**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 №2**

**Тема:** «**Решение начально-краевой задачи**

**для уравнения теплопроводности**»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 линейных и нелинейных начально-краевых задач для уравнений параболического типа с использованием различных конечно-разностных схем на примере задачи для одномерного уравнения теплопроводности с источником.

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

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |
|  | (3) |
|  | (4) |

**Задача 1**.

1) Написать вычислительную программу на языке программирования C++ решения задачи (1)-(4) 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 зависимости погрешности, оцениваемой в равномерной норме по пространственной переменной, от времени и шагов сетки.



Рисунок 1 ошибки на различных шагах (явная схема)

Как мы видим из (рис1), что при большем разбиении сетки - погрешность уменьшается, но на последнем шаге мы видим, что не выполняется условная сходимость, отсюда наше решение расходится.

Изображение выглядит как линия, треугольник, диаграмма, оригами

Автоматически созданное описание

Рисунок 2. График аналитического решения.

Изображение выглядит как диаграмма, линия, дизайн

Автоматически созданное описание

Рисунок 3. График численного решения.

***Задача 2.***

1. Написать вычислительную программу на языке программирования C++ решения задачи (1)-(4) по полностью неявной схеме и схеме Кранка-Николсона на равномерной 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.
3. Сравнить время решения задач по трем схемам (явной, полностью неявной и Кранка-Николсона), обеспечивающих получение решения с одинаковым уровнем погрешности.



Рисунок 3 ошибки на различных шагах сетки (неявная схема)



Рисунок 4 ошибки на различных шагах (схема Кранка-Николсона)

Решается нелинейная задача (1)-(4) с дополнительными исходными данными и

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |
|  | (3) |
|  | (4) |

***Задача 3.***

1. Написать вычислительную программу на языке программирования C++ решения задачи (1)-(4) с использованием консервативной схемы на равномерной сетке.
2. Убедиться в корректности программы на примере задачи 1.
3. Исследовать зависимость получаемого решения от величины шага сетки по пространственной и временной переменным. Построить графики решений для различных значений шага.



Рисунок 5 ошибки на различных шагах сетки (сведенная к линейной)

Изображение выглядит как линия, диаграмма, шаблон, дизайн

Автоматически созданное описание

Рисунок 6 график решения нелинейного уравнения

***Задача 4.***

1. Выполнить модификацию программы из задачи 3 путем организации внутренних итераций на каждом временном шаге для повышения точности вычисления нелинейных слагаемых. Условием остановки итерационного процесса является достижение заданного преподавателем уровня погрешности вычислений нелинейных функций.
2. Выполнить сравнение получаемых решений по исходной и модифицированной программам.
3. Сравнить время работы двух программ для построения решений с одинаковым уровнем погрешности.

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 7 Разность между обычной и модифицированной версиями программмы.



Рисунок 8 Время работы программы из 3 задания



Рисунок 9 время работы для модифицированной

Вывод: в ходе лабораторной работы был получен навык численного решения линейных и нелинейных начально-краевых задач для уравнений параболического типа с использованием различных конечно-разностных схем на примере задачи для одномерного уравнения теплопроводности с источником.

**Листинг**

// TRS2.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

#include <vector>

#include <fstream>

#include <string>

#include <iomanip>

#include <time.h>

#include <chrono>

**using** **namespace** std;

**double** F(**double** u)

{

**return** u\*u\*u;

//return 1;

}

**double** func(**double** x, **double** t)

{

**return** ((1. - x) \* (1. - x) - x \* sin(t) - 2. \* t);

}

**double** k1(**double** u)

{

**return** sin(u);

//return 1;

}

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;

}

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

{

**return** x \* cos(t) + t \* (1 - x) \* (1 - x);

}

**void** z1() {

**double** ht, hx;

vector<**double**> nX{ 10,100,130 };

vector<**double**> nT(nX.size());

