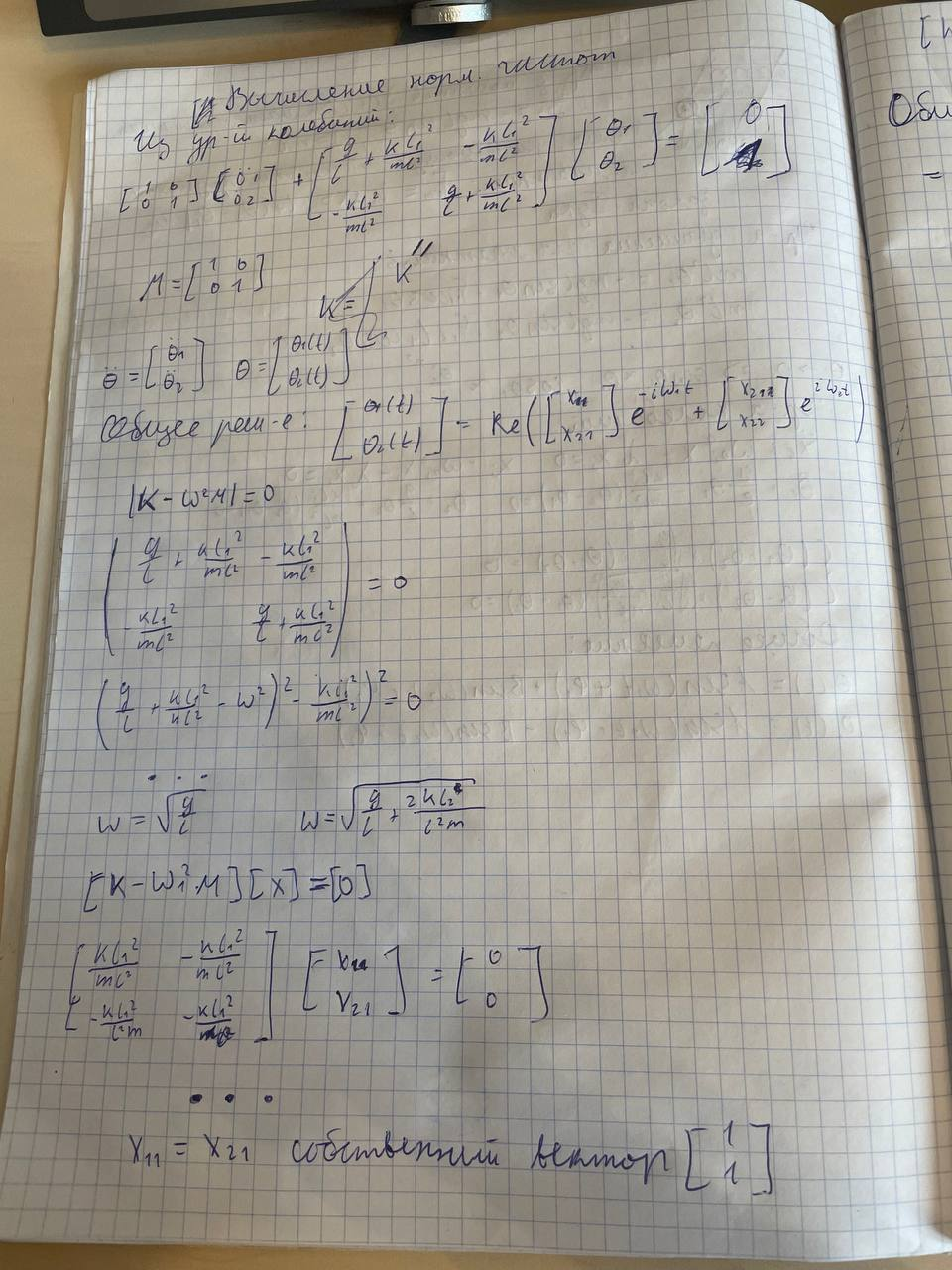
**Рабочий протокол и отчет по  
моделированию № 1**

