МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Нижегородский государственный университет
им. Н.И. Лобачевского»
Национальный исследовательский университет**

**Институт информационных технологий, математики и механики
Кафедра алгебры, геометрии и дискретной математики**

ЛАБОРАТОРНАЯ РАБОТА
«Численное решение задачи Коши для ОДУ 2 порядка»

Выполнил: студент группы 381706-02
Окмянский Андрей Владимирович

_____ Подпись

Руководитель:
Эгамов Альберт Исмаилович

_____ Подпись

Нижний Новгород

2020

Оглавление

Введение

Обыкновенные дифференциальные уравнения (ОДУ) широко используются для математического моделирования процессов и явлений в различных областях науки и техники. Переходные процессы в радиотехнике, кинетика химических реакций, динамика биологических популяций, движение космических объектов, модели экономического развития исследуются с помощью ОДУ.

Первоначально дифференциальные уравнения возникли из задач механики, в которых требовалось определить координаты тел, их скорости и ускорения, рассматриваемые как функции времени при различных воздействиях. К дифференциальным уравнениям приводили также некоторые рассмотренные в то время геометрические задачи.

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 (формально он ничем не ограничен). Производные, функции, независимые переменные и параметры могут входить в уравнение в различных комбинациях или могут отсутствовать вовсе, кроме хотя бы одной производной. Не любое уравнение, содержащее производные неизвестной функции, является дифференциальным уравнением. Например, $\boldsymbol{f'(x) = f(f(x))}$ не является дифференциальным уравнением.

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. Это позволило некоторым исследователям утверждать, что решение задачи получено, если её удалось свести к решению обыкновенного дифференциального уравнения.

Наиболее часто встречаются дифференциальные уравнения вида:

$$\boldsymbol{y^{(n)}=f(x,y,y',y'',\ldots,y^{(n-1)})},$$

в которых старшая производная $\boldsymbol{y^{(n)}}$ выражается в виде функции от переменных x,y и производных $\boldsymbol{y^{(i)}}$ порядков меньше $\boldsymbol{n}$. Такие дифференциальные уравнения называются нормальными или разрешенными относительно производной.

Порядок дифференциального уравнения — наивысший порядок производных, входящих в него.

Если дифференциальное уравнение является многочленом относительно старшей производной, то степень этого многочлена называется степенью дифференциального уравнения.

Решением (интегралом) дифференциального уравнения порядка n называется функция ***y(x)***, имеющая на некотором интервале (a, b)производные ***y'(x), y''(x), …, $y^{(n)}(x)$*** до порядка n включительно и удовлетворяющая этому уравнению.

Начальным условием для написанного выше уравнения называется условие

$$\boldsymbol{y(x_0)=y_0},\ \boldsymbol{y'(x_0)=y_0^{(1)}},\ \boldsymbol{y''(x_0)=y_0^{(2)}},\ \ldots,\ \boldsymbol{y^{(n-1)}(x_0)=y_0^{(n-1)}},$$

где $\boldsymbol{x_0}$ – некоторое фиксированное значение независимой переменной, а $\boldsymbol{y_0}$ и $\boldsymbol{y_0^{(i)}}$ – соответственно, фиксированные значения функции y и всех её производных до порядка $\boldsymbol{n-1}$ включительно.

Дифференциальное уравнение вместе с начальным условием называется начальной задачей или задачей Коши:

$$\{\boldsymbol{y^{(n)}=f(x,y,y',y'',\ldots,y^{(n-1)})}\ \boldsymbol{y(x_0)=y_0},\ \boldsymbol{y'(x_0)=y_0^{(1)}},\ \boldsymbol{y''(x_0)=y_0^{(2)}},\ldots,\boldsymbol{y^{(n-1)}(x_0)=y_0^{(n-1)}}.$$

В работе будет рассмотрен наиболее распространённый метод Рунге-Кутты 4-го порядка при вычислениях с постоянным шагом интегрирования.

Метод Рунге-Кутты 4-го порядка

Ме́тоды Ру́нге-Ку́тты — большой класс численных методов решения задачи Коши для обыкновенных дифференциальных уравнений и их систем. Первые методы данного класса были предложены около 1900 года немецкими математиками К. Рунге и М. В. Куттой.

