**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**"Уфимский университет науки и технологий"**

**Кафедра** Высокопроизводительных вычислений и дифференциальных уравнений

**Дисциплина:** Теория разностных схем

**Отчет по лабораторной работе № 2**

Тема: «**РЕШЕНИЕ НАЧАЛЬНО-КРАЕВОЙ ЗАДАЧИ**

**ДЛЯ УРАВНЕНИЯ ТЕПЛОПРОВОДНОСТИ**»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа МКН-318 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Хаматвалиева А.Р. |  |  |  |
| Принял | Гайнетдинова А.А. |  |  |  |

**Уфа 2025**

**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.

**Теоретический материал**

* **Начально-краевая задача**

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

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

* **Явная разностная схема.**

При решении данной задачи (1)-(5) используется явная конечно-разностная схема с шаблоном «левый уголок» на равномерной пространственно-временной сетке:

Порядок аппроксимации данной схемы: .

* **Неявная разностная схема.**

При решении данной задачи (1)-(5) используются неявная конечно-разностная схема и схема Кранка-Николсона.

Неявная схема представляется следующим образом:

Схема Кранка-Схема Кранка-Николсона имеет вид:

Порядок аппроксимации полностью неявной схемы: , а схемы Кранка-Николсона - . Полностью неявная схема устойчива при любых соотношениях между шагами сетки схема Кранка-Николсона обычно абсолютно устойчива, выбирается как .

* **Решение нелинейной задачи с использованием консервативной схемы.**

Решается нелинейная задача (1)-(4) с дополнительными исходными данными и из таблицы 2, где

а функция и остальные данные берутся из таблицы 1.

* **Консервативная схема на равномерной сетке.**

При решении данной задачи (1)-(4) используется консервативная схема на равномерной сетке:

где

**Индивидуальное задание**

***Начально-краевая задача***

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

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

***I. Сравнение конечно-разностных схем для линейной задачи***

Рассматривается линейный случай уравнения теплопроводности (1):

(5)

Методом разделения переменных построить аналитическое решение задачи.

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

(5)

***Вариант 7:***

*– коэффициент теплопроводности*

*– функция источника*

***Решение:***

Наше уравнение является уравнением параболического типа. Решение будем искать в виде:

v(t,x) и w(t,x)должны удовлетворять условиям.

Пусть

Проверка краевого условия *:*

Проверка краевого условия :

= =

Проверка начально условия:

Таким образом, аналитическое решение

График аналитического решения показан на рисунке 1.

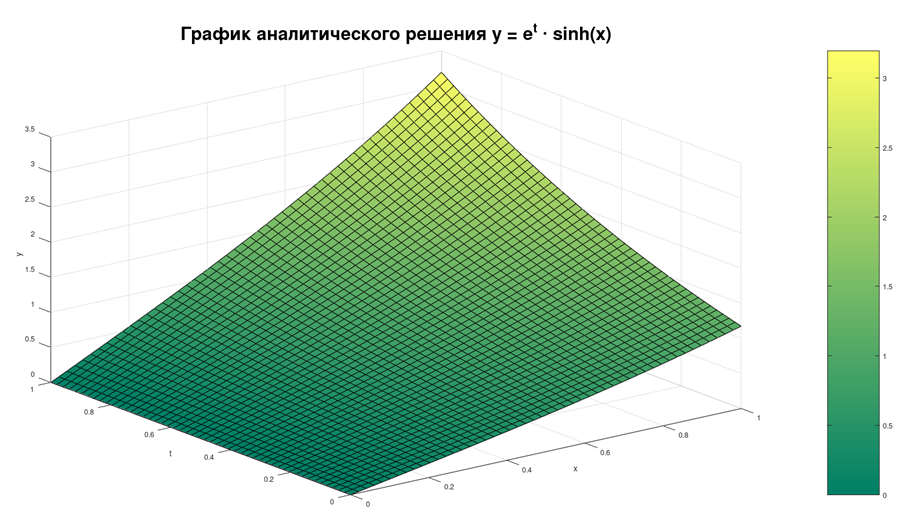
******

Рисунок 1 – График аналитического решения

***Задача 1 (2 балла).***

1. Написать вычислительную программу на языке программирования C++ решения задачи (1)-(5) 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.

На рисунке 2 показан график решения исходной задачи с использованием явной разностной схемы. Используются значения на текущем временном слое для вычисления значений на следующем. Данная схема условно устойчива. Шаг по пространству на рисунке 2 равен 0,001.

