**Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования**

**"Уфимский государственный авиационный технический университет"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Теория разностных схем.

**Отчет по лабораторной работе № 1**

**Тема:** «Решение начальных и краевых задач для обыкновенных дифференциальных уравнений»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа ПМ-353 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Шамаев И.Р. |  |  |  |
| Принял | Белевцов Н.С. |  |  |  |

**Уфа 2022**

**Цель работы:** получить навык численного решения задач для обыкновенных дифференциальных уравнений с использованием различных методов на примере задачи Коши для системы обыкновенных дифференциальных уравнений первого порядка и начально-краевой задачи для обыкновенного дифференциального уравнения второго порядка.

**Теоретический материал**

**Задача Коши для системы ОДУ**

Рассматривается задача Коши для системы уравнений движения материальной точки в потенциальном полк U(x):

Рассматриваемые численные методы решения

1. Метод Эйлера с постоянным шагом.
2. Явная двухшаговая схема Адамса.
3. Метод Рунге-Кутта 4-го порядка.

**Метод Эйлера**

Рассмотрим задачу Коши для ОДУ первого порядка

Обозначим за h шаг сетки и грубо аппроксимируем приращение функции:

Подставляя исходное уравнение в правую часть и заменяя переменную и функцию сеточными функциями, получим схему для метода Эйлера:

**Метод Адамса**

Аппроксимируем приращение функции более точно:

Экстраполируем функцию f(t, u(t)) линейно по уже известным двум значениям в точках tn-1 и tn и проинтегрируем. После преобразования получим явную схему Адамса второго порядка:

Так как схема для расчета нового значения требует два предыдущих, дополним начальное условие значением, посчитанным, например, методом Эйлера:

**Метод Рунге-Кутты 4 порядка**

Данный метод строит четырехчленную схему на основе разложения функции погрешности в ряд Тейлора и приравнивания первых четырех ее производных к нулю.

Наиболее употребительная схема:

**Задания на лабораторную работу**

***I. Задача Коши для системы уравнений движения***

Рассматривается задача Коши для системы уравнений движения материальной точки в потенциальном поле :

|  |  |
| --- | --- |
|  | (1) |

Параметры задачи выбираются в соответствии с индивидуальным заданием (Таблица 1). Перед началом решения задачи необходимо привести ее к безразмерному виду, выбрав подходящие масштабы для всех величин.

***Задача 1 (2 балла).***

1. Написать вычислительную программу на языке программирования C++ решения задачи (1) методом Эйлера с постоянным шагом.
2. Исследовать зависимость решения при больших временах от величины шага временной сетки. Построить графики решений для различных значений шага.
3. Выполнить сравнение полученных решений с численным решением в каком-либо математическом пакете, полученным с помощью метода высокого порядка точности (например, Рунге-Кутта 4–5). Построить графики разности решений.
4. Проверить применимость правила Рунге и с его помощью повысить точность решения.

***Задача 2 (2 балла).***

1. Написать вычислительную программу на языке программирования C++ решения задачи (1) по явной двухшаговой схеме Адамса с постоянным шагом.
2. Исследовать зависимость решения при больших временах от величины шага временной сетки. Построить графики решений для различных значений шага.
3. Выполнить сравнение полученных решений с решением по методу Эйлера (задача 1) и численным решением в каком-либо математическом пакете, полученным с помощью метода высокого порядка точности (например, Рунге-Кутта 4–5). Построить графики разности решений.

***Задача 3 (2 балла).***

1. Написать вычислительную программу на языке программирования C++ решения задачи (1) методом Рунге-Кутта 4-го порядка.
2. Исследовать зависимость решения при больших временах от величины шага временной сетки. Построить графики решений для различных значений шага.
3. Выполнить сравнение полученных решений с численным решением в каком-либо математическом пакете, полученным с помощью метода высокого порядка точности (например, Рунге-Кутта 4–5). Построить графики разности решений.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 |  |  |

Решается следующая краевая задача для неоднородного ОДУ второго порядка:

|  |  |
| --- | --- |
|  | (2) |

Параметры задачи выбираются в соответствии с индивидуальным заданием (Таблицы 2 и 3).