К классу методов Рунге-Кутты относятся явный метод Эйлера и модифицированный метод Эйлера с пересчётом, которые представляют собой соответственно методы первого и второго порядка точности. Существуют стандартные явные методы третьего порядка точности, не получившие широкого распространения. Наиболее часто используется и реализован в различных математических пакетах (Maple, MathCAD, Maxima) классический метод Рунге-Кутты, имеющий четвёртый порядок точности. При выполнении расчётов с повышенной точностью всё чаще применяются методы пятого и шестого порядков точности. Построение схем более высокого порядка сопряжено с большими вычислительными трудностями.

Методы седьмого порядка должны иметь по меньшей мере девять стадий, а методы восьмого порядка — не менее 11 стадий. Для методов девятого и более высоких порядков (не имеющих, впрочем, большой практической значимости) неизвестно, сколько стадий необходимо для достижения соответствующего порядка точности.

Желание повысить вычислительную эффективность привело к появлению различных вычислительных версий методов Рунге-Кутты. В таких версиях стремились получить формулы из семейства методов Рунге-Кутты, которые бы использовали одни и те же значения функции – правой части уравнения – и определяли бы разные конкретные методы одного порядка (или смежных порядков, например, четвертого и пятого); при этом, чтобы по разности результатов подсчета приближенных значений решения по выведенным близким формулам (с одним и тем же шагом $\boldsymbol{h}$) можно было судить о точности одного из них.

Итак, в работе был рассмотрен метод Рунге-Кутты 4-го порядка, который является самым распространенным среди семейства методов, что его часто называют просто методом Рунге-Кутты.

Методы Рунге-Кутта обладают следующими свойствами:

1. Эти методы являются одноступенчатыми: чтобы найти $\boldsymbol{y_{m+1}}$, нужна информация о предыдущей точке $\boldsymbol{x_m}$, $\boldsymbol{y_m}$.

2. Они согласуются с рядом Тейлора вплоть до членов порядка $\boldsymbol{h_p}$, где степень $\boldsymbol{p}$ различна для различных методов и называется порядковым номером или порядком метода.

3. Они не требуют вычисления производных от f (x,y), а требуют вычисления самой функции. Метод Рунге-Кутты используют для расчета стандартных моделей достаточно часто, так как при небольшом объеме вычислений он обладает точностью метода $O^4(h)$.

Рассмотрим Задачу Коши для системы обыкновенных дифференциальных уравнений первого порядка. $\boldsymbol{y'=f(x,y), y(x_0)=y_0}$.

Тогда приближенное значение в последующих точках вычисляется по формуле:

$$y_{i+1}=y_i+16(k_1+2k_2+2k_3+k_4),$$

$$k_1=hf(x_i,y_i),$$

$$k_2=hf(x_i+h/2,yi+k_1/2),$$

$$k_3=hf(x_i+h/2,yi+k_2/2),$$

$$k_4=hf(x_i+h,y_i+k_3),$$

$$h=(x_k-x_1)/n.$$

Где $i=\{1, 2, \ldots, k\}$, а n есть количество участков, на которые будет разбит фазовый портрет для подсчета коэффициентов.

Погрешность формулы, как и всех формул Рунге-Кутта четвертого порядка точности, представляется следующим образом:

$$\rho_1=(h_5/120)*\varphi_4^{(5)}(0)+o(h^5),$$

где

$$\varphi_4(h)=y(x_0+h)-y_0-p_{41}k_1(h)-p_{42}k_2(h)-p_{43}k_3(h)-p_{44}k_4(h).$$

Расчет на i-ом шаге методом Рунге-Кутты (рис.1).

