**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**

**Тема:** «**Решение краевых задач для уравнений гиперболического типа.**»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа МКН-316 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Чучкалов Д.А. Хромых К. М.. |  |  |  |
| Принял | Гайнетдинова А.А. |  |  |  |

**Уфа 2024**

**Цель:** 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.

Рассматривается начально-краевая задача для нелинейного одномерного уравнения теплопроводности с источником:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | (3) |
|  |  |  | (4) |
|  |  |  |  |

**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.



Условие устойчивости схемы:

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 1. Ошибка на различных сетках

Изображение выглядит как диаграмма, рисунок, зарисовка, искусство

Автоматически созданное описание

Рисунок 2. График аналитического решения.

Изображение выглядит как диаграмма, зарисовка, дизайн

Автоматически созданное описание

Рисунок 3. График численного решения.

***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.

Изображение выглядит как зарисовка, диаграмма, рисунок, дизайн

Автоматически созданное описание

Рисунок 4. График численного решения

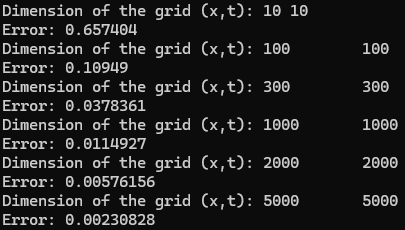


Рисунок 5. Ошибки на различных сетках.

***II. Начально-краевая задача для волнового уравнения***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | (5) |  |  | (3) |
|  | (6) |  |  | (4) |
|  | (7) |  |  |  |
|  | (8) |  |  |  |
|  | (9) |  |  |  |

Изображение выглядит как диаграмма, линия, зарисовка, текст

Автоматически созданное описание

Рисунок 6. График аналитического решения

***Задача 3.***

1. Написать вычислительную программу на языке программирования C++ решения задачи (5)-(9) с использованием явной разностной схемы 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.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7. Ошибка на различных сетках.

Изображение выглядит как линия, диаграмма, текст, График

Автоматически созданное описание

Рисунок 8. График решения нелинейного уравнения

***Задача 4 (2 балла).***

1. Написать вычислительную программу на языке программирования C++ решения задачи (5)-(9) с использованием неявной разностной схемы с весами 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.

Изображение выглядит как текст, линия, диаграмма, снимок экрана

Автоматически созданное описание

Рисунок 9. График численного решения

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 10. Ошибка на различных сетках и при весовом параметре равном 0.3.

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 11. Ошибка при несоблюдении условия устойчивости схемы.

***Задача 5 (2 балла).***

1. Написать вычислительную программу на языке программирования C++ решения задачи (5)-(9) по схеме повышенного порядка аппроксимации на равномерной сетке.
2. Выполнить сравнение точности получаемого решения с использованием точного решения. Построить графики погрешностей как функций координат и времени, а также графики норм погрешностей как функций шагов сетки.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 12. Ошибка на различных сетках и при весовом параметре равном 0.75.

Изображение выглядит как линия, График, диаграмма, текст

Автоматически созданное описание

Рисунок 12. График численного решения

Вывод: в ходе 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.

