ГОУ ВПО «Донецкий национальный университет»

Физико-технический факультет

Кафедра компьютерных технологий

**ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА**

**ЛАБОРАТОРНАЯ РАБОТА №5**

**«ЧИСЛЕННОЕ РЕШЕНИЕ ЗАДАЧИ КОШИ»**

Выполнил:

Ермоленко Евгений

Студент 2 уск.курса

группы ИВТ-4

Проверил:

Пшеничный К.А.

**Ход работы**

Решить задачу Коши методом Эйлера и Рунге-Кутта 4-го порядка на трёх сетках при *N* =10, 50, 500.

, .

Построить графики решений для обоих методов на всех сетках.

Значения по варианту:

, , ,

Для задачи Коши , схема **Эйлера** принимает такой вид:

**,**

Или

Погрешность на шаге или локальная погрешность — это разность между численным решением после одного шага вычисления  и точным решением в точке .

Погрешность в целом, глобальная или накопленная погрешность — это погрешность в последней точке произвольного конечного отрезка интегрирования уравнения. Для вычисления решения в этой точке требуется  шагов, где S - длина отрезка. Поэтому глобальная погрешность метода .

Таким образом, метод Эйлера является методом первого порядка — имеет погрешность на шаге  и погрешность в целом .

**Метод Рунге — Кутты** четвёртого порядка при вычислениях с постоянным шагом интегрирования столь широко распространён, что его часто называют просто методом Рунге — Кутты.

Для задачи Коши приближенное значение в точках  вычисляется по итерационной формуле:

Вычисление нового значения проходит в четыре стадии:

где  - величина шага сетки по .

Этот метод имеет четвёртый порядок точности. Это значит, что ошибка на одном шаге имеет порядок , а суммарная ошибка на конечном интервале интегрирования имеет порядок .

Реализация данного алгоритма средствами языка C# представлена ниже.

**Функция нахождения значения вычисляемой функции f(x,v):**

static public double f(double x, double v)

{

return x / v;

}

**Функция для нахождения значения f(x,v) методом Эйлера:**

static public void methodEuler(int N, double x0, double xN, double v0, Chart chart, int chartSeries)

{

double h = (xN - x0) / N;

List<double> X = new List<double>();

List<double> func = new List<double>();

List<double> methodEuler = new List<double>();

X.Add(x0);

func.Add(f(x0, v0));

methodEuler.Add(v0);

int i = 1;

for (double x = x0; x <= xN; x += h)

{

methodEuler.Add(methodEuler[i - 1] + h \* func[i - 1]);

func.Add(f(X[i - 1] + h, methodEuler[i]));

X.Add(x + h);

i++;

}

for (i = 0; i < methodEuler.Count; i++)

{

chart.Series[chartSeries].Points.AddXY(X[i], methodEuler[i]);

//Console.WriteLine("i=" + i + "\tx= " + X[i] + "\tf(x,v)= " + func[i] + "\t Эйлер= " + methodEuler[i]);

}

X.Clear();

func.Clear();

methodEuler.Clear();

}

**Функция для нахождения значения f(x,v) методом Рунге-Кутта:**

static public void methodRungeKutta(int N, double x0, double xN, double v0, Chart chart, int chartSeries)

{

double h = (xN - x0) / N;

List<double> X = new List<double>();

List<double> k1 = new List<double>();

List<double> k2 = new List<double>();

List<double> k3 = new List<double>();

List<double> k4 = new List<double>();

List<double> methodRK = new List<double>();

X.Add(x0);

methodRK.Add(v0);

int i = 1;

for (double x = x0; x <= xN; x += h)

{

//Console.WriteLine(X[i-1]);

k1.Add(f(X[i - 1], methodRK[i - 1]));

//Console.WriteLine(k1);

k2.Add(f(X[i - 1] + h / 2, methodRK[i - 1] + (h \* k1[i - 1]) / 2));

//Console.WriteLine(k2[i-1]);

k3.Add(f(X[i - 1] + h / 2, methodRK[i - 1] + (h \* k2[i - 1]) / 2));

//Console.WriteLine(k3[i-1]);

k4.Add(f(X[i - 1] + h, methodRK[i - 1] + h \* k3[i - 1]));

//Console.WriteLine(k4[i-1]);

methodRK.Add(methodRK[i - 1] + h / 6 \* (k1[i - 1] + 2 \* k2[i - 1] + 2 \* k3[i - 1] + k4[i - 1]));

//Console.WriteLine(methodRK[i]);

X.Add(x + h);

i++;

}

for (i = 0; i < methodRK.Count; i++)

{

chart.Series[chartSeries].Points.AddXY(X[i], methodRK[i]);

//Console.WriteLine("i=" + i + "\tx= " + X[i] + "\t Рунге-Кутта= " + methodRK[i]);

}

X.Clear();