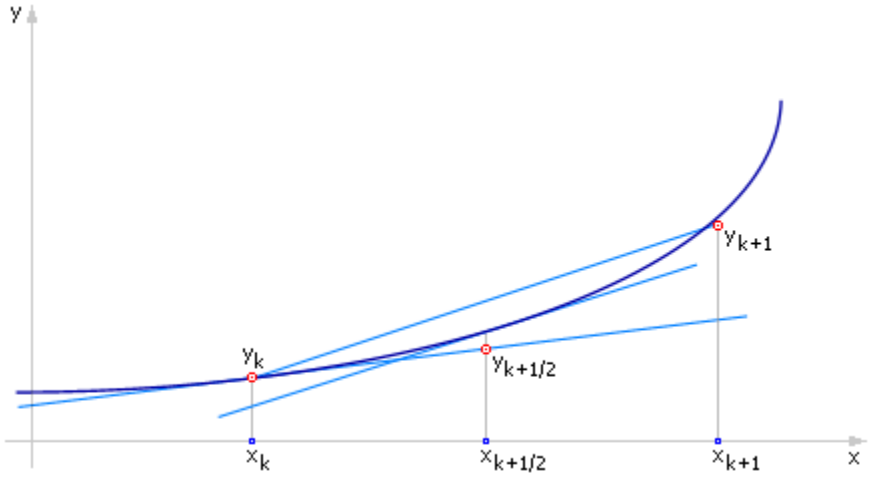

рис. 1

Структура программы

Программа принимает на вход параметры уравнения x"+δx'sin(nx)+cos(nx)=0 δ, n, шаг h, x’(0), начало и конец графика.

Программа состоит из следующих модулей:

1. RungeKutta.cs – описание класса RungeKutta. В данном классе реализован метод Рунге-Кутта, позволяющий решать задачу Коши.
2. Form1.cs – реализация формы для графического приложения. В данном классе реализована отрисовка фазового портрета.

Руководство пользователя

Работа с программой осуществляется следующим образом:

1. При запуске пользователю будет предложено ввести параметры дифференциального уравнения x"+δx'sin(nx)+cos(nx)=0, задать отрезок интегрирования и шаг. Изначально, при открытии программы, начальные параметры уравнения уже присутствуют на своих местах (рис. 2).

рис. 2

2. При нажатии на кнопку “Построить”, на появившейся координатной плоскости будет изображена получившееся фазовая траектория (рис. 3).

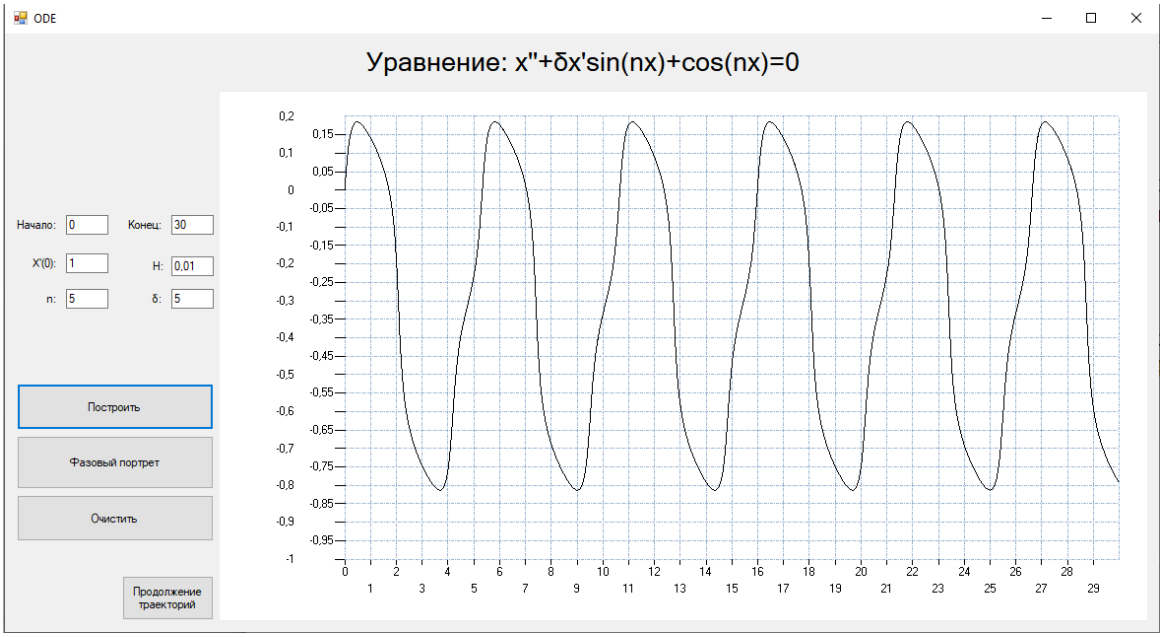

рис. 3

3. Для того, чтобы продолжить фазовый портрет или фазовую траекторию, внизу экрана расположена соответствующая кнопка “Продолжение траекторий” (рис. 4).

