**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**"Уфимский государственный авиационный технический университет"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Теория разностных схем

**Отчет по лабораторной работе № 4**

«Решение краевых задач для уравнений гиперболического типа»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 краевых задач для уравнений гиперболического типа на примере начально-краевой задачи для линейного одномерного уравнения переноса и линейного одномерного неоднородного волнового уравнения.

**Теоретический материал**

**Начально-краевая задача для уравнения переноса**

Рассматривается линейная одномерная задача для уравнения переноса:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**Схема «Явный левый уголок»**

Разностная схема:

Значение сеточной функции на верхнем временном слое p + 1 рассчитывается по ее значениям на нижнем слое p:

Порядок аппроксимации схемы .

Схема устойчива при .

**Начально-краевая задача для волнового уравнения**

Рассматривается начально-краевая задача для линейного одномерного волнового уравнения с источником:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Схема «Крест»**

Разностная схема:

Вычисление решения:

На нулевом слое решение известно из второго уравнения: Первый слой вычисляют используя второе уравнение: . Краевые узлы всех слоев вычисляют с помощью двух последних уравнений:

Остальные слои вычисляются с помощью первого уравнения:

Порядок аппроксимации схемы .

Схема устойчива при .

**Неявная схема**

Разностная схема:

Где

Чтобы вычислить решение, на каждом временном слое требуется решить трехдиагональную СЛАУ.

Порядок аппроксимации схемы .

Устойчивость:

При схема безусловно устойчива, а при условие устойчивости имеет вид .

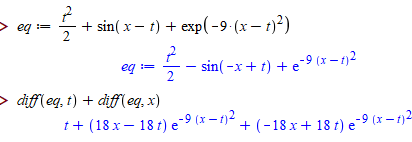
**Практическая часть**

***I. Начально-краевая задача для уравнения переноса***

Рассматривается простейшая линейная одномерная задача для уравнения переноса:

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |
|  | (3) |

Начальное и граничные условия, а также функция восстанавливаются по заданному точному решению





***Задача 1***

1. Написать вычислительную программу на языке программирования C++ решения задачи (1)-(3) с использованием явной конечно-разностной схемы с шаблоном «левый уголок» на равномерной пространственно-временной сетке.
2. Непосредственными расчетами продемонстрировать условную устойчивость схемы и справедливость условия устойчивости.
3. Исследовать зависимость решения от величины шагов сетки по пространственной и временной переменным посредством сравнения с построенным аналитическим решением. Построить графики зависимости погрешности, оцениваемой в равномерной норме по пространственной переменной, от времени и шагов сетки.

Решение:

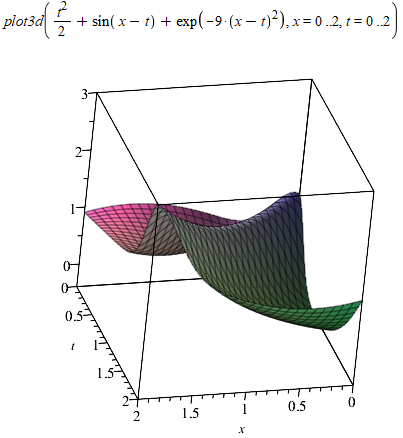


Рисунок 1. График точного решения

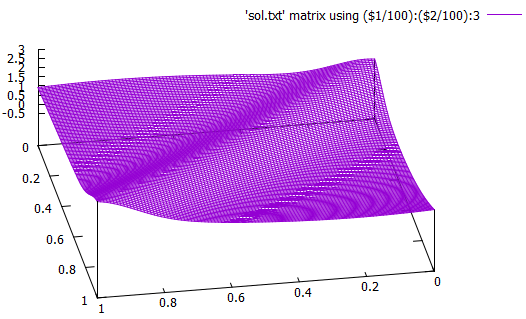


Рисунок 2. График численного решения

Для задачи необходимо выполнение условия Иначе решение по данной схеме будет расходиться.