}

**Результат работы приложения:**

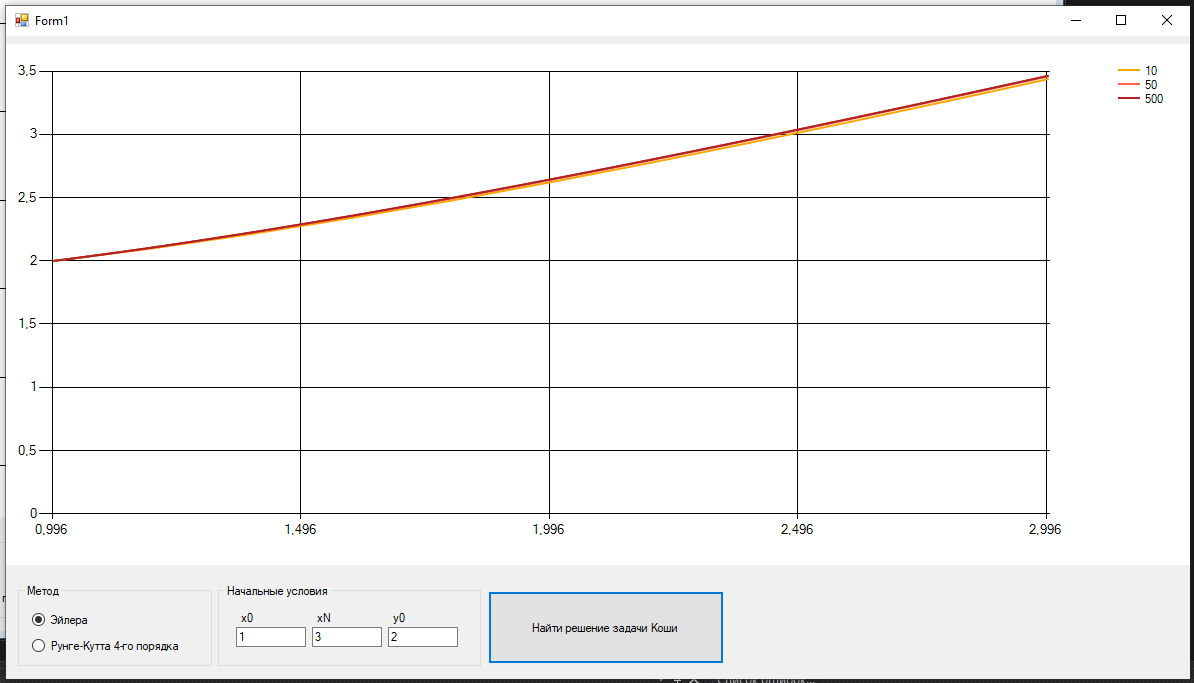


Рисунок 1 – Результат работы программы

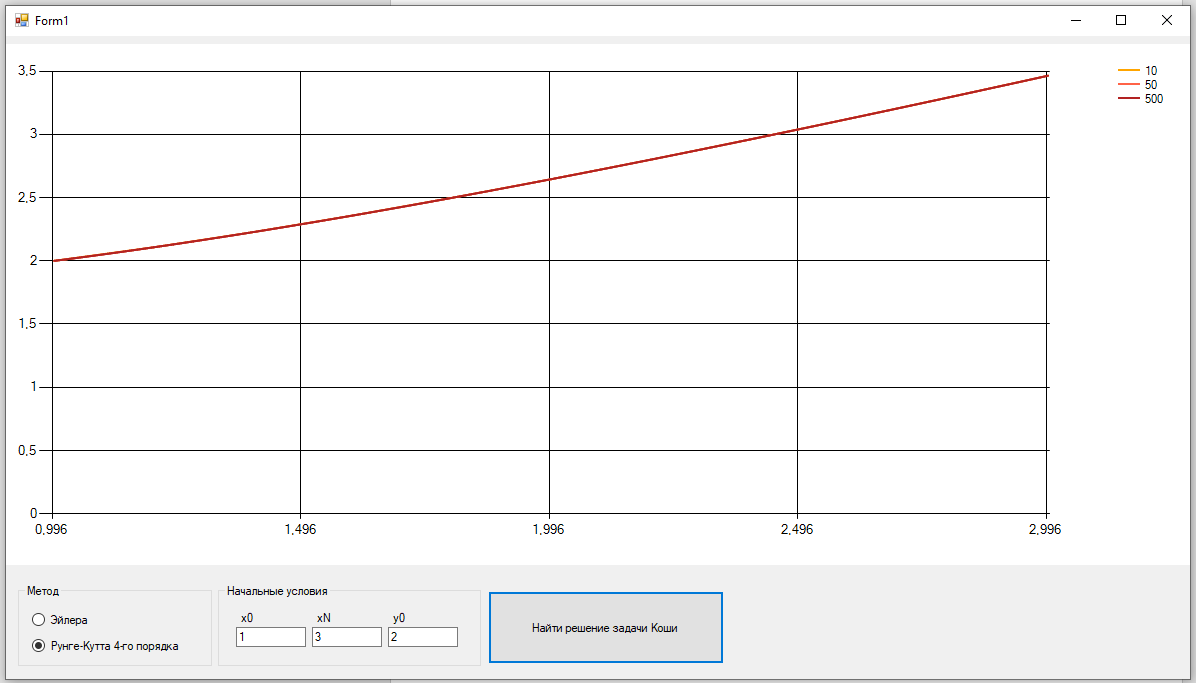
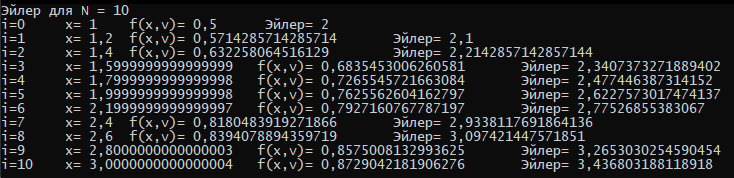
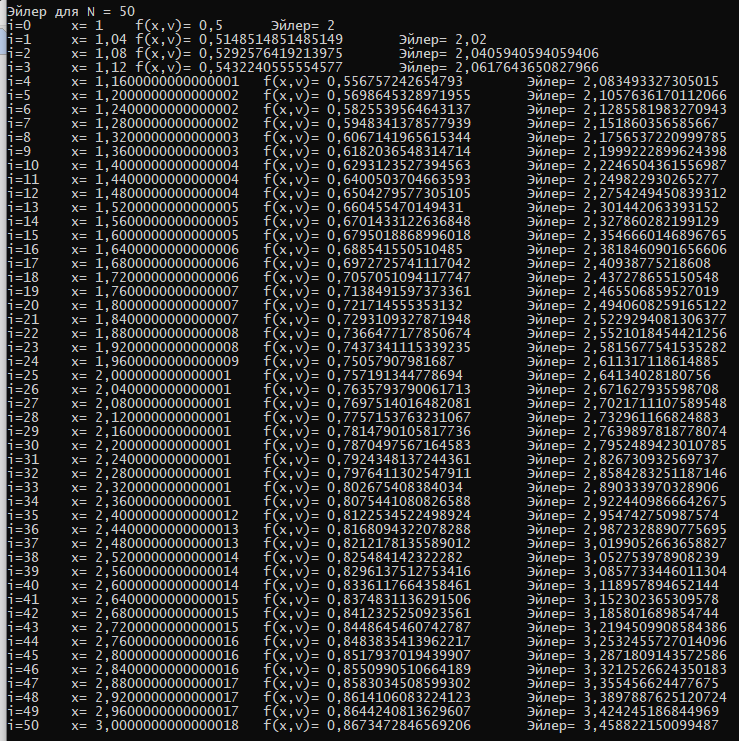


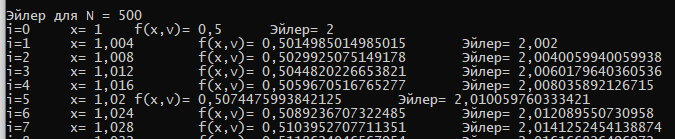
Рисунок 2 – Результат работы программы

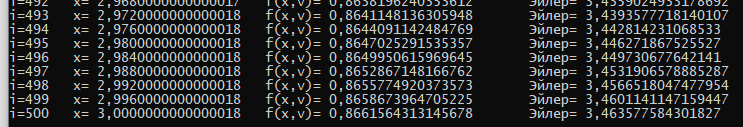
Рассчитанные значения на всех сетках и точные значения искомой функции в избранных узлах для каждого продемонстрированы ниже.