Порядок аппроксимации схемы: O(*τ* + *h*2), т.е. первый по времени и второй по пространству. Особенностью подобных схем является преимущественно условная устойчивость, то есть наличие ограничения на размер дискретных шагов по времени и пространству

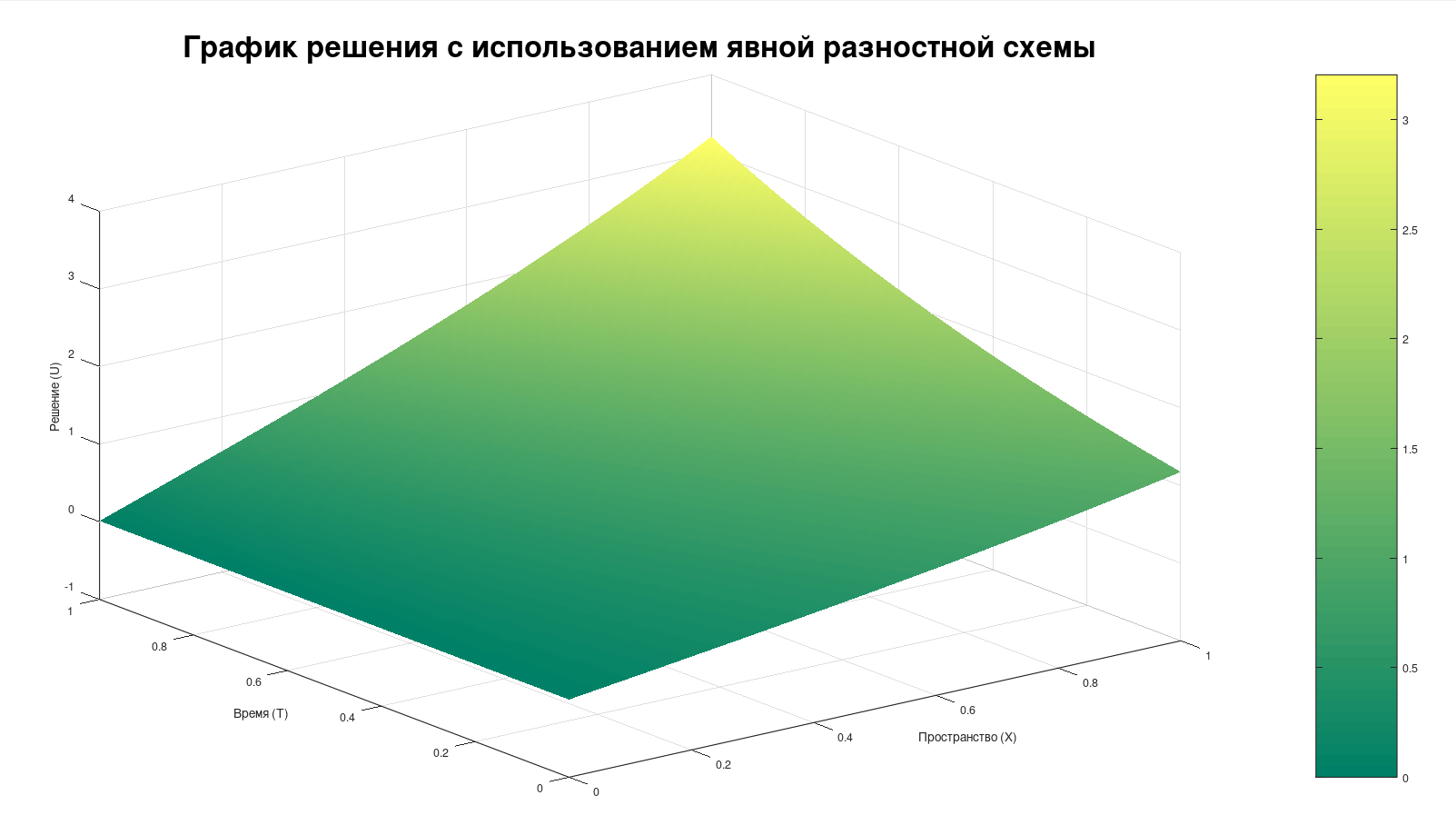
**

Рисунок 2 – Решение с использованием явной схемы

Рисунок 3 демонстрирует результаты вычисления ошибки по пространственному и временным шагам. С увеличением шага, ошибка уменьшается.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Шаг h | Шаг t | Ошибка |
| Явная схема | 0,1 | 0,005 | 0,153577 |
| Явная схема | 0,01 | 0,00005 | 0,011784 |
| Явная схема | 0,001 | 0,0000005 | 0,001150 |

Рисунок 3 – значения ошибки, используя явную схему в зависимости от шага

Условная устойчивость явной разностной схемы:

Явная разностная схема имеет следующий вид:

Сделаем замены:

Подставляем в явную разностную схему и получаем:

Необходимое условие устойчивости , то есть в нашем случае:

Так как максимальное значение достигается в и равно единице:

Таким образом, получили следующее условие устойчивости разностной схемы:

***Задача 2 (4 балла).***

1. Написать вычислительную программу на языке программирования C++ решения задачи (1)-(5) по полностью неявной схеме и схеме Кранка-Николсон 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-Николсон), обеспечивающих получение решения с одинаковым уровнем погрешности.