***Задача 4 (3 балла).***

1. Написать вычислительную программу на языке программирования C++ решения задачи (2) конечно-разностным методом с решением получающейся СЛАУ методом прогонки.
2. Исследовать зависимость решения от величины шага сетки. Построить графики решений для различных значений шага.
3. Выполнить сравнение полученных решений с численным решением в каком-либо математическом пакете. Построить графики разности решений.

***Задача 5 (2 балла).***

1. Написать вычислительную программу на языке программирования C++ решения задачи (2) методом стрельбы (пристрелки). Решение соответствующей задачи Коши выполнить методом Рунге-Кутта 4-го порядка (использовать результаты задачи 3).
2. Выполнить сравнение полученного решения с решением, полученным в задаче 4.

**Практическая реализация**

В первых трех заданиях решается следующая задача Коши:

**Задача 1: Метод Эйлера**

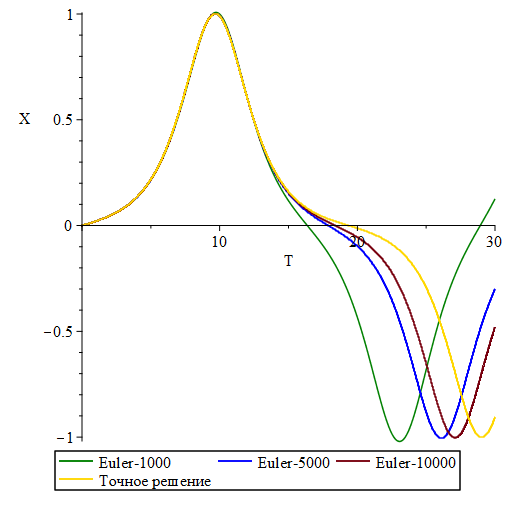


Рисунок 1. Сравнение «точного» решения с методом Эйлера при N=1000, N=5000, 10000

По графику (рисунок 1) видно, что с увеличением размерности график решения методом Эйлера «приближается» к «точному» решению, однако даже на N=20000 не достигает его.

