Отчёт по лабораторной работе 4

дисциплина: Математическое моделирование

Василиса Михайловна Крючкова, НПИбд-02-18

Содержание

# Цель работы

Построить модель гармонических колебаний с помощью Python.

# Задание

**Вариант 41** Постройте фазовый портрет гармонического осциллятора и решение уравнения гармонического осциллятора для следующих случаев:

1. Колебания гармонического осциллятора без затуханий и без действий внешней силы
2. Колебания гармонического осциллятора c затуханием и без действий внешней силы
3. Колебания гармонического осциллятора c затуханием и под действием внешней силы

На интервале (шаг 0,05) с начальными условиями

# Выполнение лабораторной работы

**1. Колебания без затуханий и без действий внешней силы**

1.1. Уравнение свободных колебаний гармонического осциллятора имеет следующий вид:

Изучила начальные условия. Перед нами уравнение консервативного осциллятора, энергия колебания которого сохраняется во времени. Т. е. потери в системе отсутствуют, значит, . Собственная частота колебаний . . Правая часть уравнения .

1.2. Оформила начальные условия в код на Python:

x0 = np.array([1.0, 1.2])  
  
w1 = 3.5 #частота, уже в квадрате  
g1 = 0.0 #затухание, уже умноженное на 2  
  
def F1(t):  
 f = 0  
 return f

1.3. Решение ищем на интервале (шаг 0,05), значит, – начальный момент времени, – предельный момент времени, – шаг изменения времени.

1.4. Добавила в программу условия, описывающие время:

t0 = 0  
tmax = 37  
dt = 0.05  
t = np.arange(t0, tmax, dt)

1.5. Представила заданное уравнение второго порядка в виде системы двух уравнений первого порядка и запрограммировала:

def Y1(x, t):  
 dx1\_1 = x[1]  
 dx1\_2 = - w1\*x[0] - g1\*x[1] - F1(t)  
 return dx1\_1, dx1\_2

1.6. Запрограммировала решение системы уравнений:

x1 = odeint(Y1, x0, t)

1.7. Переписала отдельно в , а в :

y1\_1 = x1[:, 0]  
y1\_2 = x1[:, 1]

1.8. Описала построение фазового портрета:

plt.plot(y1\_1, y1\_2)

**2. Колебания c затуханием и без действий внешней силы**

2.1. Изучила начальные условия. Потери энергии в системе . Собственная частота колебаний . и те же, что и в п. 1.1. Правая часть уравнения такая же, как и в п. 1.1.

2.2. Т. к. вектор начальных условий одинаков для всех пунктов задачи, задаю его один раз в начале. Остальные начальные условия оформила в код на Python (функцию F1 переименовала в F12, т. к. она подходит как для 1-ого, так и для 2-ого случаев):

w2 = 3.0  
g2 = 7.0  
  
def F12(t):  
 f = 0  
 return f

2.3. Т. к. интервал, на котором ищем решение, одинаков для всех пунктов задачи, задаю его один раз в начале.

2.4. Представила заданное уравнение второго порядка в виде системы двух уравнений первого порядка и запрограммировала:

def Y2(x, t):  
 dx2\_1 = x[1]  
 dx2\_2 = - w2\*x[0] - g2\*x[1] - F12(t)  
 return dx2\_1, dx2\_2

2.5. Запрограммировала решение системы уравнений:

x2 = odeint(Y2, x0, t)

2.6. Переписала отдельно в , а в :

y2\_1 = x2[:, 0]  
y2\_2 = x2[:, 1]

2.7. Описала построение фазового портрета:

plt.plot(y2\_1, y2\_2)

**3. Колебания c затуханием и под действием внешней силы**

3.1. Изучила начальные условия. Потери энергии в системе . Собственная частота колебаний . и те же, что и в п. 1.1. Правая часть уравнения .

3.2. Т. к. вектор начальных условий одинаков для всех пунктов задачи, задаю его один раз в начале. Остальные начальные условия оформила в код на Python:

w3 = 2.0  
g3 = 5.0  
  
def F3(t):  
 f = 2\*np.sin(6\*t)  
 return f

3.3. Т. к. интервал, на котором ищем решение, одинаков для всех пунктов задачи, задаю его один раз в начале.