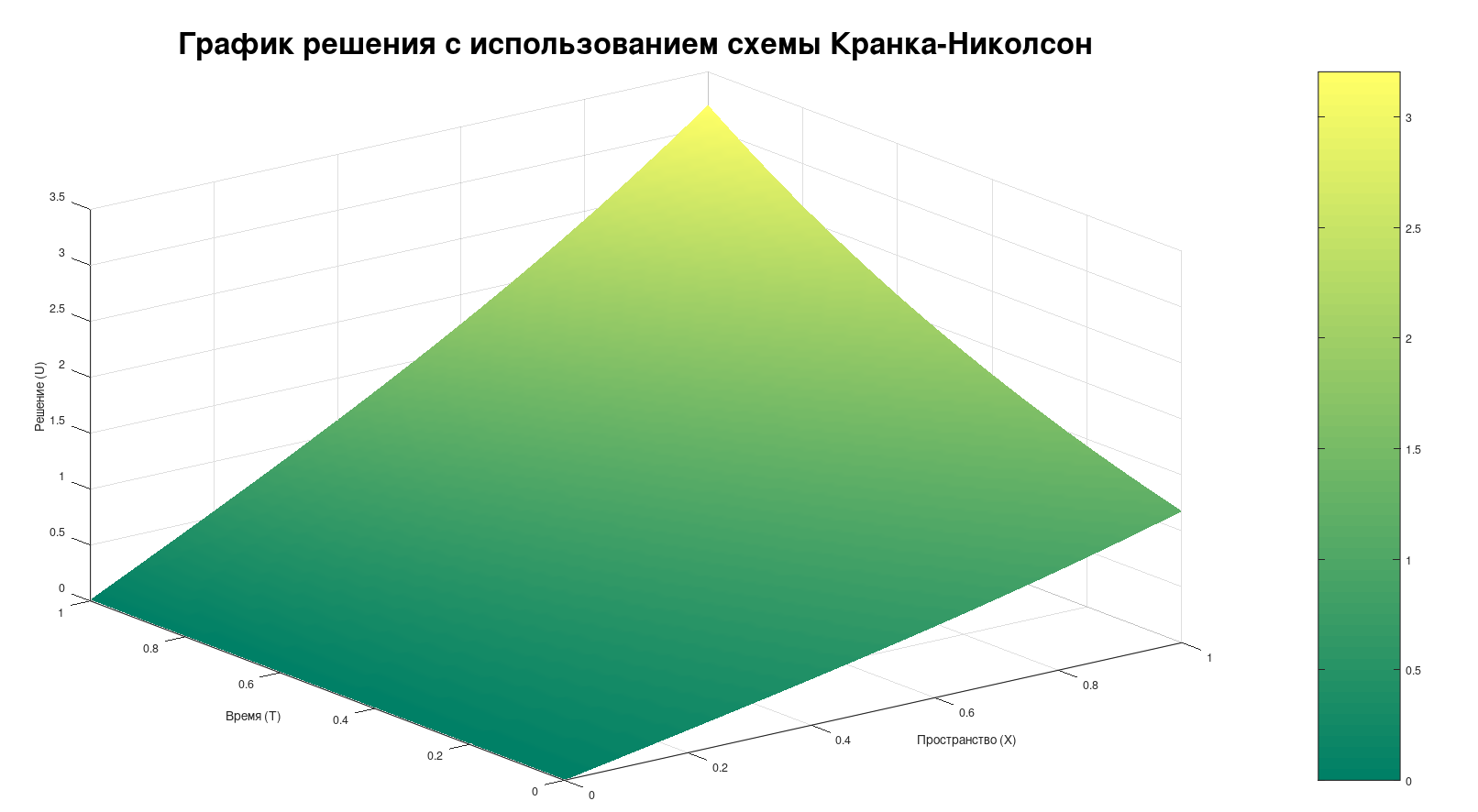


Рисунок 4 – График решения с использованием схемы Кранка-Николсона

На рисунках 4 и 5 показаны графики решения задачи с использованием схемы Кранка-Николсона и неявной разностной схемы соответственно. Неявная схема использует значения на следующем временном слое и является абсолютно устойчивой. Порядок аппроксимации схемы такой же, как и в случае явной схемы: O(*τ* + *h*2)

Схема Кранка – Николсон предполагает при записи конечно-разностных соотношений использовать узловые точки сразу с двух временных шагов. Такой подход приводит к увеличению порядка аппроксимации по времени O(*τ* 2 + *h*2). Схемы Кранка – Николсона обычно являются абсолютно устойчивыми