**Листинг**

*#include <iostream>*

*#include <vector>*

*#include <fstream>*

*#include <string>*

*using namespace std;*

*// x = 0..2 t>0*

*double phi\_z1(double x)*

*{*

*return sin(5 \* x) / 6 + sin(x);*

*}*

*double psi\_z1(double t)*

*{*

*return -5 \* sin(t) / 6;*

*}*

*double f1(double x, double t)*

*{*

*return cos(t + 5 \* x);*

*}*

*double u1\_analitic(double x, double t)*

*{*

*return sin(t + 5 \* x) / 6. + sin(x - t);*

*}*

*//x = 0..1, t = 0..1 (t>0)*

*double phi1\_z2(double x)*

*{*

*return 0;*

*}*

*double phi2\_z2(double x)*

*{*

*return 0;*

*}*

*double psi1\_z2(double t)*

*{*

*return t / 2. - sin(2 \* t) / 4.;*

*}*

*double psi2\_z2(double t)*

*{*

*return 0;*

*}*

*double f2(double x, double t)*

*{*

*return sin(2 \* t);*

*}*

*double f5(double x, double t, double hx)*

*{*

*return f2(x, t) - 4 \* sin(2 \* t) \* hx \* hx / 12.;*

*}*

*double u2\_analitic(double x,double t)*

*{*

*return t / 2. - sin(2 \* t) / 4.;*

*}*

*void create\_analitic\_file(int nt,int nx)*

*{*

*ofstream ftc("u\_analitic.txt");*

*double hx = 2. / nx;*

*double ht = 10. / nt;*

*for (int j = 0; j < nt; j++)*

*{*

*for (int i = 0; i < nx; i++)*

*{*

*ftc << u1\_analitic(i \* hx, j \* ht) << " ";*

*}*

*ftc << "\n";*

*}*

*}*

*vector<double> TripleDiag(vector<double> A, vector<double> B, vector<double> C, vector<double> D) {*

*vector<double> P(A.size());*

*vector<double> Q(A.size());*

*vector<double> X(A.size());*

*int N = A.size();*

*P[0] = C[0] / B[0];*

*Q[0] = D[0] / B[0];*

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

*{*

*if (i < N - 1)*

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

*Q[i] = (D[i] - A[i] \* Q[i - 1]) / (B[i] - A[i] \* P[i - 1]);*

*}*

*// backward*

*X[N - 1] = Q[N - 1];*

*for (int i = N - 2; i >= 0; --i)*

*{*

*X[i] = Q[i] - P[i] \* X[i + 1];*

*}*

*return X;*

*}*

*void z1(bool fileToCout = 0)*

*{*

*vector<int> Nt = {100,1000,10000};*

*vector<int> Nx = {10,100,10500};*

*double c = 1.;*

*double a = 0;*

*double b = 2.;*

*double T = 10.;*

*for (int k = 0; k < Nt.size(); k++)*

*{*

*double hx = (b - a) / Nx[k];*

*double ht = T / Nt[k];*

*vector<vector<double>> U(Nt[k], vector<double>(Nx[k]));*

*for (int i = 0; i < Nx[k]; i++)*

*{*

*U[0][i] = phi\_z1(i \* hx);*

*}*

*for (int j = 0; j < Nt[k]; j++)*

*{*

*U[j][0] = psi\_z1(j \* ht);*

*}*

*for (int j = 1; j < Nt[k]; j++)*

*{*

*for (int i = 1; i < Nx[k]; i++)*

*{*

*U[j][i] = (1. - c \* ht / hx) \* U[j - 1][i] + c \* ht / hx \* U[j - 1][i - 1] + ht \* f1((i - 1) \* hx, j \* ht);*

*}*

*}*

*double buf\_error = 0, maxError = -1;*

*for (int j = 1; j < Nt[k]; j++)*

*{*

*for (int i = 1; i < Nx[k]; i++)*

*{*

*buf\_error = abs(U[j][i] - u1\_analitic(i \* hx, j \* ht));*

*maxError = max(maxError, buf\_error);*

*}*

*}*

*cout << "Dimension of the grid (x,t): " << Nx[k] << "\t" << Nt[k] << "\nError: " << maxError << endl;*

*if (fileToCout)*

*{*

*ofstream ftc("z1\_" + to\_string(k) + ".txt");*

*for (int j = 0; j < Nt[k]; j++)*

*{*

*for (int i = 0; i < Nx[k]; i++)*

*{*

*ftc << U[j][i] << " ";*

*}*

*ftc << endl;*

*}*

*}*

*}*

*}*

*void z2(bool fileToCout = 0)*

*{*

*vector<int> Nt = { 10, 100,300, 1000, 2000, 5000 };*

*vector<int> Nx = { 10, 100,300, 1000, 2000, 5000 };*

*double c = 1.;*

*double a = 0;*

*double b = 2.;*

*double T = 10.;*

*for (int k = 0; k < Nt.size(); k++)*

*{*

*double hx = (b - a) / (Nx[k] - 1);*

*double ht = T / (Nt[k] - 1);*

*vector<vector<double>> U(Nt[k], vector<double>(Nx[k]));*

*for (int i = 0; i < Nx[k]; i++)*

*{*

*U[0][i] = phi\_z1(i \* hx);*

*}*

*for (int j = 0; j < Nt[k]; j++)*

*{*

*U[j][0] = psi\_z1(j \* ht);*

*}*

*for (int j = 1; j < Nt[k]; j++)*

*{*

*for (int i = 1; i < Nx[k]; i++)*

*{*

*U[j][i] = (c \* ht / hx \* U[j][i - 1] + U[j - 1][i] + ht \* f1((i - 1) \* hx, j \* ht)) \* hx / (hx + c \* ht);*