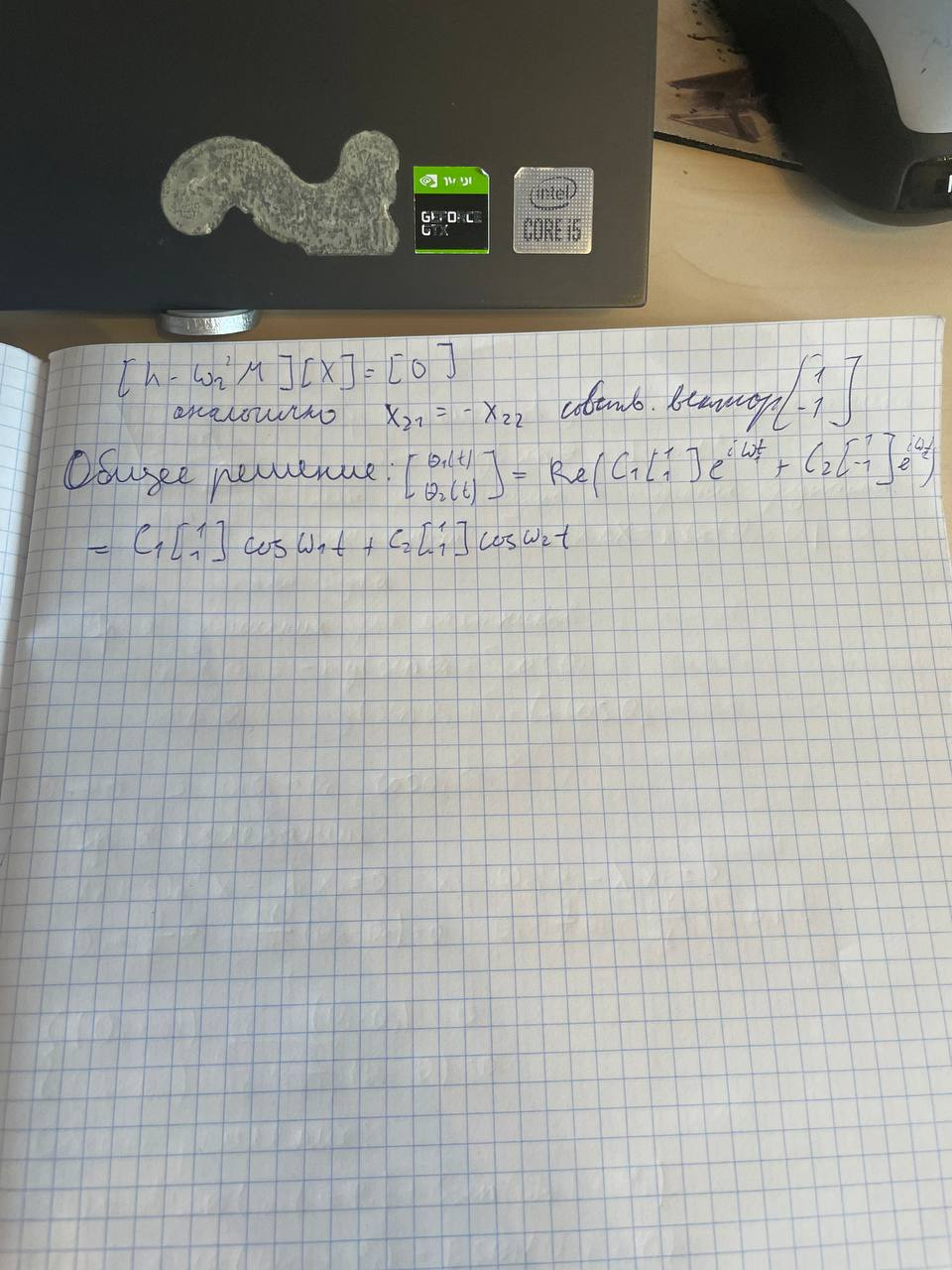
**Связанные маятники**

Два одинаковых математических маятника, связанных пружиной с коэффициентом жёсткости 𝑘 на расстоянии 𝐿1 от точки крепления маятников. Точки крепления обоих связанных маятников находятся на одном уровне. Оба математических маятника имеют одинаковые длины подвеса 𝐿 и массы 𝑚. Сила сопротивления для каждого маятника прямо пропорциональна скорости. Коэффициент затухания каждого маятника равен β. Для заданных начальных отклонений построить графики зависимостей углов и скоростей от времени для каждого маятника. Найти нормальные частоты. Параметры должны задаваться.