**Метод Эйлера для всех N:**

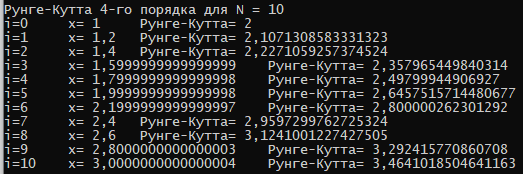


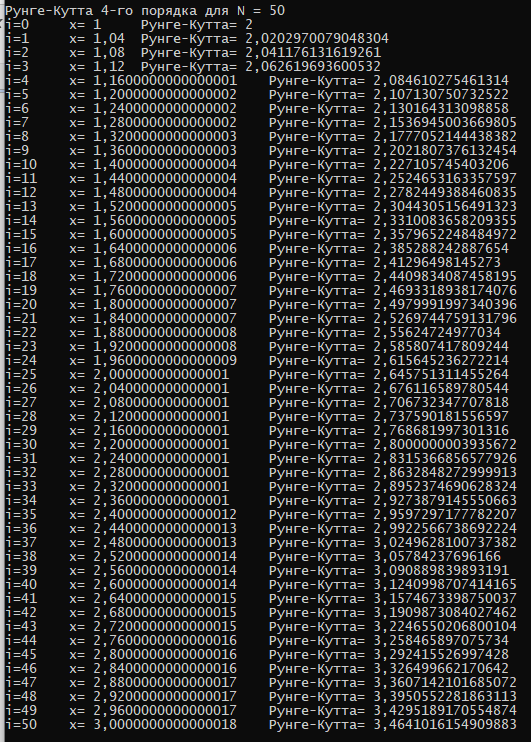


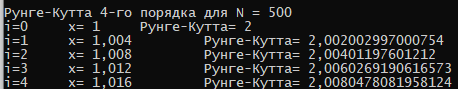


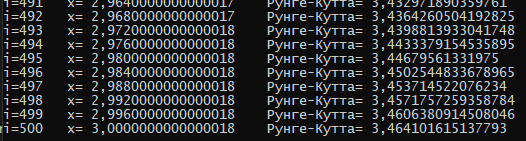


**Метод Рунге-Кутта для всех N:**









**Вывод:**

Можно заметить, что чем больше N и, соответственно, меньше шаг, тем ближе подходит аппроксимируемое решение к реальной функции. При этом, метод Рунге-Кутта при любом шаге точнее метода Эйлера.

**Контрольные вопросы**

**1. Напишите формулу метода Эйлера. Какова точность этого метода?**

**.**

Метод Эйлера является методом первого порядка — имеет погрешность на шаге  и погрешность в целом .

**2. Напишите формулу какого-либо метода Рунге-Кутта. Какова точность этого метода?**





где  - величина шага сетки по .

Ошибка на одном шаге имеет порядок , а суммарная ошибка на конечном интервале интегрирования имеет порядок .

**3. Как численно вычисляются правая, левая и центральная производные? Какая схема вычислений даёт наименьшую погрешность?**

 - правая разностная производная

 - левая разностная производная

 - центральная разностная производная

Оценки погрешности:

Правая :

Левая : 

Центральная : 

Таким образом, центральная разностная производная аппроксимирует производную со вторым порядком точности относительно .

**4. Как численно можно найти вторую производную?**



**5. Что такое явные и неявные конечно-разностные схемы решения дифференциальных уравнений? Что в них хорошего и плохого?**

**Явной** называют схему, в которой неизвестное значение искомой функции (на i+1 шаге) стоит только в левой части, а в правой части стоят уже вычисленные ранее значения функций. Если в правой части стоят неизвестные значения функций, то схема - **неявная**.

Преимущества неявных схем:

* устойчивость на любом шаге
* для любой явной схемы с шагом  найдется неявная схема с таким же шагом с точностью в два раза больше, чем у явной.

**6. Что такое жёсткость системы обыкновенных дифференциальных уравнений? Физический смысл жёсткости?**

Строгого общепринятого математического определения жёстких ОДУ нет. Принято считать, что жесткие системы - это те уравнения, решение которых получить намного проще с помощью определенных неявных методов, чем с помощью явных методов. Примерно такое определение было предложено в 1950-х годах классиками в этой области Кертиссом и Хиршфельдером.

Степень различия порядков коэффициентов при разных слагаемых чаще всего и определяет жесткость системы ОДУ. В качестве соответствующей характеристики выбирают матрицу Якоби (якобиан) векторной функции , т.е. функциональную матрицу, составленную из производных . Чем вырожденнее матрица Якоби, тем жестче система уравнений.

Физический смысл жёсткости:

* ОДУ моделирует процесс, компоненты которого обладают несоизмеримыми характерными временами
* ОДУ моделирует процесс, характерное время которого намного меньше отрезка интегрирования

**7. Как решают задачу Коши в случае дифференциального уравнения высокого порядка?**

Численное решение задачи Коши

,

, ,, … , 

состоит в построении таблицы приближенных значений  решения  в точках .

Задача о численном решении дифференциального уравнения порядка выше первого чаще всего сводится к численному решению задачи Коши для нормальной системы дифференциальных уравнений.

Обозначив

, , , … , 

получим задачу Коши для системы n дифференциальных уравнений 1-го порядка

, ,…, ,

, ,…, 

которая в векторной форме имеет вид

, 

, 



Численное решение задачи Коши для этой системы состоит в построении таблицы приближенных значений yi,1 , yi,2 , ..., yi,N компонент yi(xj) вектора решения в точках . Чтобы получить расчетные формулы метода Рунге—Кутты для систем дифференциальных уравнений, достаточно в расчетных формулах для уравнений первого порядка заменить

y, f(x, y), k1, k2, k3, k4 на , , `k1, `k2, `k3, `k4 .