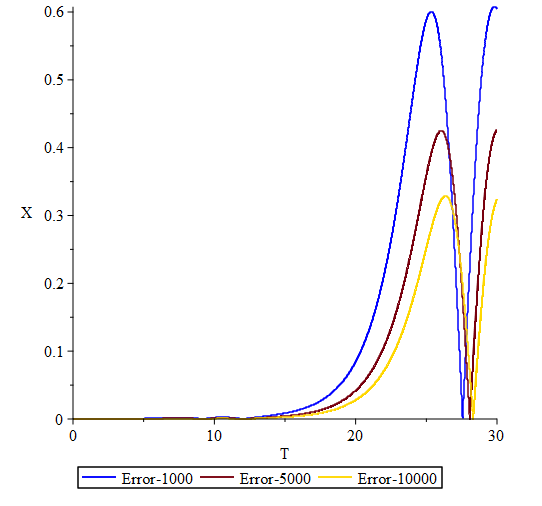
**

Рисунок 2. График погрешности для метода Эйлера

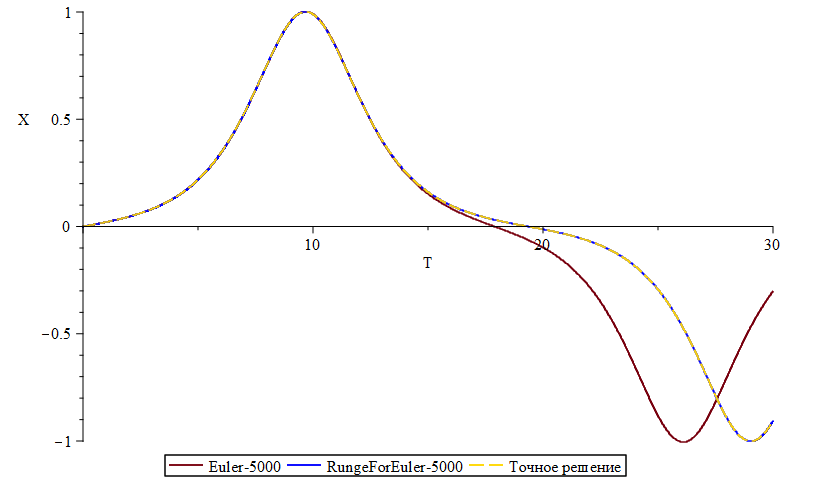


Рисунок 3. Применение правила Рунге для метода Эйлера

По графику (рисунок 2) видно, что погрешность метода Эйлера сильно возрастает при пересечении оси абсцисс и дальше продолжает возрастать.

Применив правило Рунге (рисунок 3), можно значительно уменьшить погрешность и приблизить график к «точному» решению.