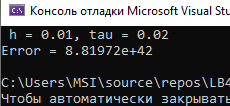


Рисунок 3. Пример выполнения программы с нарушением условия устойчивости

Исследуем зависимость решения от величины шагов сетки по пространственной и временной переменным посредством сравнения с построенным аналитическим решением

|  |  |  |  |
| --- | --- | --- | --- |
| h | N | tau | Error |
| 0,2 | 10 | 0,0001 | 0,697712 |
| 0,02 | 100 | 0,0001 | 0,235193 |
| 0,002 | 1000 | 0,0001 | 0,0322946 |
| 0,001 | 2000 | 0,0001 | 0,015793 |

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| h | tau | M | Error |
| 0,02 | 0,02 | 100 | 0,2 |
| 0,02 | 0,002 | 1000 | 0,2211 |
| 0,02 | 0,001 | 2000 | 0,2287 |
| 0,02 | 0,0002 | 10000 | 0,234477 |

Таблица 2

Построим соответствующие графики

Погрешность

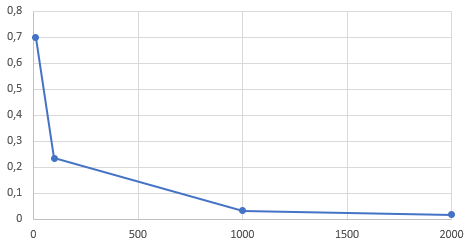
N

Рисунок 4. Зависимость погрешности решения от величины шагов сетки по пространству

Погрешность

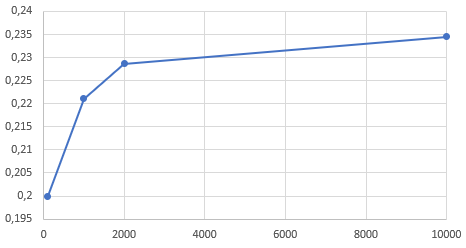
M

Рисунок 5. Зависимость погрешности решения от величины шагов сетки по времени

Как видим, при увеличении числа шагов по пространству погрешность уменьшается, а при увеличении числа шагов по времени погрешность растет.

***Задача 2***

1. Написать вычислительную программу на языке программирования C++ решения задачи (1)-(3) с использованием неявной конечно-разностной схемы с шаблоном «левый уголок» (схема «бегущего счета») на равномерной пространственно-временной сетке.
2. Выполнить сравнение точности получаемого решения по двум схемам с использованием точного решения. Построить графики погрешностей как функций координат и времени, а также графики норм погрешностей как функций шагов сетки.

Решение:

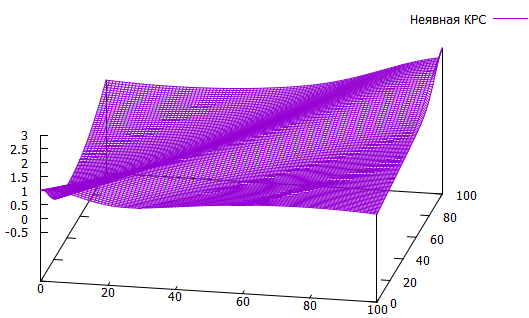


Рисунок 6. График численного решения

Исследуем зависимость решения от числа шагов сетки по пространственной и временной переменным и сравним с явной схемой

|  |  |  |
| --- | --- | --- |
| N | tau | Error |
| 10 | 0,0001 | 0,68094 |
| 100 | 0,0001 | 0,235052 |
| 1000 | 0,0001 | 0,0352692 |
| 2000 | 0,0001 | 0,01896 |

Таблица 3

|  |  |  |
| --- | --- | --- |
| h | M | Error |
| 0,02 | 100 | 0,341543 |
| 0,02 | 1000 | 0,248095 |
| 0,02 | 2000 | 0,24133 |
| 0,02 | 10000 | 0,235759 |

Погрешность

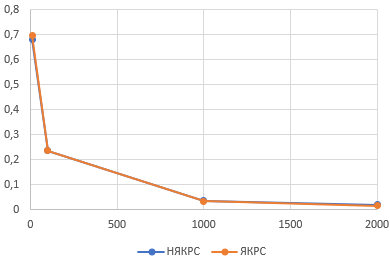
N

Рисунок 7. Зависимость погрешности решения от величины шагов сетки по пространству

Погрешность

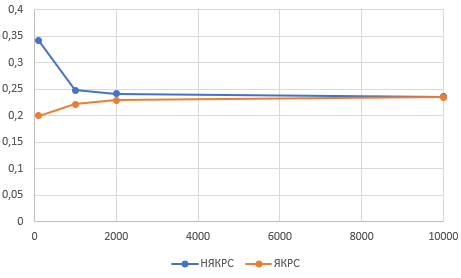
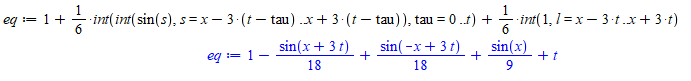
M