*}*

*}*

*double buf\_error = 0, maxError = -1;*

*for (int j = 1; j < Nt[k]; j++)*

*{*

*for (int i = 1; i < Nx[k]; i++)*

*{*

*buf\_error = abs(U[j][i] - u1\_analitic(i \* hx, j \* ht));*

*maxError = max(maxError, buf\_error);*

*}*

*}*

*cout << "Dimension of the grid (x,t): " << Nx[k] << "\t" << Nt[k] << "\nError: " << maxError << endl;*

*if (fileToCout)*

*{*

*ofstream ftc("z2\_" + to\_string(k) + ".txt");*

*for (int j = 0; j < Nt[k]; j++)*

*{*

*for (int i = 0; i < Nx[k]; i++)*

*{*

*ftc << U[j][i] << " ";*

*}*

*ftc << endl;*

*}*

*}*

*}*

*}*

*void z3(bool fileToCout = 0)*

*{*

*vector<int> Nt = { 10, 100,300, 1000, 2000, 5000 };*

*vector<int> Nx = { 10, 100,300, 1000, 2000, 7000 };*

*double c = 1.;*

*double a = 0;*

*double b = 1.;*

*double T = 1.;*

*for (int k = 0; k < Nt.size(); k++)*

*{*

*double hx = (b - a) / (Nx[k] - 1.);*

*double ht = T / (Nt[k] - 1.);*

*vector<vector<double>> U(Nt[k], vector<double>(Nx[k]));*

*for (int i = 0; i < Nx[k]; i++) {*

*U[0][i] = phi1\_z2(i \* hx);*

*}*

*double gamma = ht \* ht / hx / hx;*

*for (int j = 1; j < Nt[k]; j++) {*

*for (int i = 0; i < Nx[k]; i++) {*

*U[j][i] = f2(i \* hx, j \* ht) \* ht \* ht;*

*if (i == 0) {*

*U[j][i] += 2 \* gamma \* (U[j - 1][1] - U[j - 1][0] + psi1\_z2(ht \* j) \* hx - hx \* U[j - 1][0]);*

*}*

*else if (i == Nx[k] - 1) {*

*U[j][i] += 2 \* gamma \* (U[j - 1][Nx[k] - 2] - U[j - 1][Nx[k] - 1]);*

*}*

*else {*

*U[j][i] += gamma \* (U[j - 1][i + 1] - 2 \* U[j - 1][i] + U[j - 1][i - 1]);*

*}*

*if (j == 1) {*

*U[j][i] += 2 \* U[0][i];*

*U[j][i] /= 2;*

*}*

*else {*

*U[j][i] += 2 \* U[j - 1][i] - U[j - 2][i];*

*} // Fap*

*}*

*}*

*double buf\_error = 0, maxError = -1;*

*for (int j = 1; j < Nt[k]; j++)*

*{*

*for (int i = 1; i < Nx[k]; i++)*

*{*

*buf\_error = abs(U[j][i] - u2\_analitic(i \* hx, j \* ht));*

*if (buf\_error > maxError)*

*{*

*maxError = buf\_error;*

*}*

*}*

*}*

*cout << "Dimension of the grid (x,t): " << Nx[k] << "\t" << Nt[k] << "\nError: " << maxError << endl;*