Рисунок 5 - График решения с использованием схемы Кранка-Николсона

Рисунки 6 и 7 демонстрируют ошибку при различных шагах по времени и пространству для неявной разностной схемы и для схемы Кранка-Николсона. По выведенным результатам можно сделать вывод, что ошибка при использовании схемы Кранка Николсона меньше. И также при уменьшении шага повышается точность решения.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Шаг h | Шаг t | Ошибка |
| Неявная схема | 0,1 | 0,005 | 0,297929 |
| Неявная схема | 0,01 | 0,00005 | 0,020395 |
| Неявная схема | 0,001 | 0,0000005 | 0,001970 |

Рисунок 6 – Ошибка вычислений в неявной схеме

|  |  |  |  |
| --- | --- | --- | --- |
|  | Шаг h | Шаг t | Ошибка |
| Схема Кранка-Николсона | 0,1 | 0,005 | 0,153577 |
| Схема Кранка-Николсона | 0,01 | 0,00005 | 0,011784 |
| Схема Кранка-Николсона | 0,001 | 0,0000005 | 0,001150 |

Рисунок 7 – ошибка вычислений, используя схему Кранка-Николсон

На рисунках 8-10 показаны графики разности аналитического решения и решения по разностной схеме Кранка-Николсона с шагами равными 0.1, 0.01, 0.001