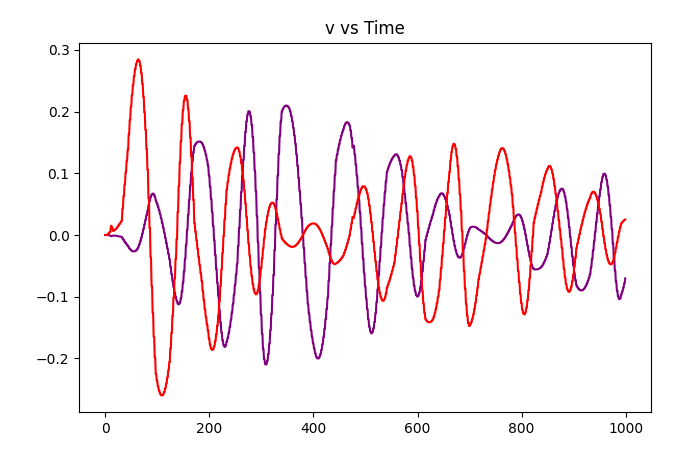
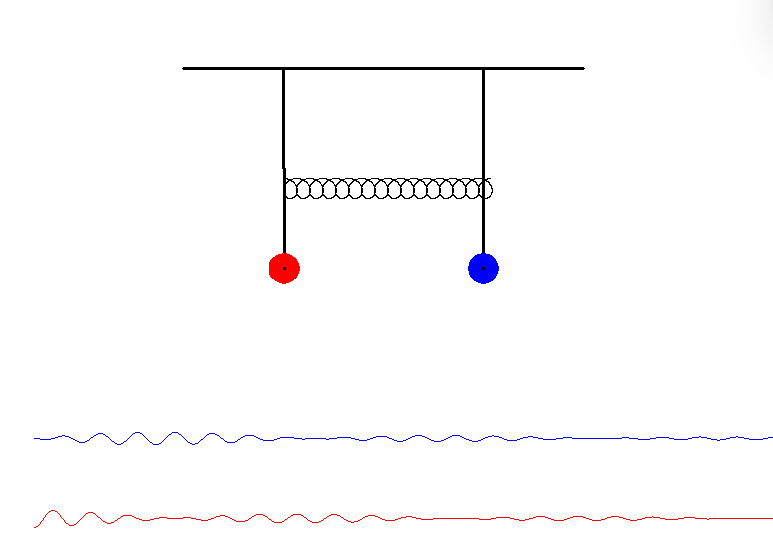
Вывод формул и решение дифференциальных уравнений:



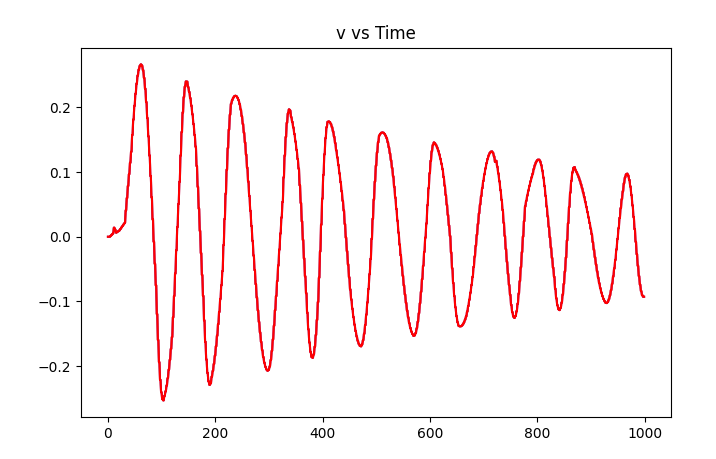
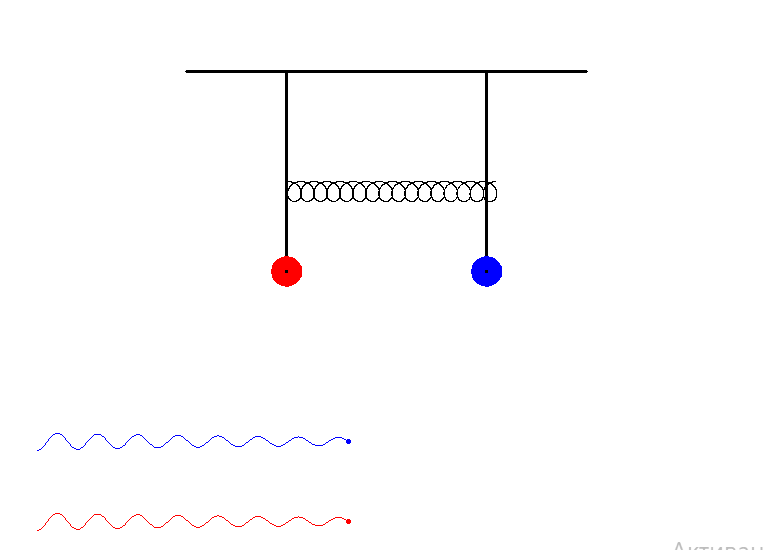