Рисунок 8. Зависимость погрешности решения от величины шагов сетки по времени

Рассматривается начально-краевая задача для линейного одномерного волнового уравнения с источником:

|  |  |
| --- | --- |
|  | (5) |
|  | (6) |
|  | (7) |
|  | (8) |
|  | (9) |

Аналитическое решение строим по формуле Даламбера:



***Задача 3***

1. Написать вычислительную программу на языке программирования C++ решения задачи (5)-(9) с использованием явной разностной схемы (шаблон «крест») на равномерной пространственно-временной сетке.
2. Непосредственными расчетами продемонстрировать условную устойчивость схемы и справедливость условия устойчивости.
3. Исследовать зависимость решения от величины шагов сетки по пространственной и временной переменным посредством сравнения с построенным аналитическим решением. Построить графики погрешностей как функций координат и времени, а также графики норм погрешностей как функций шагов сетки.

Решение:

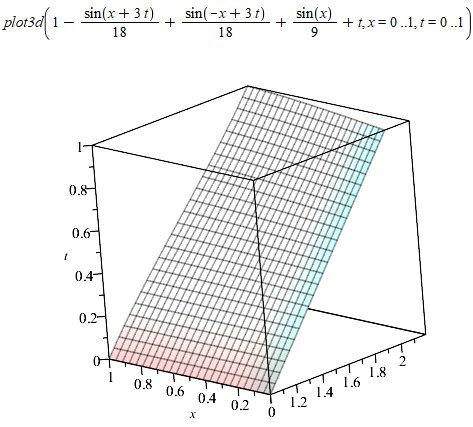


Рисунок 9. График точного решения

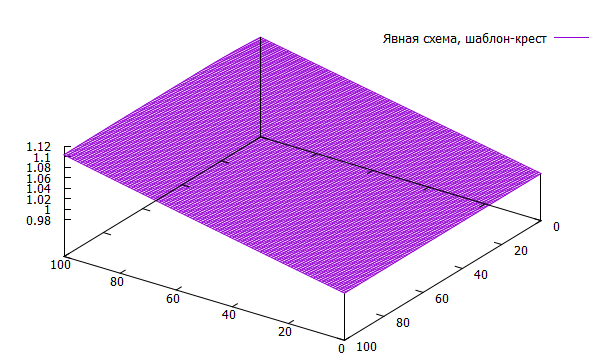


Рисунок 10. График численного решения

Для задачи необходимо выполнение условия Иначе решение по данной схеме будет расходиться.

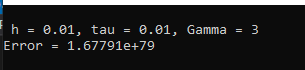


Рисунок 11.Пример выполнения программы с нарушением условия устойчивости

Исследуем зависимость решения от величины шагов сетки по пространственной и временной переменным

|  |  |  |
| --- | --- | --- |
| N | tau | Error |
| 100 | 5e-05 | 0,154346 |
| 1000 | 5e-05 | 0,0109015 |
| 5000 | 5e-05 | 0,00120157 |

Таблица 5.

|  |  |  |
| --- | --- | --- |
| h | M | Error |
| 0,01 | 1000 | 0,038 |
| 0,01 | 5000 | 0,00475 |
| 0,01 | 10000 | 0,00186 |

Таблица 6.

Далее построим графики зависимостей

Погрешность

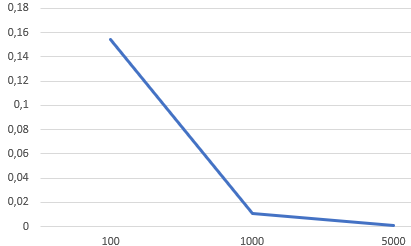
 N

Рисунок 12. Зависимость погрешности решения от величины шагов по пространству

Погрешность

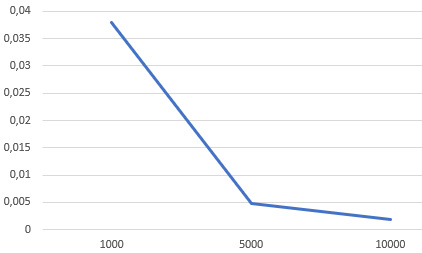
M

Рисунок 13. Зависимость погрешности решения от величины шагов сетки по времени

Таким образом, при росте числа узлов сетки погрешность уменьшается.