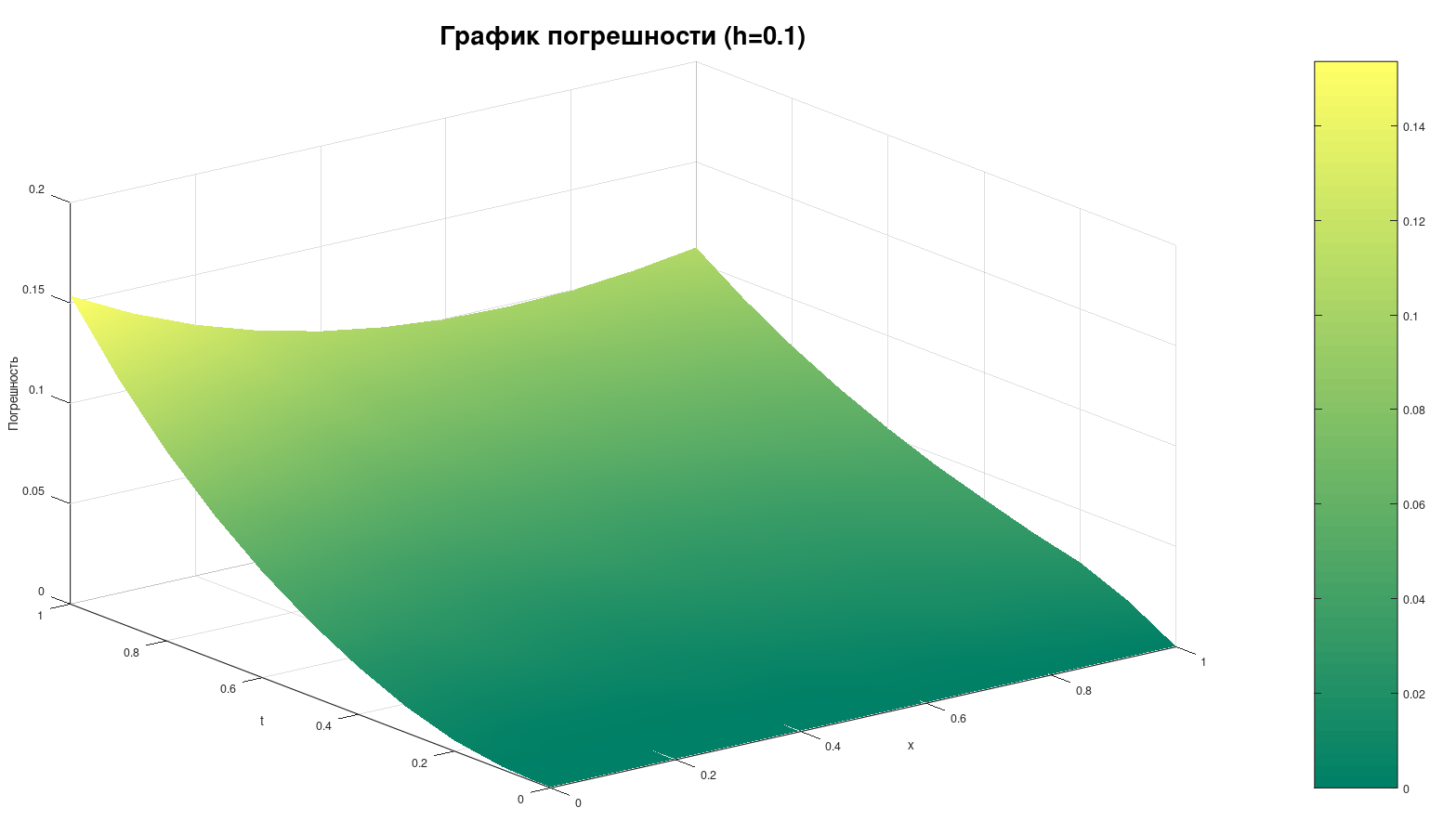


Рисунок 8 – График погрешности схемы Кранка-Николсона при шаге 0.1

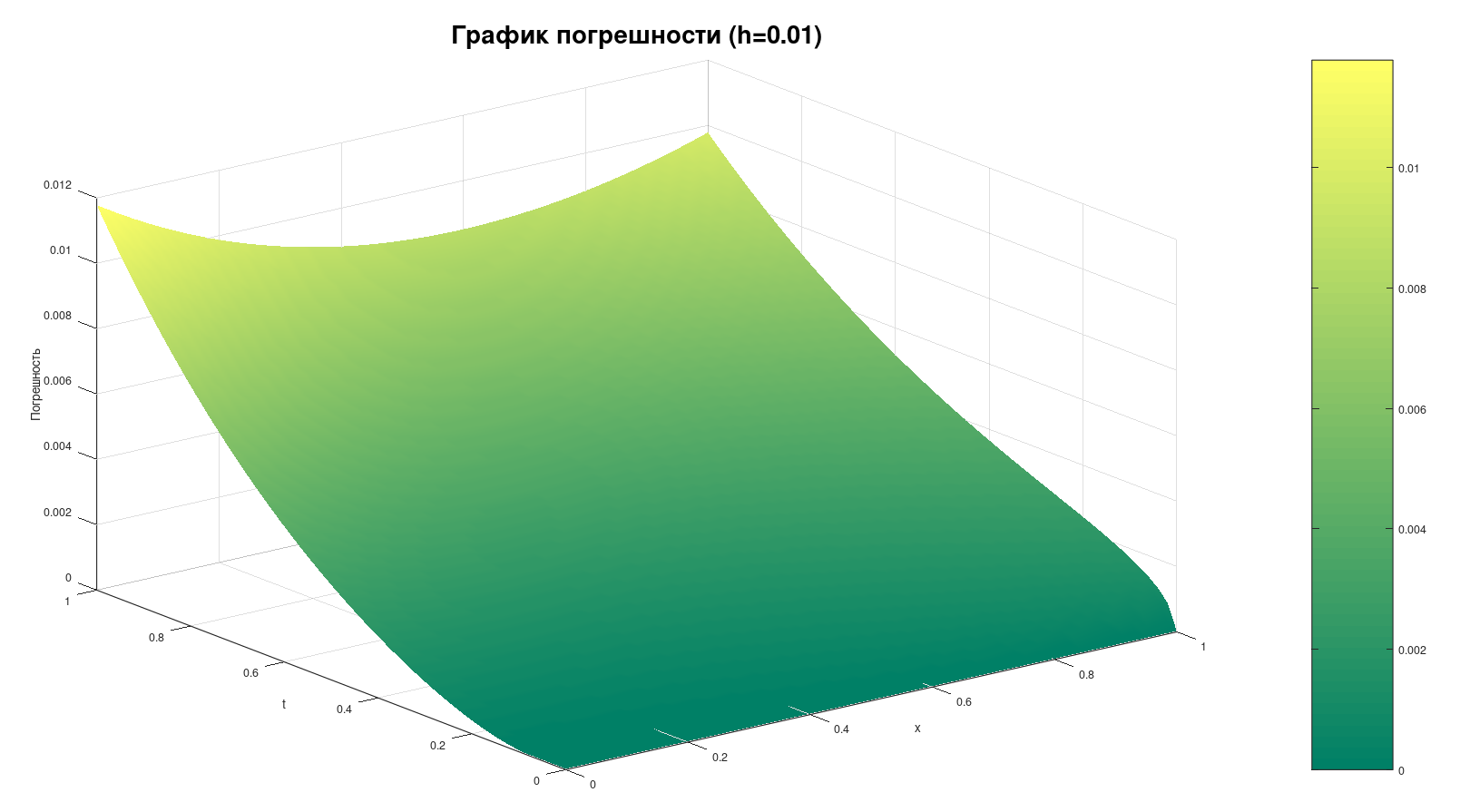


Рисунок 9 - График погрешности схемы Кранка-Николсона при шаге 0.01

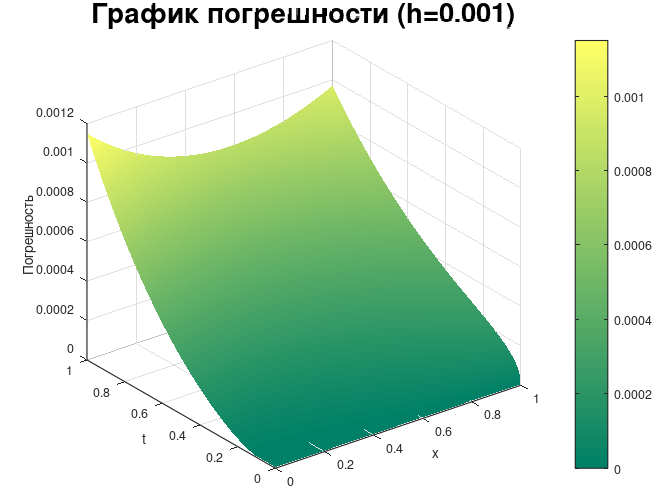


Рисунок 10 - График погрешности схемы Кранка-Николсона при шаге 0.001

По рисунку 11 можно сделать вывод, что явная разностная схема работает быстрее на всех шагах, чем остальные разностные схемы.

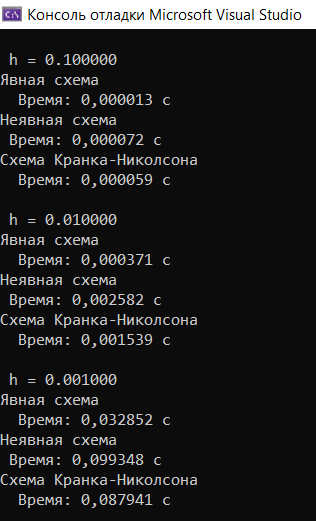


Рисунок 11 – Время выполнения программы

***Задача 3 (2 балла).***

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.

|  |
| --- |
|  |
|  |
|  |
| Консервативная конечно-разностная схема строится на основе законов сохранения, следствием которых является уравнение в частных производных. |

На рисунке 12 показано время выполнения и ошибка при различных шагах. С уменьшением шага время, необходимое на расчеты увеличивается, но ошибка уменьшается

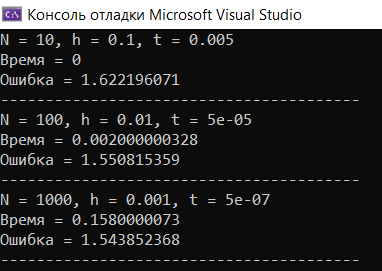
****

Рисунок 12 – результаты консервативной разностной схемы

**Вывод**

В результате выполнения 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**

#include <iostream>

#include <vector>

#include <fstream>

#include <cmath>

#include <algorithm>

#include <string>

#include <chrono>

#include <iomanip>

#include <utility>

using namespace std;

using namespace chrono;

string format\_duration(double seconds) {

if (seconds < 1e-6) {

return to\_string(seconds \* 1e9) + " нс";

}

else if (seconds < 1e-3) {

return to\_string(seconds \* 1e6) + " мкс";

}

else if (seconds < 1) {

return to\_string(seconds \* 1e3) + " мс";

}