**Задача 2: Метод Адамса**

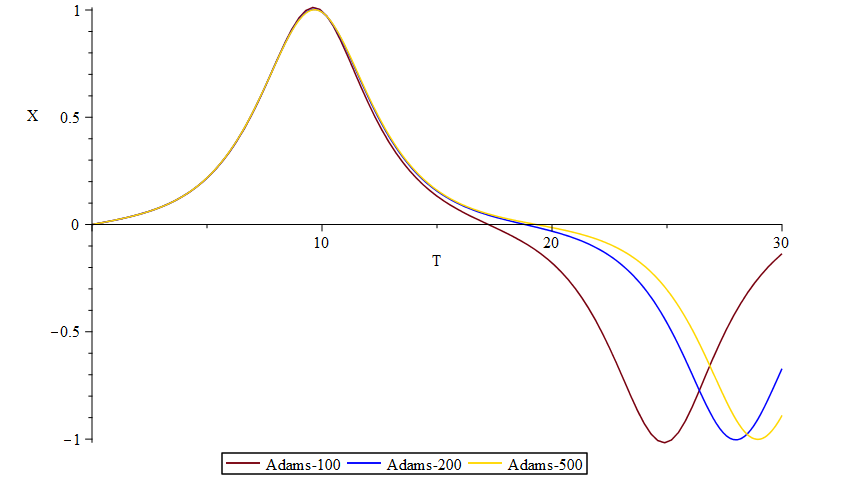


Рисунок 4. График решения системы методом Адамса при N=100, 200 и 500

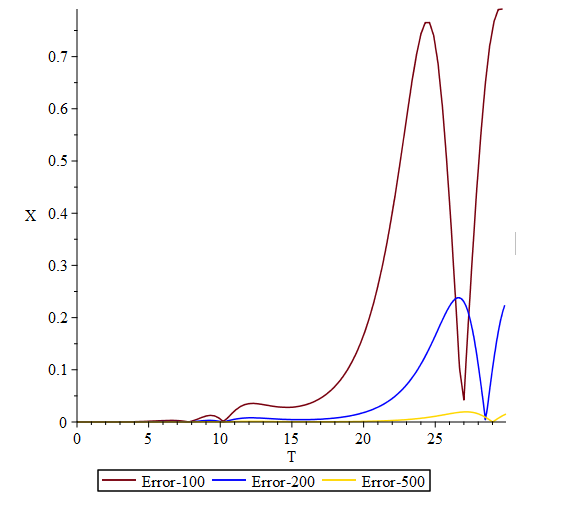


Рисунок 5. График погрешности для метода Адамса

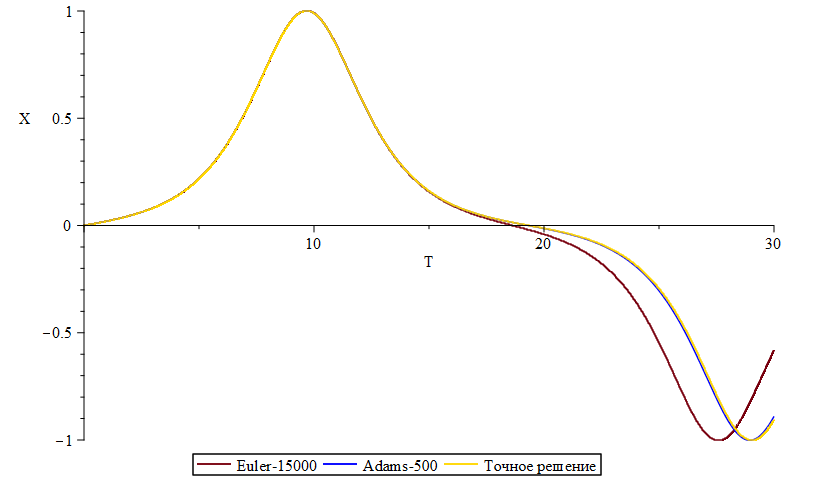


Рисунок 5. Сравнение решения методом Адамса, Эйлера и Рунге-Кутта

По графику на рисунке 5 видно, что метод Адамса с крупной сеткой значительно ближе к «точному» решению, чем метод Эйлера с мелкой сеткой.

**Задача 3: Метод Рунге-Кутта**

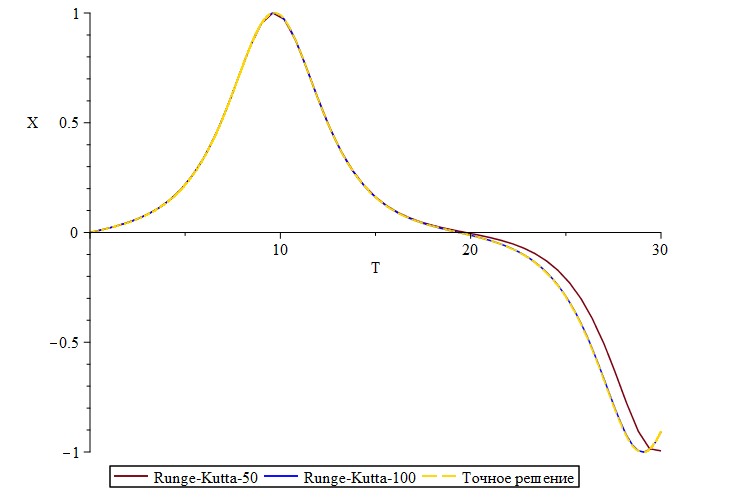


Рисунок 6. График решения методом Рунге-Кутта 4–5.

По графику видно, что начиная уже с *N=100* разница в решении методом Рунге-Кутта 4-5 порядка незначительна.

В следующих задачах решается краевая задача для неоднородного ОДУ второго порядка:

**Задача 4: Конечно-разностный метод**

Проведя расчет на сетках с узлами N=15, 32, 64 и 128 получим следующий график решения (рисунок 7):