*if (fileToCout)*

*{*

*ofstream ftc("z3\_" + to\_string(k) + ".txt");*

*for (int j = 0; j < Nt[k]; j++)*

*{*

*for (int i = 0; i < Nx[k]; i++)*

*{*

*ftc << U[j][i] << " ";*

*}*

*ftc << endl;*

*}*

*}*

*}*

*}*

*void z4(bool fileToCout = 0, double sigma = 0.3)*

*{*

*vector<int> Nt = { 10,100,1000,10000 };*

*vector<int> Nx = { 10,100,1000,10000 };*

*double a = 0; // start x*

*double aa = 1.; //coeff a*

*double b = 2.; // end x*

*double T = 1.; // end t*

*double eps = 1e-10;*

*for (int k = 0; k < Nt.size(); k++)*

*{*

*double hx = (b - a) / (Nx[k] - 1);*

*double ht = T / (Nt[k] - 1);*

*//sigma = 1. / 4 / (1. - eps) - hx \* hx / 4. / ht / ht;*

*vector<vector<double>> U(Nt[k], vector<double>(Nx[k]));*

*for (int i = 0; i < Nx[k]; i++)*

*{*

*U[0][i] = phi1\_z2(i \* hx);*

*}*

*vector<double> A(Nx[k]);*

*vector<double> B(Nx[k]);*

*vector<double> C(Nx[k]);*

*vector<double> D(Nx[k]);*

*////TODO*

*//j==1;*

*A[0] = 0; //closer*

*B[0] = 2. / ht / ht + 4 \* sigma / hx / hx \* (1 + hx); //bound*

*C[0] = -4 \* sigma / hx / hx; // bound*

*D[0] = 2 \* U[0][0] / ht / ht*

*+ 2 \* (1 - 2 \* sigma) / hx / hx \* (U[0][1] - U[0][0] \* (1 + hx))*

*+ 2. / hx \* (2 \* sigma \* psi1\_z2(ht) + (1 - 2 \* sigma) \* psi1\_z2(0))*

*+ f2(0, 0); // bound*

*A[Nx[k] - 1] = -4 \* sigma / hx / hx;*

*B[Nx[k] - 1] = 2. / ht / ht + 4 \* sigma / hx / hx;*

*C[Nx[k] - 1] = 0;*

*D[Nx[k] - 1] = 2 \* U[0][Nx[k] - 1] / ht / ht + 2 \* (1 - 2 \* sigma) / hx / hx \* (U[0][Nx[k] - 2] - U[0][Nx[k] - 1]) + f2((Nx[k] - 1) \* hx, 0);*

*for (int i = 1; i < Nx[k] - 1; i++) {*

*A[i] = -2 \* sigma / hx / hx;*

*B[i] = 2. / ht / ht + 4 \* sigma / hx / hx;*

*C[i] = -2 \* sigma / hx / hx;*

*D[i] = 2 \* U[0][i] / ht / ht*

*+ (1 - 2 \* sigma) / hx / hx \* (U[0][i + 1] - 2 \* U[0][i] + U[0][i - 1])*

*+ f2(i \* hx, 0);*

*}*

*U[1] = TripleDiag(A, B, C, D);*

*////////*

*//*

*//*

*//*

*////////*

*for (int j = 2; j < Nt[k]; j++) {*

*A[0] = 0; //closer*

*B[0] = 1. / ht / ht + 2 \* sigma / hx / hx \* (1 + hx); //bound*

*C[0] = -2 \* sigma / hx / hx; // bound*

*D[0] = 2 \* U[j - 1][0] / ht / ht - U[j - 2][0] / ht / ht*

*+ 2 \* (1 - 2 \* sigma) / hx / hx \* (U[j - 1][1] - U[j - 1][0] \* (1 + hx))*

*+ 2 \* sigma / hx / hx \* (U[j - 2][1] - U[j - 2][0] \* (1 + hx))*

*+ 2. / hx \* (sigma \* psi1\_z2(j \* ht) + (1 - 2 \* sigma) \* psi1\_z2((j - 1) \* ht) + sigma \* (psi1\_z2((j - 2) \* ht)))*

*+ f2(0, (j - 1) \* ht); // bound*

*A[Nx[k] - 1] = -2 \* sigma / hx / hx;*

*B[Nx[k] - 1] = 2 \* sigma / hx / hx + 1. / ht / ht;*

*C[Nx[k] - 1] = 0;*

*D[Nx[k] - 1] = (2 \* U[j - 1][Nx[k] - 1] - U[j - 2][Nx[k] - 1]) / ht / ht +*

*2 \* (1 - 2 \* sigma) / hx / hx \* (U[j - 1][Nx[k] - 2] - U[j - 1][Nx[k] - 1])*

*+ 2 \* sigma / hx / hx \* (U[j - 2][Nx[k] - 2] - U[j - 2][Nx[k] - 1])*

*+ f2((Nx[k] - 1) \* hx, (j - 1) \* ht);*

*for (int i = 1; i < Nx[k] - 1; i++) {*

*A[i] = -sigma / hx / hx;*

*B[i] = 2 \* sigma / hx / hx + 1. / ht / ht;*

*C[i] = -sigma / hx / hx;*

*D[i] = 2 \* U[j - 1][i] / ht / ht - U[j - 2][i] / ht / ht*

*+ (1 - 2 \* sigma) / hx / hx \* (U[j - 1][i + 1] - 2 \* U[j - 1][i] + U[j - 1][i - 1])*