***Задача 4***

1. Написать вычислительную программу на языке программирования C++ решения задачи (5)-(9) с использованием неявной разностной схемы (T-образный шаблон) на равномерной пространственно-временной сетке.
2. Непосредственными расчетами продемонстрировать абсолютную устойчивость схемы (сравнением с явной схемой).
3. Исследовать зависимость решения от величины шагов сетки по пространственной и временной переменным посредством сравнения с построенным аналитическим решением. Построить графики погрешностей как функций координат и времени, а также графики норм погрешностей как функций шагов сетки.

Решение:

При выборе веса неявная схема безусловно сходится с точностью .



Выражение , погрешность вполне приемлемая для N=100, M=100, что говорит о безусловной устойчивости.

Исследуем зависимость решения от величины шагов сетки по пространственной и временной переменным

|  |  |  |
| --- | --- | --- |
| N | tau | Error |
| 100 | 5e-05 | 0,109 |
| 1000 | 5e-05 | 0,018 |
| 5000 | 5e-05 | 0,00228 |

Таблица 7.

|  |  |  |
| --- | --- | --- |
| h | M | Error |
| 0,01 | 1000 | 0,04008099 |
| 0,01 | 5000 | 0,007868 |
| 0,01 | 10000 | 0,0041 |

Таблица 8.

Погрешность

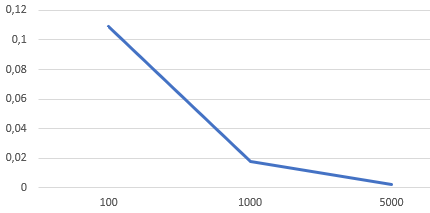
N

Рисунок 14.Зависимость погрешности решения от величины шагов по пространству

Погрешность

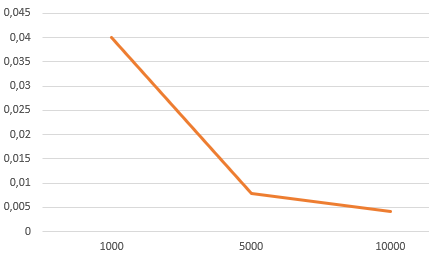
M

Рисунок 15.Зависимость погрешности решения от величины шагов по времени

***Задача 5***

Решение:

Исследуем зависимость решения от величины шагов сетки по пространственной и временной переменным

|  |  |  |
| --- | --- | --- |
| N | tau | Error |
| 100 | 5e-05 | 0,109752 |
| 1000 | 5e-05 | 0,0180422 |
| 5000 | 5e-05 | 0,00238 |

Таблица 9.

|  |  |  |
| --- | --- | --- |
| h | M | Error |
| 0,01 | 1000 | 0,0408591 |
| 0,01 | 5000 | 0,00791045 |
| 0,01 | 10000 | 0,00402989 |

Таблица 10.