3.4. Представила заданное уравнение второго порядка в виде системы двух уравнений первого порядка и запрограммировала:

def Y3(x, t):  
 dx3\_1 = x[1]  
 dx3\_2 = - w3\*x[0] - g3\*x[1] - F3(t)  
 return dx3\_1, dx3\_2

3.5. Запрограммировала решение системы уравнений:

x3 = odeint(Y3, x0, t)

3.6. Переписала отдельно в , а в :

y3\_1 = x3[:, 0]  
y3\_2 = x3[:, 1]

3.7. Описала построение фазового портрета:

plt.plot(y3\_1, y3\_2)

**4. Сборка программы**

4.1. Собрала код программы воедино и получила следующее:

import math  
import numpy as np  
from scipy.integrate import odeint  
import matplotlib.pyplot as plt  
  
x0 = np.array([1.0, 1.2]) #вектор начальных условий  
  
w1 = 3.5 #частота, уже в квадрате  
g1 = 0.0 #затухание, уже умноженное на 2  
  
w2 = 3.0  
g2 = 7.0  
  
w3 = 2.0  
g3 = 5.0  
  
def F12(t):  
 f = 0  
 return f  
  
def F3(t):  
 f = 2\*np.sin(6\*t)  
 return f  
  
t0 = 0  
tmax = 37  
dt = 0.05  
t = np.arange(t0, tmax, dt)  
  
def Y1(x, t):  
 dx1\_1 = x[1]  
 dx1\_2 = - w1\*x[0] - g1\*x[1] - F12(t)  
 return dx1\_1, dx1\_2  
  
def Y2(x, t):  
 dx2\_1 = x[1]  
 dx2\_2 = - w2\*x[0] - g2\*x[1] - F12(t)  
 return dx2\_1, dx2\_2  
  
def Y3(x, t):  
 dx3\_1 = x[1]  
 dx3\_2 = - w3\*x[0] - g3\*x[1] - F3(t)  
 return dx3\_1, dx3\_2  
  
x1 = odeint(Y1, x0, t)  
x2 = odeint(Y2, x0, t)  
x3 = odeint(Y3, x0, t)  
  
y1\_1 = x1[:, 0]  
y1\_2 = x1[:, 1]  
  
y2\_1 = x2[:, 0]  
y2\_2 = x2[:, 1]  
  
y3\_1 = x3[:, 0]  
y3\_2 = x3[:, 1]  
  
plt.plot(y1\_1, y1\_2)  
plt.grid(axis = 'both')  
  
plt.plot(y2\_1, y2\_2)  
plt.grid(axis = 'both')  
  
plt.plot(y3\_1, y3\_2)  
plt.grid(axis = 'both')

4.2. Получила фазовые портреты гармонического осциллятора (см. рис. 1, 2 и 3):

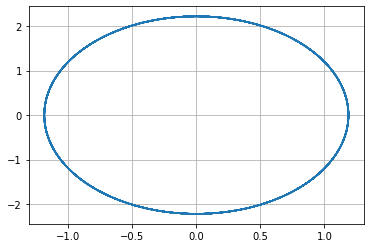


Figure 1: Колебания без затуханий и без действий внешней силы

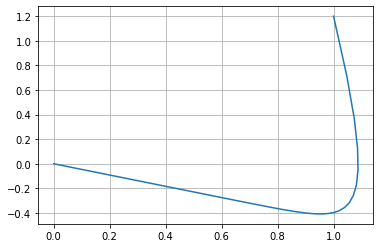


Figure 2: Колебания c затуханием и без действий внешней силы

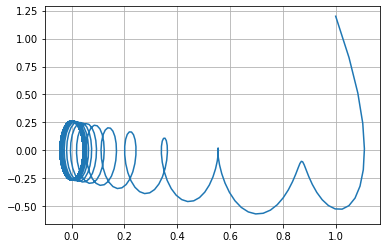


Figure 3: Колебания c затуханием и под действием внешней силы

# Выводы

Построила модель гармонических колебаний с помощью Python.

# Ответы на вопросы к лабораторной работе

*1. Запишите простейшую модель гармонических колебаний*

Простейшим видом колебательного процесса являются простые гармонические колебания, которые описываются уравнением .

*2. Дайте определение осциллятора*

Осциллятор — система, совершающая колебания, то есть показатели которой периодически повторяются во времени.

*3. Запишите модель математического маятника*

Уравнение динамики принимает вид:

В случае малых колебаний полагают . В результате возникает линейное дифференциальное уравнение

или

*4. Запишите алгоритм перехода от дифференциального уравнения второго порядка к двум дифференциальным уравнениям первого порядка*

Пусть у нас есть дифференциальное уравнение 2-го порядка:

Для перехода к системе уравнений первого порядка сделаем замену (это метод Ранге-Кутты):

Тогда получим систему уравнений:

*5. Что такое фазовый портрет и фазовая траектория?*

Фазовый портрет — это то, как величины, описывающие состояние системы (= динамические переменные), зависят друг от друга.

Фазовая траектория — кривая в фазовом пространстве, составленная из точек, представляющих состояние динамической системы в последовательные моменты времени в течение всего времени эволюции.