*+ sigma / hx / hx \* (U[j - 2][i + 1] - 2 \* U[j - 2][i] + U[j - 2][i - 1])*

*+ f2(i \* hx, (j - 1) \* ht);*

*}*

*U[j] = TripleDiag(A, B, C, D);*

*}*

*/////////////// Full complete part*

*double buf\_error = 0, maxError = -1;*

*for (int j = 0; j < Nt[k]; j++)*

*{*

*for (int i = 0; i < Nx[k]; i++)*

*{*

*buf\_error = abs(U[j][i] - u2\_analitic(i \* hx, j \* ht));*

*maxError = max(maxError, buf\_error);*

*}*

*}*

*cout << "Dimension of the grid (x,t): " << Nx[k] << "\t" << Nt[k] << "\nError: " << maxError << endl;*

*if (fileToCout)*

*{*

*ofstream ftc("z4\_" + to\_string(k) + ".txt");*

*for (int j = 0; j < Nt[k]; j++)*

*{*

*for (int i = 0; i < Nx[k]; i++)*

*{*

*ftc << U[j][i] << " ";*

*}*

*ftc << endl;*

*}*

*}*

*}*

*}*

*void z5(bool fileToCout = 0,double eps = 1e-2)*

*{*

*vector<int> Nt = { 10,100,1000,10000 };*

*vector<int> Nx = { 10,100,1000,10000 };*

*double a = 0; // start x*

*double aa = 1.; //coeff a*

*double b = 2.; // end x*

*double T = 1.; // end t*

*for (int k = 0; k < Nt.size(); k++)*

*{*

*double hx = (b - a) / (Nx[k] - 1);*

*double ht = T / (Nt[k] - 1);*

*double sigma = 0.75;*

*//1. / 4. / (1 - eps) - hx \* hx / 4. / ht / ht*

*vector<vector<double>> U(Nt[k], vector<double>(Nx[k]));*

*for (int i = 0; i < Nx[k]; i++)*

*{*

*U[0][i] = phi1\_z2(i \* hx);*

*}*

*vector<double> A(Nx[k]);*

*vector<double> B(Nx[k]);*

*vector<double> C(Nx[k]);*

*vector<double> D(Nx[k]);*

*////TODO*

*//j==1;*

*A[0] = 0; //closer*

*B[0] = 2. / ht / ht + 4 \* sigma / hx / hx \* (1 + hx); //bound*

*C[0] = -4 \* sigma / hx / hx; // bound*

*D[0] = 2 \* U[0][0] / ht / ht*

*+ 2 \* (1 - 2 \* sigma) / hx / hx \* (U[0][1] - U[0][0] \* (1 + hx))*

*+ 2. / hx \* (2 \* sigma \* psi1\_z2(ht) + (1 - 2 \* sigma) \* psi1\_z2(0))*

*+ f5(0, 0,hx); // bound*

*A[Nx[k] - 1] = -4 \* sigma / hx / hx;*

*B[Nx[k] - 1] = 2. / ht / ht + 4 \* sigma / hx / hx;*

*C[Nx[k] - 1] = 0;*

*D[Nx[k] - 1] = 2 \* U[0][Nx[k] - 1] / ht / ht + 2 \* (1 - 2 \* sigma) / hx / hx \* (U[0][Nx[k] - 2] - U[0][Nx[k] - 1]) + f5((Nx[k] - 1) \* hx, 0,hx);*