Погрешность

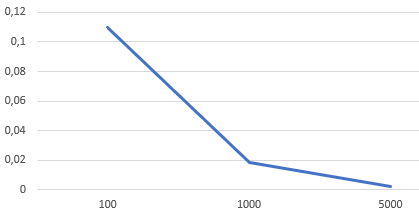
N

Рисунок 16.Зависимость погрешности решения от величины шагов по пространству

Погрешность

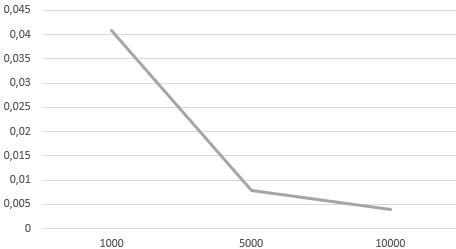
M

Рисунок 17.Зависимость погрешности решения от величины шагов по времени

Построим графики для наглядной оценки точности методов

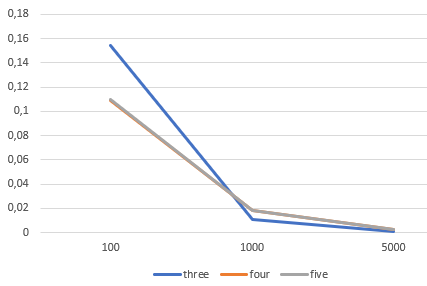


Рисунок 18.Зависимость погрешности решения от величины шагов по пространству

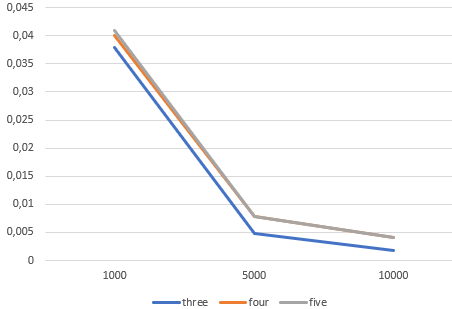


Рисунок 19.Зависимость погрешности решения от величины шагов по времени

Если сравнить результаты численного эксперимента с результатами предыдущих задачи, то видим, что точность схемы с весами и схемы повышенного порядка почти одинакова.

**Вывод**

В ходе выполнения лабораторной работы был получен навык численного решения краевых задач для уравнений гиперболического типа на примере начально-краевой задачи для линейного одномерного уравнения переноса и линейного одномерного неоднородного волнового уравнения. Было проведено сравнение методов решения.

**Приложение**

Задачи 1-2:

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <stdio.h>

#include <fstream>

#include <math.h>

#include <iomanip>

using namespace std;

const double C = 1.;

double f(double x, double t) {

return t \* t / 2 + sin(x - t) + exp(-9 \* (x - t) \* (x - t));

}

double F(double x, double t) {

return t;

}

double explicit\_sch(int N, int M, double h, double tau) {

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

double\* T = new double[M + 1];

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

double gamma = C \* tau / h;

for (int i = 0; i < N + 1; i++)

{

xx[i] = i \* h;

}

for (int i = 0; i < M + 1; i++)

{

T[i] = i \* tau;

}

for (int t = 0; t < M + 1; t++) {

u[t] = new double[N + 1];

}

for (int i = 0; i < M + 1; i++)

{

for (int j = 0; j < N + 1; j++)

{

u[i][j] = 0.0;

}

}

for (int j = 0; j < N + 1; j++)

{

u[0][j] = f(j \* h, 0);

}

for (int t = 0; t < M; t++)

{

u[t + 1][0] = f(0, (t + 1) \* tau);

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

{

u[t + 1][x] = (1. - gamma) \* u[t][x] + gamma \* u[t][x - 1] + F(xx[x], T[t]) \* tau;

}

}

ofstream fl("sol.txt");

double pogr = 0;

for (int tt = 0; tt < M + 1; tt++)

{

for (int j = 0; j < N + 1; j++)

{

double analsol;

analsol = T[tt] \* T[tt] / 2 + sin(xx[j] - T[tt]) + exp(-9 \* (xx[j] - T[tt]) \* (xx[j] - T[tt]));

double delta = abs(analsol - u[tt][j]);

fl << u[tt][j] << "\t";

if (delta > pogr) {

pogr = delta;

}

}

fl << endl;

}

fl.close();

cout << "Error = " << pogr << endl;

return 0;

}

double notexplicit\_sch(int N, int M, double h, double tau) {

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

double\* T = new double[M + 1];

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

double gamma = C \* tau / h;

for (int i = 0; i < N + 1; i++)

{

xx[i] = i \* h;

}

for (int i = 0; i < M + 1; i++)

{

T[i] = i \* tau;

}

for (int t = 0; t < M + 1; t++) {

u[t] = new double[N + 1];

}

for (int i = 0; i < M + 1; i++)

{

for (int j = 0; j < N + 1; j++)

{

u[i][j] = 0.0;

}

}

for (int j = 0; j < N + 1; j++)

{

u[0][j] = f(j \* h, 0);

}

for (int t = 0; t < M; t++)

{

u[t + 1][0] = f(0, (t + 1) \* tau);

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

{

u[t + 1][x] = (u[t][x] + (tau / h) \* u[t + 1][x - 1] + F(xx[x], T[t + 1]) \* tau) / (1. + tau / h);

}

u[t + 1][N] = f(N \* h, (t + 1) \* tau);

}

ofstream fl("sol.txt");

double pogr = 0;

for (int tt = 0; tt < M + 1; tt++)

{

for (int j = 0; j < N + 1; j++)