**for** (**int** i = 0; i < nX.size(); i++) {

clock\_t start = clock();

hx = 1. / (nX[i] - 1);

nT[i] = 2 \* (nX[i]) \* (nX[i]);

ht = 1. / (nT[i] - 1);

vector<vector<**double**>> M(nT[i], vector<**double**>(nX[i])); // t , x ;

ofstream file\_to\_cout("z1\_" + to\_string(i + 1) + ".txt");

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

M[0][k] = hx \* k;

}

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

**for** (**int** j = 1; j < nT[i]; j++) {

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

**if** (k == 0) {

M[j][k] = j \* ht;

**continue**;

}

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

M[j][k] = (2. \* cos(j \* ht) \* hx + M[j][k - 1]) / (1. + hx);

**continue**;

}

M[j][k] = gamma \* (M[j - 1][k - 1] - 2. \* M[j - 1][k] + M[j - 1][k + 1]) + ht \* func(hx \* k, ht \* (j - 1)) + M[j - 1][k];

}

}

clock\_t end = clock();

**double** time = (**double**)(end - start) / CLOCKS\_PER\_SEC;

**for** (**int** j = 0; j < nT[i]; j++) {

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

file\_to\_cout << M[j][k] << " ";

}

file\_to\_cout << endl;

}

**double** dif;

**double** max\_dif;

**for** (**int** j = 0; j < nT[i]; j++)

{

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

{

max\_dif = 0;

dif = abs(u1\_func(k \* hx, j \* hx) - M[j][k]);

**if** (dif > max\_dif)

{

max\_dif = dif;

}

}

}

cout << "Step x: " << hx << setw(13 + nX.size()) << "Step T: " << ht

<< setw(8 + nX.size()) << "Error: " << max\_dif << setw(8 + nX.size()) << "Time: " << time << endl;

}

}

**void** z2\_p1() {

**double** ht, hx;

vector<**double**> nX{ 10,100,130 };

vector<**double**> nT = nX;

**for** (**int** i = 0; i < nX.size(); i++) {

clock\_t start = clock();

hx = 1. / (nX[i] - 1);

//nT[i] = 2 \* (nX[i]) \* (nX[i]);

ht = 1. / (nT[i] - 1);

vector<vector<**double**>> M(nT[i], vector<**double**>(nX[i])); // t , x ;

ofstream file\_to\_cout("z2\_p1" + to\_string(i + 1) + ".txt");

vector<**double**> A(nX[i]), B(nX[i]), C(nX[i]), D(nX[i]);

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

M[0][k] = k \* hx;

}

A[0] = 0;

B[0] = 1.;

C[0] = 0;

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

A[k] = -1. / hx / hx;

B[k] = 1. / ht + 2. / hx / hx;

C[k] = -1. / hx / hx;

}

A[nX[i] - 1] = -1. / hx;

B[nX[i] - 1] = 1. + 1. / hx;

C[nX[i] - 1] = 0;

**for** (**int** j = 1; j < nT[i]; j++) {

D[0] = j \* ht;

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

D[k] = func(hx \* k, ht \* j) + M[j - 1][k] / ht;

}

D[nX[i] - 1] = 2 \* cos(ht \* j);

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

}

clock\_t end = clock();

**double** time = (**double**)(end - start) / CLOCKS\_PER\_SEC;

**for** (**int** j = 0; j < nT[i]; j++) {//cout

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

file\_to\_cout << M[j][k] << " ";

}

file\_to\_cout << endl;

}

**double** dif;

**double** max\_dif;

**for** (**int** j = 0; j < nT[i]; j++)

{

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

{

max\_dif = 0;

dif = abs(u1\_func(k \* hx, j \* hx) - M[j][k]);

**if** (dif > max\_dif)

{

max\_dif = dif;

}

}

}

cout << "Step x: " << hx << setw(13 + nX.size()) << "Step T: " << ht

<< setw(8 + nX.size()) << "Error: " << max\_dif << setw(8 + nX.size()) << "Time: " << time << endl;