Рассмотрим нормальные частоты при различых ситуациях:

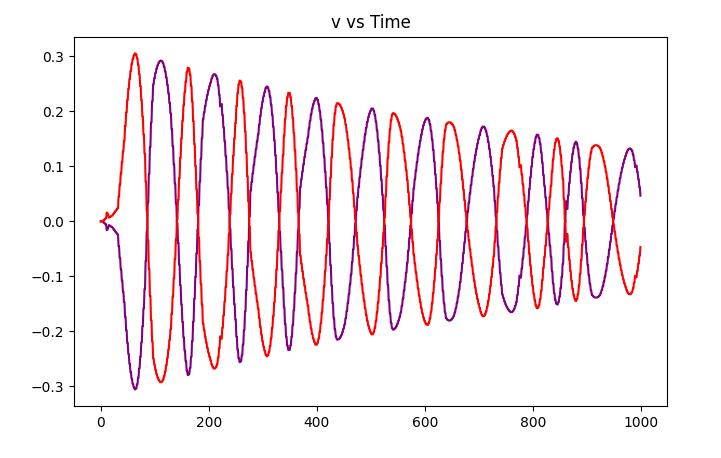
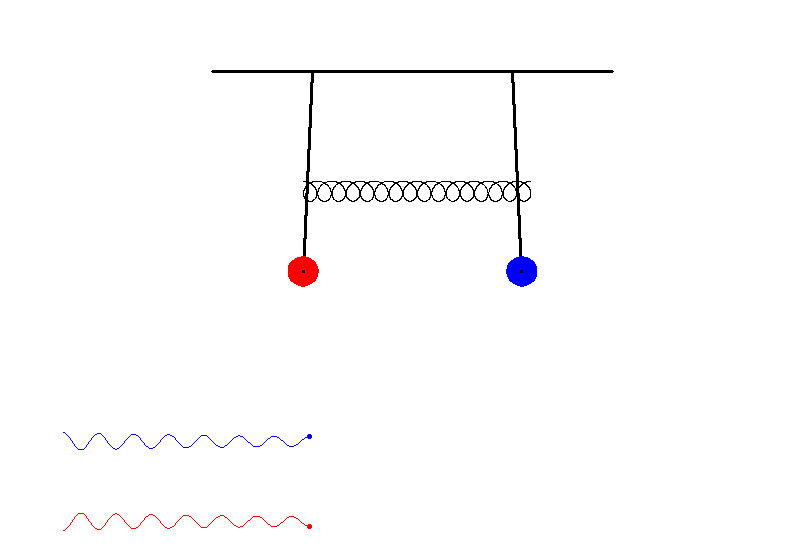
1. При отклонении только одного маятника на угол двух близких частот

1. При отклонении обоих маятников на одинаковый угол от положения равновесия в одну сторону – оба маятника синхронно (синфазно) колеблются с первой нормальной частотой

1. При отклонении обоих маятников на одинаковый угол от положения равновесия в разные стороны – маятники противофазно колеблются со второй нормальной частотой

import turtle  
from math import \*  
import scipy.integrate as integrate  
import numpy as np  
import time  
  
from matplotlib import pyplot as plt  
  
L = 200  
l1 = 100  
g = 9.8  
m = 0.5  
b = 0.05  
k = 3  
  
  
def spring\_extension(len, stretch):  
 *"""Рисует пружину."""* spring.clear()  
 for i in range(len + 1):  
 angle = 0  
 while angle <= 2 \* pi:  
 angle = angle + 0.01  
 x = 0.1 \* cos(angle)  
 y = 0.1 \* sin(angle)  
 spring.goto((spring.xcor() + x + stretch), spring.ycor() - y - height)  
 spring.goto(spring.xcor(), spring.ycor())  
  
  
def euler\_method(state, t):  
 *"""Вычисляет производные углов и угловых скоростей для системы двух маятников."""* theta1 = state[0]  
 omega1 = state[1]  
 theta2 = state[2]  
 omega2 = state[3]  
 dtheta1 = omega1  
 dtheta2 = omega2  
 v1.append(dtheta1)  
 v2.append(dtheta2)  
 domega1 = -(b / m) \* omega1 - g / (L / 200) \* sin(theta1) - ((k \* l1 \*\* 2) / (m \* L \*\* 2)) \* (  
 sin(theta1) - sin(theta2))  
 domega2 = -(b / m) \* omega2 - g / (L / 200) \* sin(theta2) + ((k \* l1 \*\* 2) / (m \* L \*\* 2)) \* (  
 sin(theta1) - sin(theta2))  
 dydx = [dtheta1, domega1, dtheta2, domega2]  
 return dydx  
  