return to\_string(seconds) + " с";

}

vector<double> Progonka(const vector<double>& A, const vector<double>& B,

const vector<double>& C, const vector<double>& D) {

int n = A.size();

vector<double> P(n), Q(n), X(n);

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

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

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

double denominator = B[i] - A[i] \* P[i - 1];

P[i] = (i < n - 1) ? C[i] / denominator : 0;

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

}

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;

}

// Аналитическое решение

double AnalyticalSolution(double x, double t) {

return exp(t) \* sinh(x);

}

// Явная схема

pair<vector<vector<double>>, double> ExplicitScheme(double hx, double ht, int nX, int nT, double& time) {

vector<vector<double>> u(nT, vector<double>(nX, 0));

auto start = high\_resolution\_clock::now();

for (int k = 0; k < nX; k++) {

u[0][k] = sinh(k \* hx);

}

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

for (int j = 1; j < nT; j++) {

double t = j \* ht;

u[j][0] = (hx \* exp(t) - u[j - 1][1]) / (hx - 1);

for (int k = 1; k < nX - 1; k++) {

u[j][k] = gamma \* u[j - 1][k - 1] + (1 - 2 \* gamma) \* u[j - 1][k] + gamma \* u[j - 1][k + 1];

}

u[j][nX - 1] = (hx \* exp(t + 1) + u[j][nX - 2]) / (hx + 1);

}

auto stop = high\_resolution\_clock::now();

time = duration\_cast<microseconds>(stop - start).count() / 1e6;

double max\_error = 0;

for (int j = 0; j < nT; j++) {

for (int k = 0; k < nX; k++) {

double exact = AnalyticalSolution(k \* hx, j \* ht);

max\_error = max(max\_error, abs(u[j][k] - exact));

