

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение
высшего образования**

**«Нижегородский государственный университет им. Н.И. Лобачевского»
Институт информационных технологий, математики и механики**

**ЛАБОРАТОРНАЯ РАБОТА
ПО ЧИСЛЕННЫМ МЕТОДАМ
«Численное решение задачи Коши для ОДУ»**

Выполнил:

студент группы 381706-4

Доброхотов Виталий

_____ Подпись

Проверил:

Доцент кафедры ДУМЧА

Морозов Кирилл Евгеньевич

_____ Подпись

**Нижний Новгород
2020**

Оглавление

Введение.....	3
Выбор модели.....	4
Выбор Метода.....	4
Программная реализация.....	5
Заключение.....	8
Литература.	9
Приложение.....	10

Введение.

Дифференциальное уравнение - уравнение, в которое входят производные функции, и может входить сама функция, независимая переменная и параметры. Порядок входящих в уравнение производных может быть различен. Производные, функции, независимые переменные и параметры могут входить в уравнение в различных комбинациях или могут отсутствовать вовсе, кроме хотя бы одной производной.

В отличие от алгебраических уравнений, в результате решения которых ищется число (несколько чисел), при решении дифференциальных уравнений ищется функция (семейство функций).

Дифференциальное уравнение порядка выше первого можно преобразовать в систему уравнений первого порядка, в которой число уравнений равно порядку исходного дифференциального уравнения.

Современные быстродействующие ЭВМ эффективно дают численное решение обыкновенных дифференциальных уравнений, не требуя получения его решения в аналитическом виде.

Выбор модели.

Для этой лабораторной работы я выбрал уравнение маятника с диссипацией:

$$x'' + kx' + \sin(x) = 0.$$

Выбор Метода.

Для данной лабораторной работы я решил использовать метод Рунге-Кутты 4-го порядка.

Для начала рассмотрим задачу Коши для ОДУ второго порядка:

$$x'' = F(t, x, x') \quad t_0, \quad x(t_0) = x_0, \quad x'(t_0) = x'_0$$

Данную задачу Коши можно свести к решению системы двух ДУ первого порядка, если ввести некоторую функцию $z = x'$, тогда $z' = x''$, и система примет вид

$$\begin{cases} x'' = F(t, x, z) \\ z' = G(t, x, z) \end{cases}$$

При начальных условиях $x(t_0) = x_0, z(t_0) = z_0$

Адаптируем формулы Рунге-Кутты 4-го порядка для данной системы уравнений:

$$k_1 = G(t_i, x_i, z_i)$$

$$l_1 = F(t_i, x_i, z_i)$$

$$k_2 = G(t_i + \frac{h}{2}, x_i + h\frac{k_1}{2}, z_i + h\frac{l_1}{2})$$

$$l_2 = F(t_i + \frac{h}{2}, x_i + h\frac{k_1}{2}, z_i + h\frac{l_1}{2})$$

$$k_3 = G(t_i + \frac{h}{2}, x_i + h\frac{k_2}{2}, z_i + h\frac{l_2}{2})$$

$$l_3 = F(t_i + \frac{h}{2}, x_i + h\frac{k_2}{2}, z_i + h\frac{l_2}{2})$$

$$k_4 = G(t_i + h, x_i + hk_3, z_i + hl_3)$$

$$l_4 = F(t_i + h, x_i + hk_3, z_i + hl_3)$$

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$z_{i+1} = z_i + \frac{h}{6}(l_1 + 2l_2 + 2l_3 + l_4)$$

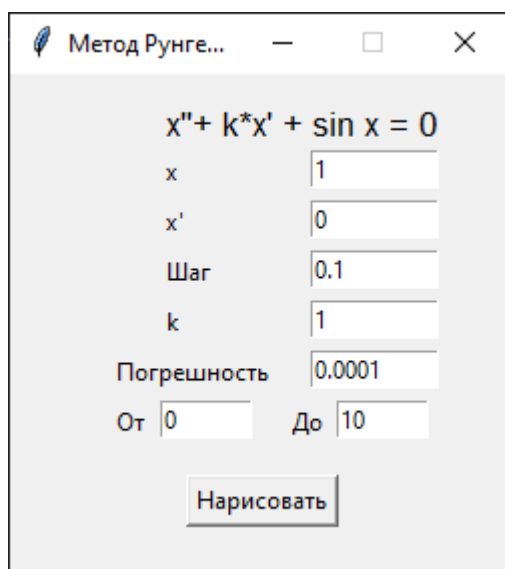
h – шаг

Приведенные формулы $k_1, k_2, k_3, k_4, l_1, l_2, l_3, l_4$, последовательно вычисляются на каждом шаге, после чего вычисляются y_{i+1} и z_{i+1} .

Программная реализация.