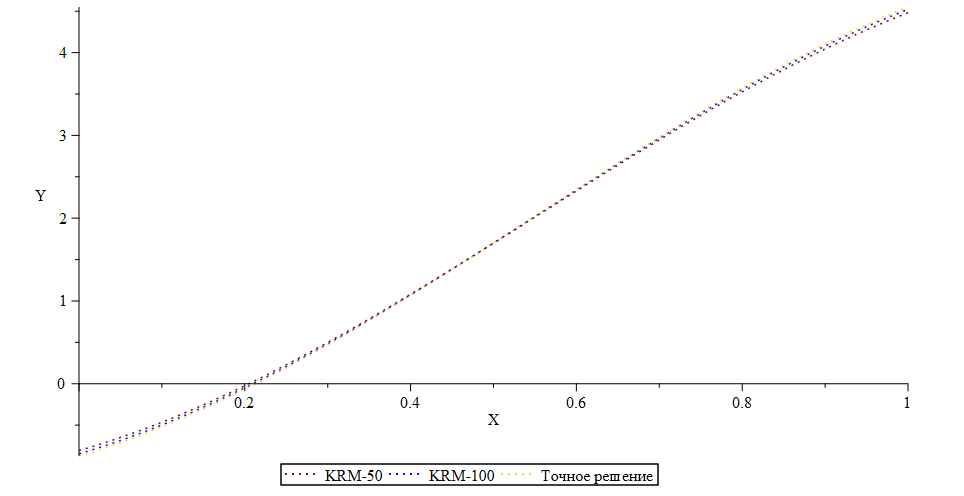


Рисунок 7. Сравнение решений конечно-разностным методом на сетках N=25, 50 и 100

По графику видно, что увеличение числа узлов с 25 до 100 незначительно влияет на решение.

**Задача 5: Метод стрельбы (пристрелки)**

Проведя расчет методом стрельбы на таком же количестве узлом, что и в прошлом задании, получим:

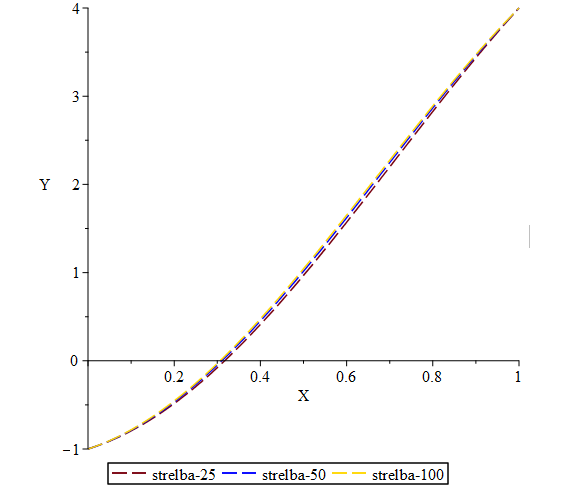


Рисунок 8. Сравнение решений методом стрельбы для N=25, 50 и 100

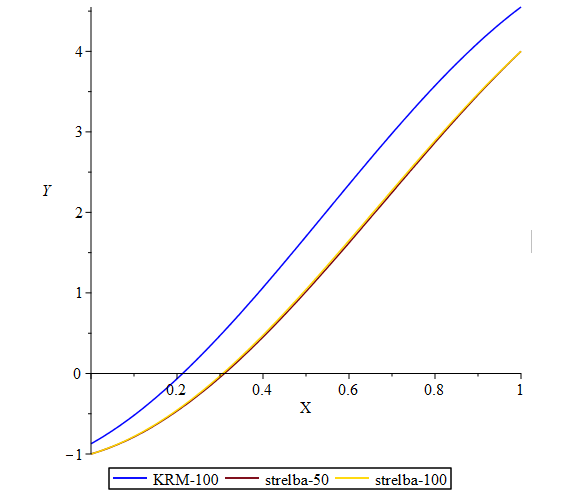
****

Рисунок 9. Сравнение решений методом стрельбы и решений конечно-разностным методом

Сравнив график решения методом стрельбы и график решения конечно-разностным методом можно сделать вывод, что метод стрельбы менее зависим от крупности сетки.

**Вывод**

В результате проделанной лабораторной работы был изучен теоретический материал необходимый для решения начальных и краевых задач для обыкновенных дифференциальных уравнений.

Для каждой поставленной задачи написана вычислительная программа на языке программирования С++, выполняющая необходимые построения и расчеты.