}

}

return make\_pair(u, max\_error);

}

// Неявная схема

pair<vector<vector<double>>, double> ImplicitScheme(double hx, double ht, int nX, int nT, double& time) {

vector<vector<double>> u(nT, vector<double>(nX, 0));

auto start = high\_resolution\_clock::now();

for (int k = 0; k < nX; k++) {

u[0][k] = sinh(k \* hx);

}

for (int j = 1; j < nT; j++) {

double t = j \* ht;

vector<double> A(nX), B(nX), C(nX), D(nX);

A[0] = 0; B[0] = hx - 1; C[0] = 1; D[0] = hx \* exp(t);

for (int k = 1; k < nX - 1; k++) {

A[k] = -1.0 / (hx \* hx);

B[k] = 1.0 / ht + 2.0 / (hx \* hx);

C[k] = -1.0 / (hx \* hx);

D[k] = u[j - 1][k] / ht;

}

A[nX - 1] = -1; B[nX - 1] = hx + 1; C[nX - 1] = 0; D[nX - 1] = hx \* exp(t + 1);

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

}

auto stop = high\_resolution\_clock::now();

time = duration\_cast<microseconds>(stop - start).count() / 1e6;

double max\_error = 0;

for (int j = 0; j < nT; j++) {

for (int k = 0; k < nX; k++) {

double exact = AnalyticalSolution(k \* hx, j \* ht);

max\_error = max(max\_error, abs(u[j][k] - exact));

}

}

return make\_pair(u, max\_error);

}

pair<vector<vector<double>>, double> CrankNicolsonScheme(double hx, double ht, int nX, int nT, double& time) {

vector<vector<double>> u(nT, vector<double>(nX, 0));

auto start = high\_resolution\_clock::now();

for (int k = 0; k < nX; k++) {

u[0][k] = sinh(k \* hx);

}

for (int j = 1; j < nT; j++) {

double t = j \* ht;

vector<double> A(nX), B(nX), C(nX), D(nX);

A[0] = 0; B[0] = hx - 1; C[0] = 1; D[0] = hx \* exp(t);

for (int k = 1; k < nX - 1; k++) {

A[k] = -0.5 / (hx \* hx);

B[k] = 1.0 / ht + 1.0 / (hx \* hx);

C[k] = -0.5 / (hx \* hx);

D[k] = 0.5 / (hx \* hx) \* (u[j - 1][k - 1] + u[j - 1][k + 1]) +

(1.0 / ht - 1.0 / (hx \* hx)) \* u[j - 1][k];

}

A[nX - 1] = -1; B[nX - 1] = hx + 1; C[nX - 1] = 0; D[nX - 1] = hx \* exp(t + 1);

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

}

auto stop = high\_resolution\_clock::now();

time = duration\_cast<microseconds>(stop - start).count() / 1e6;

double max\_error = 0;

for (int j = 0; j < nT; j++) {

for (int k = 0; k < nX; k++) {

double exact = AnalyticalSolution(k \* hx, j \* ht);

max\_error = max(max\_error, abs(u[j][k] - exact));

}

}

return make\_pair(u, max\_error);

}

void WriteAbsoluteDifferenceToFile(const vector<vector<double>>& numerical,

double hx, double ht,

const string& filename) {

ofstream file(filename);

if (!file.is\_open()) {

cerr << "Error opening file: " << filename << endl;

return;

}

int nT = numerical.size();

int nX = numerical[0].size();

for (int j = 0; j < nT; ++j) {

double t = j \* ht;

for (int k = 0; k < nX; ++k) {

double x = k \* hx;

double exact = AnalyticalSolution(x, t);

double abs\_diff = abs(exact - numerical[j][k]);

file << x << " " << t << " " << abs\_diff << endl;

}

file << endl;

}

}

int main() {

vector<double> hx\_values = { 0.1, 0.01, 0.001 };

for (double hx : hx\_values) {

double ht = hx;

int nX = static\_cast<int>(1.0 / hx) + 1;

int nT = static\_cast<int>(1.0 / ht) + 1;

double time;

auto cn\_result = CrankNicolsonScheme(hx, ht, nX, nT, time);

string filename = "absolute\_difference\_hx\_" + to\_string(hx) + ".dat";

WriteAbsoluteDifferenceToFile(cn\_result.first, hx, ht, filename);

cout << "Absolute difference data written to " << filename << endl;

}

return 0;

setlocale(LC\_ALL, "RUS");

vector<double> hx\_values = { 0.1, 0.01, 0.001 };

vector<double> explicit\_errors, implicit\_errors, cn\_errors;

vector<double> explicit\_times, implicit\_times, cn\_times;//

cout << fixed << setprecision(6);

cout << " " << setw(15) << " x" << setw(15) << "t"