{

double analsol;

analsol = T[tt] \* T[tt] / 2 + sin(xx[j] - T[tt]) + exp(-9 \* (xx[j] - T[tt]) \* (xx[j] - T[tt]));

double delta = abs(analsol - u[tt][j]);

fl << u[tt][j] << "\t";

if (delta > pogr) {

pogr = delta;

}

}

fl << endl;

}

fl.close();

cout << "Error = " << pogr << endl;

return 0;

}

int main()

{

int N = 10000;

int M =20000;

double h, tau;

double a = 0., b = 2., T = 2.;

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

tau = (double)T / (double)M;

cout << " h = " << h << ", tau = " << tau << endl;

explicit\_sch(N, M, h, tau);

//notexplicit\_sch(N, M, h, tau);

return 0;

}

Задачи 3-5:

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <stdio.h>

#include <fstream>

#include <math.h>

#include <iomanip>

using namespace std;

double fiOne(double x) {

return 1;

}

double fiTwo(double x) {

return 1;

}

double F(double x, double t) {

return sin(x);

}

double psiOne(double x, double t) {

return -cos(3 \* t) / 9 + 1 / 9;

}

double psiTwo(double x, double t) {

return 1 - sin(1 + 3 \* t) / 18 + sin(-1 + 3 \* t) / 18 + sin(1) / 9 + t;

}

double sol(double x, double t) {

return 1 - sin(x + 3 \* t) / 18 + sin(-x + 3 \* t) / 18 + sin(x) / 9 + t;

}

double Gamma(double C, double h, double tau) {

return C \* tau / h;

}

double Three(int N, int Mm, double h, double tau) {

int M = N;

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

double\* T = new double[M + 1];

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

for (int i = 0; i < N + 1; i++)

{

xx[i] = i \* h;

}

for (int i = 0; i < M + 1; i++)

{

T[i] = i \* tau;

}

for (int t = 0; t < M + 1; t++) {

u[t] = new double[N + 1];

}

for (int i = 0; i < M + 1; i++)

{

for (int j = 0; j < N + 1; j++)

{

u[i][j] = 0.0;

}

}

for (int j = 0; j < N + 1; j++)

{

u[0][j] = 1.;

}

for (int j = 0; j < N + 1; j++)

{

//u[1][j] = 2 - sin(j \* h + 3) / 18 - sin(j \* h - 3) / 18 + sin(j \* h) / 9;

u[1][j] = tau + u[0][j];

}

for (int t = 1; t < M; t++)

{

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

{

u[t + 1][x] = 2. \* u[t][x] - u[t - 1][x] + (9. \* tau \* tau / h / h) \* (u[t][x + 1] - 2. \* u[t][x] + u[t][x - 1]) + (1 - sin(xx[x] + 3 \* T[t+1]) / 18 + sin(-xx[x] + 3 \* T[t+1]) / 18 + sin(xx[x]) / 9 + T[t+1]) \* tau \* tau;

}

u[t + 1][0] = 1 + (t + 1) \* tau;

u[t + 1][N] = psiTwo(N, (t + 1) \* tau);

}

ofstream fl("three.txt");

double pogr = 0;

for (int tt = 0; tt < M + 1; tt++)

{

for (int j = 0; j < N + 1; j++)

{

double analsol;

analsol = 1 - sin(xx[j] + 3 \* T[tt]) / 18 + sin(-xx[j] + 3 \* T[tt]) / 18 + sin(xx[j]) / 9 + T[tt];

fl << u[tt][j] << "\t";

double delta = abs(analsol - u[tt][j]);

if (delta > pogr) {

pogr = delta;

}

}

fl << endl;

}

fl.close();

cout << "Error= " << pogr << endl;

return 0;

}

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, int M, double h, double tau) {

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

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

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

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

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

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

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

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

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

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

tau = 1. / M;

double sigma = 1. / 4.;

for (int i = 0; i < N + 1; i++)

{

x[i] = i \* h;

uPrevPrev[i] = fiOne(x[i]);

uprev[i] = fiTwo(x[i]) \* tau + uPrevPrev[i];

unext[i] = 0;

G[i] = 0;

}

for (int i = 0; i < N + 1; i++)

{

t[i] = i \* tau;

}

double pogr = 0;

for (int tt = 0; tt < N + 1; tt++)

{

A[0] = 0.;

B[0] = -1.;

C[0] = 1.;

A[N] = 0.;

B[N] = 1.;

C[N] = 0.;

F[0] = h \* psiOne(0, t[tt]);