**Приложение**

#include <stdio.h>

#include <iostream>

#include <math.h>

#include <fstream>

#include <string>

#define a 0

#define b 30

using namespace std;

double dvdt (double x)

{

return (2/9.1) \* x \* (1 - 2 \* x \* x);

}

double f (double t, double x, double v)

{

return v;

}

double g (double t, double x, double v)

{

return (2/9.1) \* x \* (1 - 2 \* x \* x);

}

double Euler(int n, double h)

{

cout << "Euler Start..." << endl;

ofstream fout("Euler-"+to\_string(n)+".txt");

cout.precision(10);

fout.precision(10);

double \*x = new double[n + 1];

double \*v = new double[n + 1];

x[0] = 0;

v[0] = 0.02;

for (int i = 1; i <= n; i++)

{

v[i] = v[i - 1] + h \* dvdt(x[i - 1]);

x[i] = x[i - 1] + h \* v[i - 1];

}

for (int i = 0; i <= n; i++)

fout << a + i\*h << " " << x[i] << endl;

delete[] v;

delete[] x;

cout << "Euler done" << endl;

return 0;

}

double Adams(int n, double h)

{

cout << "Adams Start..." << endl;

double \*x = new double[n + 1];

double \*v = new double[n + 1];

double \*t = new double[n + 1];

ofstream fout("Adams-"+to\_string(n)+".txt");

cout.precision(10);

fout.precision(10);

x[0] = 0;

v[0] = 0.02;

t[0] = 0,

t[1] = t[0] + h;

x[1] = x[0] + h \* v[0];

v[1] = v[0] + h \* dvdt(x[1]);

for (int i = 2; i <= n; i++) {

t[i] = i \* h;

x[i] = x[i - 1] + (h / 2.) \* (3.0 \*v[i-1] - v[i - 2]);

v[i] = v[i - 1] + (h / 2.) \* (3.0 \*dvdt(x[i-1]) - dvdt(x[i-2]));

}

for (int i = 0; i <= n; i++) {

fout << a + i\*h << " " << x[i] << endl;

}

fout.close();

cout << "Adams Done" << endl;

return 0;

}

double RKutta(int n, double h)

{

cout << "Runge-Kutta Start..." << endl;

double \*x = new double[n + 1];

double \*v = new double[n + 1];

double \*t = new double[n + 1];

ofstream fout("Runge-Kutta-"+to\_string(n)+".txt");

cout.precision(10);

fout.precision(10);

x[0] = 0;

v[0] = 0.02;

t[0] = 0;

for (int i = 1; i <= n; i++) {

t[i] = i \* h;

double k1 = f(t[i - 1], x[i - 1], v[i - 1]);

double m1 = g(t[i - 1], x[i - 1], v[i - 1]);

double k2 = f(t[i - 1] + h / 2., x[i - 1] + h\*k1 / 2., v[i - 1] + h\*m1 / 2.);

double m2 = g(t[i - 1] + h / 2., x[i - 1] + h\*k1 / 2., v[i - 1] + h\*m1 / 2.);

double k3 = f(t[i - 1] + h / 2., x[i - 1] + h\*k2 / 2., v[i - 1] + h\*m2 / 2.);

double m3 = g(t[i - 1] + h / 2., x[i - 1] + h\*k2 / 2., v[i - 1] + h\*m2 / 2.);

double k4 = f(t[i - 1] + h, x[i - 1] + h\*k3, v[i - 1] + h\*m3 );

double m4 = g(t[i - 1] + h, x[i - 1] + h\*k3, v[i - 1] + h\*m3 );

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

v[i] = v[i - 1] + (h / 6.)\*(m1 + 2 \* m2 + 2. \* m3 + m4);

}

for (int i = 0; i <= n; i++) {

fout << a + i\*h << " " << x[i] << endl;

}

fout.close();

cout << "Runge-Kutta Done" << endl;

return 0;