*for (int i = 1; i < Nx[k] - 1; i++) {*

*A[i] = -2 \* sigma / hx / hx;*

*B[i] = 2. / ht / ht + 4 \* sigma / hx / hx;*

*C[i] = -2 \* sigma / hx / hx;*

*D[i] = 2 \* U[0][i] / ht / ht*

*+ (1 - 2 \* sigma) / hx / hx \* (U[0][i + 1] - 2 \* U[0][i] + U[0][i - 1])*

*+ f5(i \* hx, 0, hx);*

*}*

*U[1] = TripleDiag(A, B, C, D);*

*////////*

*//*

*//*

*//*

*////////*

*for (int j = 2; j < Nt[k]; j++) {*

*A[0] = 0; //closer*

*B[0] = 1. / ht / ht + 2 \* sigma / hx / hx \* (1 + hx); //bound*

*C[0] = -2 \* sigma / hx / hx; // bound*

*D[0] = 2 \* U[j - 1][0] / ht / ht - U[j - 2][0] / ht / ht*

*+ 2 \* (1 - 2 \* sigma) / hx / hx \* (U[j - 1][1] - U[j - 1][0] \* (1 + hx))*

*+ 2 \* sigma / hx / hx \* (U[j - 2][1] - U[j - 2][0] \* (1 + hx))*

*+ 2. / hx \* (sigma \* psi1\_z2(j \* ht) + (1 - 2 \* sigma) \* psi1\_z2((j - 1) \* ht) + sigma \* (psi1\_z2((j - 2) \* ht)))*

*+ f5(0, (j - 1) \* ht,hx); // bound*

*A[Nx[k] - 1] = -2 \* sigma / hx / hx;*

*B[Nx[k] - 1] = 2 \* sigma / hx / hx + 1. / ht / ht;*

*C[Nx[k] - 1] = 0;*

*D[Nx[k] - 1] = (2 \* U[j - 1][Nx[k] - 1] - U[j - 2][Nx[k] - 1]) / ht / ht +*

*2 \* (1 - 2 \* sigma) / hx / hx \* (U[j - 1][Nx[k] - 2] - U[j - 1][Nx[k] - 1])*

*+ 2 \* sigma / hx / hx \* (U[j - 2][Nx[k] - 2] - U[j - 2][Nx[k] - 1])*

*+ f5((Nx[k] - 1) \* hx, (j - 1) \* ht,hx);*

*for (int i = 1; i < Nx[k] - 1; i++) {*

*A[i] = -sigma / hx / hx;*

*B[i] = 2 \* sigma / hx / hx + 1. / ht / ht;*

*C[i] = -sigma / hx / hx;*

*D[i] = 2 \* U[j - 1][i] / ht / ht - U[j - 2][i] / ht / ht*

*+ (1 - 2 \* sigma) / hx / hx \* (U[j - 1][i + 1] - 2 \* U[j - 1][i] + U[j - 1][i - 1])*

*+ sigma / hx / hx \* (U[j - 2][i + 1] - 2 \* U[j - 2][i] + U[j - 2][i - 1])*

*+ f5(i \* hx, (j - 1) \* ht,hx);*

*}*

*U[j] = TripleDiag(A, B, C, D);*

*}*

*/////////////// Full complete part*

*double buf\_error = 0, maxError = -1;*

*for (int j = 0; j < Nt[k]; j++)*

*{*

*for (int i = 0; i < Nx[k]; i++)*

*{*

*buf\_error = abs(U[j][i] - u2\_analitic(i \* hx, j \* ht));*

*maxError = max(maxError, buf\_error);*

*}*

*}*

*cout << "Dimension of the grid (x,t): " << Nx[k] << "\t" << Nt[k] << "\nError: " << maxError << endl;*

*if (fileToCout)*

*{*

*ofstream ftc("z5\_" + to\_string(k) + ".txt");*

*for (int j = 0; j < Nt[k]; j++)*

*{*

*for (int i = 0; i < Nx[k]; i++)*

*{*

*ftc << U[j][i] << " ";*

*}*

*ftc << endl;*

*}*

*}*

*}*

*}*

*int main()*

*{*

*int Ex;*

*cout << "ex: ";*

*cin >> Ex;*

*switch (Ex)*

*{*

*default:*

*break;*

*case 1:*

*z1();*

*break;*

*case 2:*

*z2();*

*break;*

*case 3:*

*z3();*

*break;*

*case 4:*

*z4();*

*break;*

*case 5:*

*z5();*

*break;*

*case 6:*

*create\_analitic\_file(10000, 10000);*

*}*

*}*