}

}

**void** z2\_p2() {

**double** ht, hx;

vector<**double**> nX{ 10,100,400 };

vector<**double**> nT = nX;

**for** (**int** i = 0; i < nX.size(); i++) {

clock\_t start = clock();

hx = 1. / (nX[i] - 1);

//nT[i] = 2 \* (nX[i]) \* (nX[i]);

ht = 1. / (nT[i] - 1);

vector<vector<**double**>> M(nT[i], vector<**double**>(nX[i])); // t , x ;

ofstream file\_to\_cout("z2\_p2" + to\_string(i + 1) + ".txt");

vector<**double**> A(nX[i]), B(nX[i]), C(nX[i]), D(nX[i]);

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

M[0][k] = k \* hx;

}

**for** (**int** j = 1; j < nT[i]; j++) {

A[0] = 0;

B[0] = 1;

C[0] = 0;

D[0] = j \* ht;

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

A[k] = -1. / 2. / hx / hx;

B[k] = 1. / ht + 1. / hx / hx;

C[k] = -1. / 2. / hx / hx;

D[k] = func(hx \* k, ht \* j) + 1. / 2. / hx / hx \* (M[j - 1][k - 1] + M[j - 1][k + 1] - 2 \* M[j - 1][k]) + 1. / ht \* M[j - 1][k];

}

A[nX[i] - 1] = -1. / hx;

B[nX[i] - 1] = 1 + 1. / hx;

C[nX[i] - 1] = 0;

D[nX[i] - 1] = 2 \* cos(ht \* j);

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

}

clock\_t end = clock();

**double** time = (**double**)(end - start) / CLOCKS\_PER\_SEC;

**for** (**int** j = 0; j < nT[i]; j++) {//cout

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

file\_to\_cout << M[j][k] << " ";

}

file\_to\_cout << endl;

}

**double** dif;

**double** max\_dif;

**for** (**int** j = 0; j < nT[i]; j++)

{

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

{

max\_dif = 0;

dif = abs(u1\_func(k \* hx, j \* hx) - M[j][k]);

**if** (dif > max\_dif)

{

max\_dif = dif;

}

}

}

cout << "Step x: " << hx << setw(13 + nX.size()) << "Step T: " << ht

<< setw(8 + nX.size()) << "Error: " << max\_dif << setw(8 + nX.size()) << "Time: " << time << endl;

}

}

**void** z3()

{

vector<**double**> nX {10,100,1000}; // по x

vector<**double**> nT{ 10,100,1000};// по t

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

{

ofstream result\_file("z3\_u" + to\_string(k) + ".txt");

**auto** begin = std::chrono::steady\_clock::now();

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

**double** hx = 1 / (nX[k] - 1);

vector<vector<**double**>> u(nT[k], vector<**double**>(nX[k]));

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

{

u[0][i] = i \* hx; // началка

}

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

{

vector<**double**> A(nX[k]), B(nX[k]), C(nX[k]), D(nX[k]);

//граничные

A[0] = 0;

B[0] = 1;

C[0] = 0;

D[0] = j \* ht;

A[nX[k] - 1] = -1. / hx;

B[nX[k] - 1] = 1. + 1. / hx;

C[nX[k] - 1] = 0;

D[nX[k] - 1] = 2 \* cos(j \* ht);

//

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

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

{

**double** ke = k1((u[j - 1][i + 1] + u[j - 1][i]) / 2);

**double** kw = k1((u[j - 1][i] + u[j - 1][i - 1]) / 2);

A[i] = -kw / hx / hx;

B[i] = 1. / ht + (kw + ke) / hx / hx;

C[i] = -ke / hx / hx;

D[i] = u[j - 1][i] / ht + func(i \* hx,j \* ht) \* F(u[j - 1][i]);

}

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

}

**auto** end = std::chrono::steady\_clock::now();

**auto** elapsed\_ms = std::chrono::duration\_cast<std::chrono::microseconds>(end - begin);

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

{

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

{

result\_file << u[j][i] << " ";