  
win = turtle.Screen()  
win.setup(800, 600)  
win.bgcolor('white')  
win.tracer(0)  
  
pendulum1 = turtle.Turtle()  
pendulum1.shape('circle')  
pendulum1.color('blue')  
pendulum1.up()  
pendulum1.shapesize(1.5, 1.5)  
pendulum1.sign = 1  
  
pendulum2 = turtle.Turtle()  
pendulum2.shape('circle')  
pendulum2.color('red')  
pendulum2.up()  
pendulum2.shapesize(1.5, 1.5)  
pendulum2.sign = 0  
  
pendulum1.theta = 0  
pendulum1.omega = 0  
  
pendulum2.theta = -5  
pendulum2.omega = 0  
  
cell = turtle.Turtle()  
cell.ht()  
cell.up()  
cell.goto(-200, 200)  
cell.down()  
cell.pensize(3)  
cell.goto(200, 200)  
  
thread1 = turtle.Turtle()  
thread1.shape('square')  
thread1.shapesize(0.1, 0.1)  
thread1.pensize(3)  
thread1.up()  
  
thread2 = turtle.Turtle()  
thread2.shape('square')  
thread2.shapesize(0.1, 0.1)  
thread2.pensize(3)  
thread2.up()  
  
spring = turtle.Turtle()  
spring.shape('circle')  
spring.color('black')  
spring.up()  
spring.ht()  
  
th1 = turtle.Turtle()  
th1.shape('circle')  
th1.shapesize(0.2, 0.2)  
th1.up()  
th1.color('red')  
th1.goto(-350, -250)  
  
th2 = turtle.Turtle()  
th2.shape('circle')  
th2.shapesize(0.2, 0.2)  
th2.up()  
th2.color('blue')  
th2.goto(-350, -250)  
  
dt = 0.005  
t = np.arange(0.0, 200, dt)  
v1 = []  
v2 = []  
t\_vec = np.arange(0, 10, dt)  
  
values = np.radians([pendulum1.theta, pendulum1.omega, pendulum2.theta, pendulum2.omega])  
results = integrate.odeint(euler\_method, values, t)  
i = 1  
spring.up()  
spring.down()  
plt.plot(v1[0:1000], color='purple')  
plt.plot(v2[0:1000], color='red')  
plt.title("v vs Time")  
plt.show()  
  
while i < len(results):  
 x1 = 100 + L \* sin(results[i][0])  
 y1 = -L \* cos(results[i][0]) + L  
  
 x2 = -100 + L \* sin(results[i][2])  
 y2 = -L \* cos(results[i][2]) + L  
  
 pendulum1.goto(x1, y1)  
 thread1.goto(pendulum1.xcor(), pendulum1.ycor())  
  
 pendulum2.goto(x2, y2)  
 thread2.goto(pendulum2.xcor(), pendulum2.ycor())  
  
 th2.goto(th2.xcor() + 1, -220 + x1 / 2)  
 th1.goto(th1.xcor() + 1, -200 + x2 / 2)  
  
 length = x1 - x2  
 k = length / 9700  
 height = (y2 - y1) / 8500  
 spring\_extension(15, k)  
  
 win.update()  
  
 thread1.clear()  
 thread1.up()  
 thread1.goto(100, 200)  
 thread1.down()  
  
 thread2.clear()  
 thread2.up()  
 thread2.goto(-100, 200)  
 thread2.down()  
  
 spring.clear()  
 spring.up()  
 spring.goto(pendulum2.xcor(), pendulum2.ycor() + (l1 - 10))  
 spring.down()  
  
 if i == 1:  
 th1.down()  
 th2.down()  
  
 i += 10  
 time.sleep(0.01)

Вывод:

Связанные колебания демонстрируют возможность волновых движений. Волновое движение возможно лишь при наличии таких колеблющихся систем, которые взаимодействуют между собой и способны передавать друг другу свою энергию. Колебания маятника в общем случае представляют собой сумму двух гармонических колебаний с частотами ω1 и ω2. Если парциальные частоты маятников разные, то собственные частоты имеют вид , . Для нахождения функций *φ1(t)*и *φ2(t)* можно воспользоваться решением системы двух линейных дифференциальных уравнений.