F[N] = psiTwo(N, t[tt]);

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

{

A[j] = 9. \* ((sigma \* tau \* tau) / (h \* h));

B[j] = 9. \* (-2. \* ((sigma \* tau \* tau) / (h \* h))) - 1.;

C[j] = 9. \* ((sigma \* tau \* tau) / (h \* h));

G[j] = 9. \* sigma \* (uPrevPrev[j - 1] - 2. \* uPrevPrev[j] + uPrevPrev[j + 1]) / (h \* h) + 9. \* (1. - 2. \* sigma) \* (uprev[j - 1] - 2. \* uprev[j] + uprev[j + 1]) / (h \* h) + 1 - sin(x[j] + 3 \* t[tt]) / 18 + sin(-x[j] + 3 \* t[tt]) / 18 + sin(x[j]) / 9 + t[tt];

F[j] = -tau \* tau \* G[j] - 2. \* uprev[j] + uPrevPrev[j];

}

progonka(N + 1, A, B, C, F, unext);

ofstream fl("four.txt");

for (int i = 0; i < N + 1; i++)

{

uPrevPrev[i] = uprev[i];

uprev[i] = unext[i];

double analsol;

analsol = 1 - sin(x[i] + 3 \* t[tt]) / 18 + sin(-x[i] + 3 \* t[tt]) / 18 + sin(x[i]) / 9 + t[tt];

fl << unext[i] << "\t";

if (abs(analsol - unext[i]) > pogr) {

pogr = abs(analsol - unext[i]);

}

}

fl.close();

}

cout << "Error = " << pogr << endl;

return 0;

}

double Five(int N, int M, double h, double tauu) {

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

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

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

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

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

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

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

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

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

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

double tau = 1. / M;

double sigma = (h \* h) / (12. \* tau \* tau);

for (int i = 0; i < N + 1; i++)

{

x[i] = i \* h;

uPrevPrev[i] = fiOne(x[i]);

uprev[i] = fiTwo(x[i]) \* tau + uPrevPrev[i];

unext[i] = 0;

G[i] = 0;

}

for (int i = 0; i < N + 1; i++)

{

t[i] = i \* tau;

}

double pogr = 0;

for (int tt = 0; tt < N + 1; tt++)

{

A[0] = 0.;

B[0] = -1.;

C[0] = 1.;

A[N] = 0.;

B[N] = 1.;

C[N] = 0.;

F[0] = h \* psiOne(0, t[tt]);

F[N] = psiTwo(N, t[tt]);

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

{

A[j] = 9. \* ((sigma \* tau \* tau) / (h \* h));

B[j] = 9. \* (-2. \* ((sigma \* tau \* tau) / (h \* h))) - 1.;

C[j] = 9. \* ((sigma \* tau \* tau) / (h \* h));

G[j] = 9. \* sigma \* (uPrevPrev[j - 1] - 2. \* uPrevPrev[j] + uPrevPrev[j + 1]) / (h \* h) + 9. \* (1. - 2. \* sigma) \* (uprev[j - 1] - 2. \* uprev[j] + uprev[j + 1]) / (h \* h) + 1 - sin(x[j] + 3 \* t[tt]) / 18 + sin(-x[j] + 3 \* t[tt]) / 18 + sin(x[j]) / 9 + t[tt];

F[j] = -tau \* tau \* G[j] - 2. \* uprev[j] + uPrevPrev[j];

}

progonka(N + 1, A, B, C, F, unext);

for (int i = 0; i < N + 1; i++)

{

uPrevPrev[i] = uprev[i];

uprev[i] = unext[i];

double analsol;

analsol = 1 - sin(x[i] + 3 \* t[tt]) / 18 + sin(-x[i] + 3 \* t[tt]) / 18 + sin(x[i]) / 9 + t[tt];

double delta = abs(analsol - unext[i]);

if (delta > pogr) {

pogr = delta;

}

}

}

cout << "Error = " << pogr << endl;

return 0;

}

int main()

{

int N = 100;

int M = 1000;

double h, tau;

double a = 0., b = 1., T = 1.;

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

tau = (double)T / (double)M;

cout << " \n h = " << h << ", tau = " << tau << ", Gamma = " << Gamma(3., h, tau) << endl;

//Three(N, M, h, tau);

//Four(N, M, h, tau);

Five(N, M, h, tau);

return 0;

}