<< setw(20) << "Ошибка" << setw(20) << endl;

for (double hx : hx\_values) {

int nX = static\_cast<int>(1.0 / hx) + 1;

double ht = hx;

int nT = static\_cast<int>(1.0 / ht) + 1;

double time;

pair<vector<vector<double>>, double> explicit\_result = ExplicitScheme(hx, ht, nX, nT, time);

explicit\_errors.push\_back(explicit\_result.second);

explicit\_times.push\_back(time);

cout << "Явная схема"

<< setw(15) << fixed << setprecision(3) << hx

<< setw(15) << fixed << setprecision(3) << ht

<< setw(20) << scientific << setprecision(3) << explicit\_result.second

<< setw(20) << endl;

pair<vector<vector<double>>, double> implicit\_result = ImplicitScheme(hx, ht, nX, nT, time);

implicit\_errors.push\_back(implicit\_result.second);

implicit\_times.push\_back(time);

cout << "Неявная схема" << setw(15) << hx << setw(15) << ht

<< setw(20) << implicit\_result.second << setw(20) << endl;

pair<vector<vector<double>>, double> cn\_result = CrankNicolsonScheme(hx, ht, nX, nT, time);

cn\_errors.push\_back(cn\_result.second);

cn\_times.push\_back(time);

cout << "Явная схема" << setw(12) << hx << setw(15) << ht

<< setw(20) << cn\_result.second << setw(20) << endl;

}

return 0;

}

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <stdio.h>

#include <fstream>

#include <math.h>

#include <iomanip>

#include <vector>

using namespace std;

#define eps 0.01

double fi(double x, double t) {

return sinh(x);

}

double F(double x, double t) {

return 0.;

}

double psiOne(double x, double t) {

return exp(t);

}

double psiTwo(double x, double t) {

return exp(t + 1);

}

double K(double x, double t) {

return 1.;

}

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

vector<double> alpha(N + 1), beta(N + 1);

// Прямой ход

alpha[1] = -C[0] / B[0];

beta[1] = f[0] / B[0];

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

double denominator = B[i] + A[i] \* alpha[i];

alpha[i + 1] = -C[i] / denominator;

beta[i + 1] = (f[i] - A[i] \* beta[i]) / denominator;

}

// Обратный ход

y[N] = (f[N] - A[N] \* beta[N]) / (B[N] + A[N] \* alpha[N]);

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

y[i] = alpha[i + 1] \* y[i + 1] + beta[i + 1];

}

}

void CS(int N, double h, double tau) {

vector<double> A(N + 1), B(N + 1), C(N + 1), f(N + 1);

vector<double> x(N + 1), uPrev(N + 1), uNext(N + 1), t(N + 1);

double coef = 1.;

double ke, kw;

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

x[i] = i \* h;

t[i] = i \* tau;

uPrev[i] = sinh(x[i]);

uNext[i] = 0;

}

double pogr = 0;

float timeStart = clock() / (float)CLOCKS\_PER\_SEC;

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

A[0] = 0.;

B[0] = 1.;

C[0] = -1.;

f[0] = -h \* exp(t[tt + 1]);

A[N] = 0.;

B[N] = 1.;

C[N] = 0.;

f[N] = exp(t[tt + 1] + 1);

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

kw = (uPrev[j] + uPrev[j - 1]) / 2.;

ke = (uPrev[j + 1] + uPrev[j]) / 2.;

A[j] = (-coef / (h \* h)) \* kw;

B[j] = (1. / tau + coef \* (ke + kw) / (h \* h));

C[j] = (-coef / (h \* h)) \* ke;

f[j] = uPrev[j] / tau + (1 - coef) \* (kw \* uPrev[j - 1] - (ke + kw) \* uPrev[j] + ke \* uPrev[j + 1]) / (h \* h);

}

progonka(N, A.data(), B.data(), C.data(), f.data(), uNext.data());

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

uPrev[i] = uNext[i];

double analitik = exp(t[tt + 1]) \* sinh(x[i]);

double delta = abs(analitik - uNext[i]);

if (delta > pogr) {

pogr = delta;

}

}

}

float timeStop = clock() / (float)CLOCKS\_PER\_SEC;

cout << "Время = " << timeStop - timeStart << endl;

cout << "Ошибка = " << setprecision(10) << pogr << endl;

}

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

setlocale(LC\_ALL, "RUS");

double h, tau;

double a = 0., b = 1.;

int N\_values[] = { 10, 100, 1000 };

int num\_cases = sizeof(N\_values) / sizeof(N\_values[0]);

for (int i = 0; i < num\_cases; i++) {

int N = N\_values[i];

h = (b - a) / N;

tau = h \* h / 2; // Условие устойчивости

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

CS(N, h, tau);

cout << "----------------------------------------" << endl;

}

return 0;

}