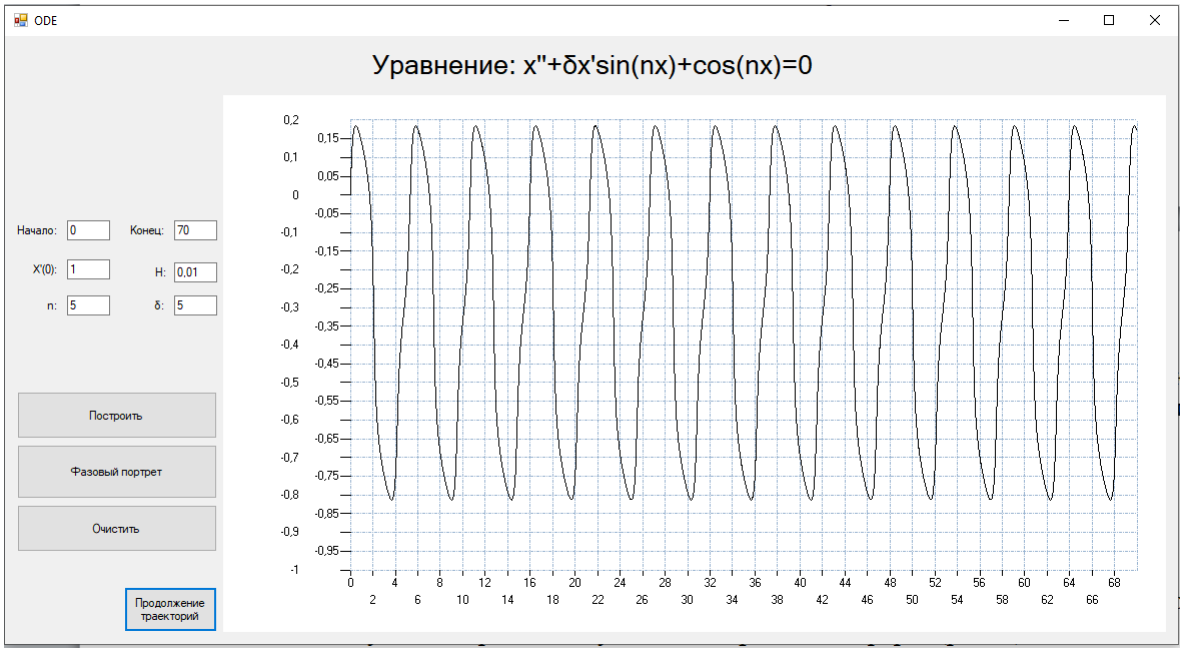

рис. 4

4. При нажатии кнопки “Фазовый портрет”, на появившейся координатной плоскости будет изображен получившийся фазовый портрет (рис. 5).