}

double RungeForEuler(int n, double h) {

cout << "RungeForEuler Start..." << endl;

ofstream fout("RungeForEuler-"+to\_string(n)+".txt");

cout.precision(10);

fout.precision(10);

double \*x1 = new double[n + 1];

double \*v1 = new double[n + 1];

double \*x2 = new double[2\*n + 1];

double \*v2 = new double[2\*n + 1];

double \*t1 = new double[n + 1];

double \*t2 = new double[2\*n + 1];

double \*x = new double[n + 1];

double \*v = new double[n + 1];

double \*t = new double[n + 1];

double x0 = 0., v0 = 0.02;

x1[0] = x0, x2[0] = x0, v1[0] = v0, v2[0] = v0, t1[0] = 0., t2[0] = 0.;

double h1 = h; double h2 = h / 2.; double sigma = h2 / (h2 - h1);

for (int i = 1; i <= n; i++) {

x1[i] = x1[i - 1] + h1 \* v1[i-1];

v1[i] = v1[i - 1] + h1 \* dvdt(x1[i]);

}

for (int i = 1; i <= 2\*n; i++) {

t2[i] = i \* h2;

x2[i] = x2[i - 1] + h2 \* v2[i-1];

v2[i] = v2[i - 1] + h2 \* dvdt(x2[i]);

}

double \*v2res = new double[n + 1];

double \*x2res = new double[n + 1];

for (int i = 0; i <= n; i++) {

x2res[i] = x2[i\*2];

v2res[i] = v2[i\*2];

}

for (int i = 0; i <= n; i++) {

t[i] = i \* h;

v[i] = sigma \* v1[i] + (1. - sigma)\*v2res[i];

x[i] = sigma \* x1[i] + (1. - sigma)\*x2res[i];

}

for (int i = 0; i <= n; i++) {

fout << a + i\*h << " " << x[i] << endl;

}

fout.close();

cout << "RungeForEuler Done" <<endl;

return 0;

}

int main(int argc, char\* argv[])

{

int n = 1000;

if (argc == 2)

{

n = atoi(argv[1]);

}

double h = ((double)b - (double)a) / (double)n;

RKutta(n, h);

return 0;

}

#include <iostream>

#include <stdio.h>

#include <fstream>

#include <math.h>

#define a 0

#define b 1

using namespace std;

double P(double x) {

return 0.0;

}

double Q(double x){

return 4.0;

}

double F(double x){

return 8.0;

}

double du\_dx(double x) {

return x;

}

double dx\_dtOne(double x, double u) {

return - 4. \* u + 8.;

}

double dx\_dtTwo(double x, double u) {

return - 4. \* u;

}

double\* progonka(int N, double \*A, double \*B, double \*C, double \*f, double \*y)

{

double \*v, \*u;

v = new double[N + 1];

u = new double[N + 1];

v[0] = -C[0] / B[0];

u[0] = f[0] / B[0];

for (int i = 1; i < N + 1; i++)

{

v[i] = -C[i] / (B[i] + A[i] \* v[i - 1]);

u[i] = (f[i] - A[i] \* u[i - 1]) / (B[i] + A[i] \* v[i - 1]);

}

y[N] = u[N];

for (int i = N - 1; i >= 0; i--)

{

y[i] = v[i] \* y[i + 1] + u[i];

}

return y;

}

double Four(int N, double h) {

cout << "KRM Started..." << endl;

double \*alpha = new double[N + 1];

double \*betta = new double[N + 1];

double \*gamma = new double[N + 1];

double \*phi = new double[N + 1];

double \*y = new double[N + 1];

double \*x = new double[N + 1];

alpha[0] = 0.;

betta[0] = -1.;

gamma[0] = 1.;

phi[0] = 3.\*h;

alpha[N] = -1.;

betta[N] = 1.;

gamma[N] = 0.;

phi[N] = 4.\*h;

x[0] = 0;

for (int i = 1; i <= N; i++) {

x[i] = i \* h;

}

for (int i = 1; i < N; i++)