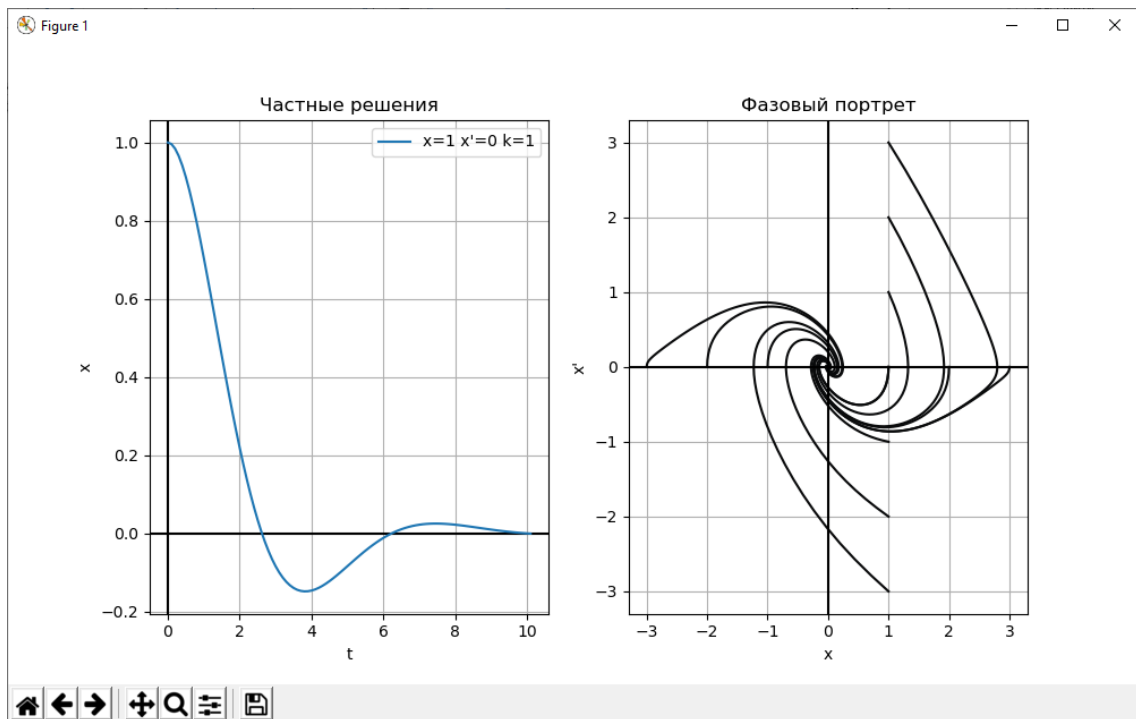
Программа для решения дифференциального уравнения реализована на языке Python 3 с использованием библиотек Decimal - для более точных расчетов, так как метод Рунге-Кутты даёт практически точное решение; tkinter - для создания GUI, matplotlib.pyplot – для рисования графиков.

После запуска программы откроется данное окно, в котором потребуется ввести некоторые параметры:

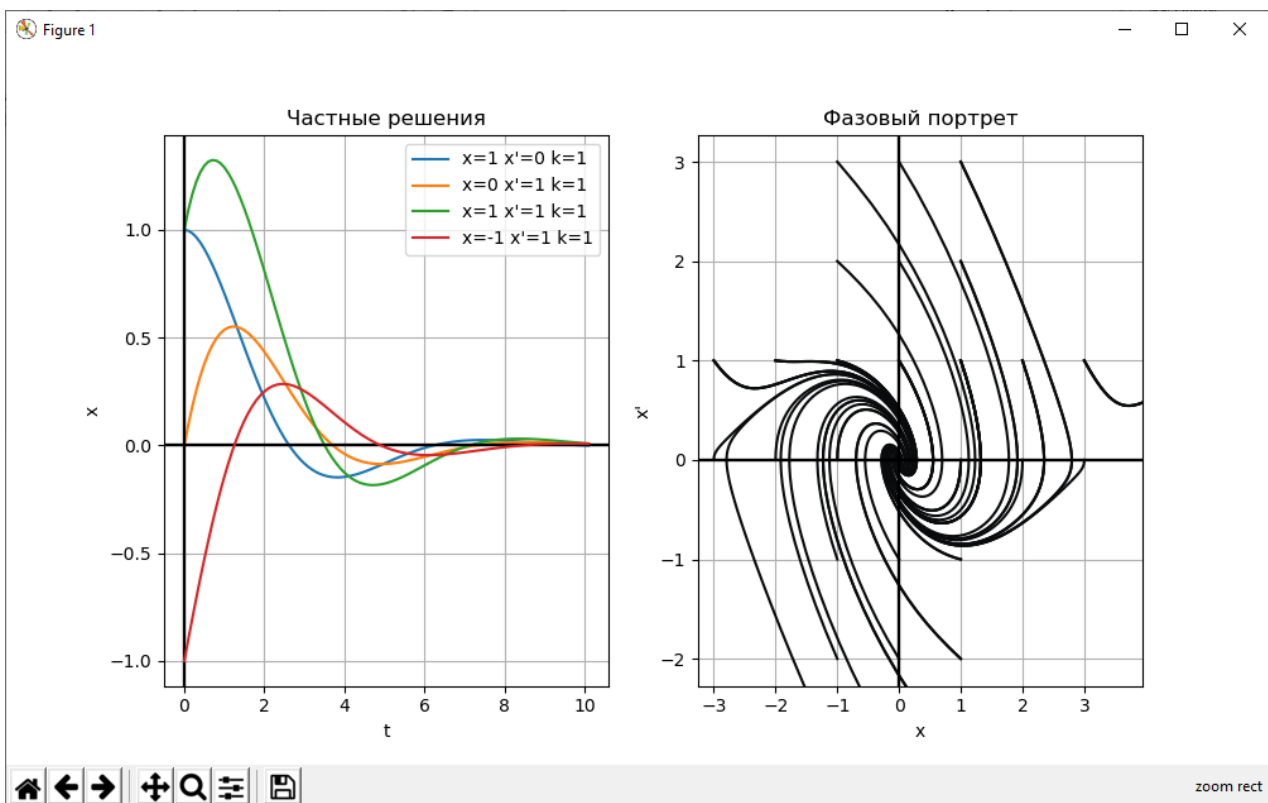


The screenshot shows a window titled "Метод Рунге...". Inside, the differential equation $x'' + kx' + \sin x = 0$ is displayed. Below the equation are several input fields: "x" with value 1, "x'" with value 0, "Шаг" (Step) with value 0.1, "k" with value 1, "Погрешность" (Error) with value 0.0001, and a range "От 0" to "До 10". At the bottom is a button labeled "Нарисовать" (Draw).

Для удобства все поля заполнены по умолчанию, но по желанию их можно поменять. Хочется уточнить, не целые числа вводятся с использованием точки, а не запятой. Также не рекомендуется вводить погрешность выше указанного дефолтного значения, в связи с тем, что метод Рунге-Кутты является довольно точным. Интервал должен быть указан строго в целых числах. По заполнению всех полей пользователю должен нажать на кнопку «Нарисовать», после чего откроется новое окно с двумя графиками:



На первом графике будут отображаться все частные решения, введенные пользователем, на втором - частично фазовый портрет уравнения с начальными условиями в некоторой точке. Если понадобится нарисовать еще один рисунок, пользователь может перейти в первое окно, где нужно заполнить некоторые поля и нажать еще раз кнопку «Нарисовать», после чего графики обновятся.



Настоятельно не рекомендуется вводить несколько графиков с огромной разницей в параметрах, так как из-за этого может испортиться фазовый портрет, потому что он рисуется только для определенной области. Для более детального просмотра графиков для пользователя слева-снизу имеются несколько кнопочек. С их помощью можно отдалить или приблизить какую-либо часть графика (значок лупы), перемещаться по нему (крестик), и если вдруг заблудится вернуться в исходное положение (домик).

Если пользователь хочет очистить графики и рисовать на новом полотне, предлагается закрыть окно с графиками, заново указать нужные параметры в меню и нажать кнопку «Нарисовать».

Заключение.

В ходе работы я ознакомился с методом Рунге-Кутты и реализовал программу, которая решает дифференциальное уравнение второго порядка.

В ходе работы я узнал, что метод Рунге-Кутты, не смотря на огромное количество формул, абсолютно не сложен в реализации. Более того этот метод имеет четвёртый порядок точности. Это значит, что ошибка на одном шаге имеет порядок $O(h^5)$, а суммарная ошибка на конечном интервале интегрирования имеет порядок $O(h^4)$. А из этого можно сделать небольшой вывод, что данный метод отличается от метода Эйлера точностью. Если требуется быстро рассчитать использовать для данной задачи метод Эйлера, если требуется более точный результат использовать метод Рунге-Кутты.

Литература.

1. А.А.Самарский. Введение в численные методы М.: Наука, 1982.
2. <http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.html>

Приложение.

```
def RungeKutta(begin, end, first, second, error, step, sigma):
    count = int((end - begin) / step)
    x = [begin]
    y = [first]
    z = [second]
    for i in range(0, count+1):
        k1 = g(x[i], y[i], z[i])
        l1 = f(x[i], y[i], z[i], sigma)
        k2 = g(x[i] + step/2, y[i] + (step * k1/2), z[i] + (step * l1/2))
        l2 = f(x[i] + step/2, y[i] + (step * k1/2), z[i] + (step * l1/2), sigma)
        k3 = g(x[i] + step/2, y[i] + (step * k2/2), z[i] + (step * l2/2))
        l3 = f(x[i] + step/2, y[i] + (step * k2/2), z[i] + (step * l2/2), sigma)
        k4 = g(x[i] + step, y[i] + (step * k3), z[i] + (step * l3))
        l4 = f(x[i] + step, y[i] + (step * k3), z[i] + (step * l3), sigma)
        tmp_y = y[i] + ((k1 + 2 * k2 + 2 * k3 + k4) * step) / 6 + error
        tmp_z = z[i] + ((l1 + 2 * l2 + 2 * l3 + l4) * step) / 6
        x.append(x[i]+step)
        y.append(tmp_y)
        z.append(tmp_z)
    return x, y, z
```