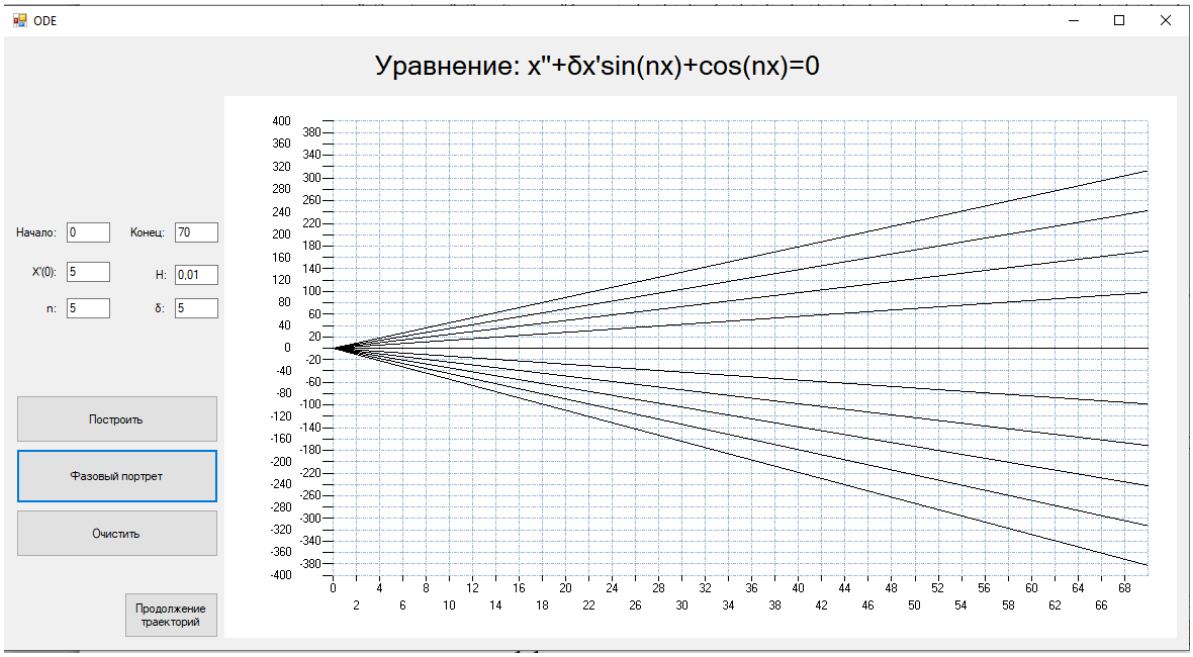

рис. 5

5. Чтобы очистить координатную плоскость, необходимо нажать на кнопку “Очистить”. После этого можно заново строить Фазовый портрет или фазовую плоскость дифференциального уравнения.

Руководство программиста

Программа написана в Visual Studio 2017 на языке C#.

Ниже представлен листинг основных функций программы:

```
namespace ODE {

    class RungeKutta {

        // вспомогательная функция для метода Рунге-Кутта
        static double func(double dv, double x, double lambda, double n) {
            return -1 * (lambda * dv * Math.Sin(2 * n * x) + Math.Cos(n * x));
        }

        // метод Рунге-Кутта 4-го порядка
        public static void RungKUtt(double begin, double end, double h, double lambda,
            double x0dash, double n, List<double> res, List<double> res_v) {

            double _dx = begin, _dv = x0dash;
            for (double i = begin; i < end; i += h) {
                res.Add(_dx);
                res_v.Add(_dv);

                double dx1 = h * _dv;
                double dv1 = h * func(_dv, _dx, lambda, n);
                double dx2 = h * (_dv + dv1 / 2);
                double dv2 = h * func(_dv + dv1 / 2, _dx + dx1 / 2, lambda, n);
                double dx3 = h * (_dv + dv2 / 2);
                double dv3 = h * func(_dv + dv2 / 2, _dx + dx2 / 2, lambda, n);
                double dx4 = h * (_dv + dv3);
                double dv4 = h * func(_dv + dv3, _dx + dx3, lambda, n);
                double dx = (dx1 + 2 * dx2 + 2 * dx3 + dx4) / 6;
                double dv = (dv1 + 2 * dv2 + 2 * dv3 + dv4) / 6;

                _dx += dx;
                _dv += dv;
            }
            res.Add(_dv);
        }
    }
}
```

Заключение

В процессе работы был изучен метод Рунге-Кутта, построение фазовых траекторий. Были решены и реализованы все поставленные задачи и цели, а именно:

Были созданы программные средства, поддерживающие ввод параметров заданного дифференциального уравнения, а так же отрисовка фазовых траекторий и фазового портрета.

Список используемой литературы

1. Метод Рунге-Кутта 4-го порядка: http://stratum.ac.ru/education/textbooks/modelir/lection15.html
2. Самарский А.А. “Введение в численный методы”, Издательство «Лань», 2005. – 228с.