}

result\_file << endl;

}

**double** dif;

**double** max\_dif;

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

{

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

{

max\_dif = 0;

dif = abs(u1\_func(j \* hx, i \* ht) - u[i][j]);

**if** (dif > max\_dif)

{

max\_dif = dif;

}

}

}

cout << "Step x: " << hx << setw(13 + nX.size()) << "Step T: " << ht

<< setw(8 + nX.size()) << "Error: " << "..." << setw(8 + nX.size()) << "Time, ms: " << elapsed\_ms.count() \* 1e-3 << endl;

}

}

**void** z4(**double** eps)

{

vector<**double**> nX{ 10,100 }; // по x

vector<**double**> nT = nX;

**double** delta = 0;

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

{

ofstream result\_file("z4\_u" + to\_string(k) + ".txt");

**auto** begin = std::chrono::steady\_clock::now();

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

**double** hx = 1 / (nX[k] - 1);

vector<vector<**double**>> u(nT[k], vector<**double**>(nX[k]));

vector<**double**> previous;

vector<**double**> A(nX[k]), B(nX[k]), C(nX[k]), D(nX[k]);

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

{

u[0][i] = i \* hx; // началка

}

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

{

//граничные

A[0] = 0;

B[0] = 1;

C[0] = 0;

D[0] = j \* ht;

A[nX[k] - 1] = -1. / hx;

B[nX[k] - 1] = 1. + 1. / hx;

C[nX[k] - 1] = 0;

D[nX[k] - 1] = 2 \* cos(j \* ht);

//

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

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

{

**double** ke = k1((u[j - 1][i + 1] + u[j - 1][i]) / 2);

**double** kw = k1((u[j - 1][i] + u[j - 1][i - 1]) / 2);

A[i] = -kw / hx / hx;

B[i] = kw / hx / hx + ke / hx / hx + 1. / ht;

C[i] = -ke / hx / hx;

D[i] = func(i \* hx, j \* ht) \* F(u[j - 1][i]) / hx + 1. / ht \* u[j - 1][i];

}

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

**int** iter = 0;

**do**

{

delta = 0;

iter++;

previous.assign(u[j].begin(), u[j].end());

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

{

**double** ke = k1((u[j][i + 1] + u[j][i]) / 2);

**double** kw = k1((u[j][i] + u[j][i - 1]) / 2);

A[i] = -kw / hx / hx;

B[i] = kw / hx / hx + ke / hx / hx + 1. / ht;

C[i] = -ke / hx / hx;

}

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

**for** (**int** y = 1; y < nX[k]; y++)

{

**double** diff = abs(k1(u[j][y] / 2 + u[j][y - 1] / 2) - k1(previous[y] / 2 + previous[y - 1] / 2));

**if** (diff > delta)

{

delta = diff;

}

}

previous.clear();

} **while** (delta > eps);

cout << "step x,t: " << hx << setw(13 + nX.size()) << "iter: " << iter << " error: " << delta << endl;

}

**auto** end = std::chrono::steady\_clock::now();

**auto** elapsed\_ms = std::chrono::duration\_cast<std::chrono::microseconds>(end - begin);

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

{

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

{

result\_file << u[j][i] << " ";

}

result\_file << endl;

}

/\* cout << "Step x: " << hx << setw(13 + nX.size()) << "Step T: " << ht

<< setw(8 + nX.size()) << "Error: " << "..." << setw(8 + nX.size()) << "Time, ms: " << elapsed\_ms.count() \* 1e-3 << endl;\*/

}

}

**int** main()

{

**int** ex;

cout << "ex: " << endl;

cin >> ex;

**switch** (ex)

{

**case** 0:

{

z1();

**break**;

}

**case** 1:

{

z2\_p1();

**break**;

}

**case** 2:

{

z2\_p2();

**break**;

}

**case** 3:

{

z3();

**break**;

}

**case** 4:

{

z4(1e-6);

**break**;

}

}

}