{

alpha[i] = 1 - P(x[i])\*h / 2;

betta[i] = -2 + Q(x[i])\*h\*h;

gamma[i] = 1 + P(x[i])\*h / 2;

phi[i] = F(x[i])\*h\*h;

}

progonka(N, alpha, betta, gamma, phi, y);

//Output

ofstream fout("KRM-"+to\_string(N)+".txt");

for (int i = 0; i <= N; i++) {

fout << a + x[i] << " " << y[i] << endl;

}

fout.close();

cout << "KRM Done" << endl;

return 0;

}

double RKuttaOne(int N, double h, double \*s) {

double \*x = new double[N + 1];

double \*u = s;

double x0One = 0, u0One = -1;

x[0] = x0One, u[0] = u0One;

for (int i = 1; i <= N; i++) {

//for u

double u1 = h \* (x[i - 1]);

double u2 = h \* (x[i - 1] + h / 2.);

double u3 = h \* (x[i - 1] + h / 2.);

double u4 = h \* (x[i - 1] + h);

u[i] = u[i - 1] + (1. / 6.)\*(u1 + 2 \* u2 + 2. \* u3 + u4);

//for x

double x1 = h \* (dx\_dtOne(x[i - 1], u[i - 1]));

double x2 = h \* (dx\_dtOne(x[i - 1] + h / 2., u[i - 1] + x1 / 2.));

double x3 = h \* (dx\_dtOne(x[i - 1] + h / 2., u[i - 1] + x2 / 2.));

double x4 = h \* (dx\_dtOne(x[i - 1] + h, u[i - 1] + x3));

x[i] = x[i - 1] + (1. / 6.)\*(x1 + 2 \* x2 + 2. \* x3 + x4);

}

return \*u;

}

double RKuttaTwo(int N, double h, double \*s) {

double \*x = new double[N + 1];

double \*u = s;

double x0Two = -1, u0Two = 0;

x[0] = x0Two, u[0] = u0Two;

for (int i = 1; i <= N; i++) {

//for u

double u1 = h \* (x[i - 1]);

double u2 = h \* (x[i - 1] + h / 2.);

double u3 = h \* (x[i - 1] + h / 2.);

double u4 = h \* (x[i - 1] + h);

u[i] = u[i - 1] + (1. / 6.)\*(u1 + 2 \* u2 + 2. \* u3 + u4);

//for x

double x1 = h \* (dx\_dtTwo(x[i - 1], u[i - 1]));

double x2 = h \* (dx\_dtTwo(x[i - 1] + h / 2., u[i - 1] + x1 / 2.));

double x3 = h \* (dx\_dtTwo(x[i - 1] + h / 2., u[i - 1] + x2 / 2.));

double x4 = h \* (dx\_dtTwo(x[i - 1] + h, u[i - 1] + x3));

x[i] = x[i - 1] + (1. / 6.)\*(x1 + 2 \* x2 + 2. \* x3 + x4);

}

return \*u;

}

double Five(int N, double h) {

cout << "Strelba Started!" << endl;

double \*u0 = new double[N + 1];

double \*u1 = new double[N + 1];

double \*u = new double[N + 1];

double \*x = new double[N + 1];

\*u0 = RKuttaOne(N, h, u0);

\*u1 = RKuttaTwo(N, h, u1);

for (int i = 0; i <= N; i++) {

x[i] = h \* i;

}

double c = (4. - u0[N]) / u1[N];

for (int i = 0; i <= N; i++) {

u[i] = u0[i] + c \* u1[i];

}

ofstream fout("strelba-"+to\_string(N)+".txt");

for (int i = 0; i <= N; i++) {

fout << a + x[i] << " " << u[i] << endl;

}

fout.close();

cout << "Strelba Done!" << endl;

return 0;

}

int main(int argc, char\* argv[])

{

int N = 100;

double h;

if (argc == 2)

{

N = atoi(argv[1]);

}

h = (double)((b - a) / double(N));

//Four(N, h);

Five(N, h);

return 0;

}