Московский авиационный институт

(Национальный исследовательский университет)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра «Вычислительная математика и программирование»

Лабораторные работы

по курсу «Численные методы»

Вариант 6

Выполнил: Наседкин Г.К.

Группа: М8О-405Б-20

Проверила: доц. Иванов И. Э.

Дата:

Оценка:

Москва, 2023

Оглавление

Лабораторная работа 5

Задание

Используя явную и неявную конечно-разностные схемы, а также схему Кранка - Николсона, решить начально-краевую задачу для дифференциального уравнения параболического типа. Осуществить реализацию трех вариантов аппроксимации граничных условий, содержащих производные: двухточечная аппроксимация с первым порядком, трехточечная аппроксимация со вторым порядком, двухточечная аппроксимация со вторым порядком. В различные моменты времени вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением $U(\mathrm{x,t})$. Исследовать зависимость погрешности от сеточных параметров t,h.

$$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2}+\cos x(\cos \mathrm{t}+\sin t),\ u(0,\mathrm{t})=\sin t,\ u_x\left(\frac{\pi}{2},\mathrm{t}\right)=-\sin t,\ u(x,0)=0.$$

Аналитическое решение: $U(\mathrm{x,t})=\sin t\cos x$.

Теоретические сведения

Постановка задачи

$$\begin{cases} \dfrac{\partial u}{\partial t}=a^2\dfrac{\partial^2 u}{\partial x^2}+f\left(x_i,t^k\right),0<x<l,t>0 \\ \alpha u_x(0,t)+\beta u(0,t)=\varphi_0(t),x=0,t>0 \\ \gamma u_x(l,t)+\delta u(0,t)=\varphi_l(t),x=l,t>0 \\ u(x,0)=\psi(x),0\le x\le l,t=0 \end{cases}$$

Нанесем на пространственно-временную область $0\le x\le l, 0\le t\le T$ конечноразностную сетку $\omega_{h\tau}=\{x_j=jh,\ j=\overline{0,N};t^k=k\tau,k=\overline{0,K}\}$ с пространственным шагом $h=\dfrac{l}{N}$ и шагом по времени $\tau=\dfrac{T}{K}$.

Аппроксимируем дифференциальные операторы отношением конечных разностей: $\left.\dfrac{\partial u}{\partial t}\right|_j^k\approx\dfrac{u_j^{k+1}-u_j^k}{\tau}+O(\tau)$, $\left.\dfrac{\partial^2 u}{\partial x^2}\right|_j^k\approx\dfrac{u_{j+1}^k-2u_j^k+u_{j-1}^k}{h^2}+O\left(h^2\right)$

Подставляем в задачу и получаем *явную конечно-разностную схему* для этой задачи в форме

$$\frac{u_j^{k+1}-u_j^k}{\tau}=a^2\frac{u_{j+1}^k-2u_j^k+u_{j-1}^k}{h^2}+f\left(x_j,t^k\right)+O\left(\tau+h^2\right),\ j=\overline{1,N-1},k=\overline{0,K-1}$$

$u_0^k = \varphi_0\left(t^k\right), u_N^k = u_{N-1}^k + h\varphi_1\left(t^k\right), k = \overline{0, K}; u_j^0 = \psi\left(x_j\right), j = \overline{0, N}$

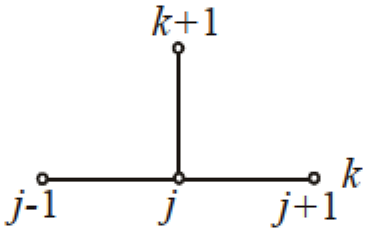
Явная схема является условно устойчивой, с условием $\sigma = a\frac{\tau}{h^2} \leq \frac{1}{2}$

Если в дифференциальный оператор по пространственной переменной аппроксимировать отношением конечных разностей на верхнем временном слое, то после подстановки в задачу получим *неявную конечно-разностную схему* для этой задачи

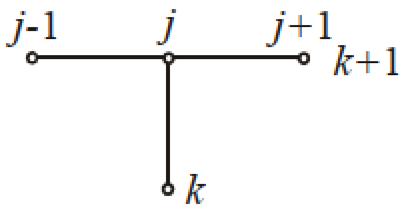
$$\frac{u_j^{k+1} - u_j^k}{\tau} = a^2\frac{u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1}}{h^2} + f\left(x_j, t^{k+1}\right) + O\left(\tau + h^2\right), j = \overline{1, N-1}, k = \overline{0, K-1}$$

$$u_0^{k+1} = \varphi_0\left(t^{k+1}\right), u_N^{k+1} = u_{N-1}^{k+1} + h\varphi_1\left(t^{k+1}\right), k = \overline{0, K}; u_j^0 = \psi\left(x_j\right), j = \overline{0, N},$$

Теперь сеточную функцию на верхнем временном слое можно получить из решения СЛАУ с трехдиагональной матрицей.



шаблон явной схемы



шаблон неявной схемы

Рассмотрим неявно-явную схему с весами для нашей задачи

$$\frac{u_j^{k+1} - u_j^k}{\tau} = \theta\left(a^2\frac{u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1}}{h^2} + f\left(x_j, t^{k+1}\right)\right) + \left(1-\theta\right)\left(a^2\frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{h^2} + f\left(x_j, t^k\right)\right)$$

где θ – вес неявной части конечно-разностной схемы, 1-θ вес для явной части, причем $0 \leq \theta \leq 1$. При θ=1 имеем полностью неявную схему, при θ=0 – полностью явную схему, и при θ=1/2 – схему *Кранка-Николсона*.

Код:

tools.h

```
#include <iostream>

#include <cmath>
```

```
#include <vector>
#include "TMA.h"
#include <iomanip>
#include "../Graphica.h"
using namespace std;

struct BorderCondition {
    double a, b;
    double (* func)(double, double);
    double operator() (double x, double y) { return func(x, y); }
};
using BC = BorderCondition;

Matrix<double> explicitMethod(size_t N, double L, double T,
                              BC leftCondition, BC rightCondition, BC initialConditions,
                              double (*f)(double, double)) {
    double h = L / N;
    size_t K = T / (0.495 * pow(h, 2.));
    double tau = T / K;
    double sigma = tau / pow(h, 2.);

    Matrix<double> u(N + 1, K + 1);
    for (size_t j = 0; j <= N; j++) u[j][0] = initialConditions(j * h, 0.);

    for (size_t k = 0; k < K; k++) {
        for (size_t j = 1; j < N; j++)
            u[j][k + 1] = u[j][k] + sigma * (u[j + 1][k] - 2 * u[j][k] + u[j - 1][k]) + tau *
f(j * h, k * tau);
        u[0][k + 1] = (leftCondition(0, (k + 1) * tau) - u[1][k + 1] * leftCondition.a / h) /
(leftCondition.b - leftCondition.a / h);
        u[N][k + 1] = (rightCondition(0, (k + 1) * tau) + u[N - 1][k + 1] * rightCondition.a /
h) / (rightCondition.b + rightCondition.a / h);
    }
    return u;
}

Matrix<double> implicitMethod(size_t N, double L, double T,
                              BC leftCondition, BC rightCondition, BC initialConditions,
                              double (*f)(double, double)) {
```

```
    double h = L / N;
    size_t K = T / (0.495 * pow(h, 2.));
    double tau = T / K;
    double sigma = tau / pow(h, 2.);

    Matrix<double> u(N + 1, K + 1);
    for (size_t j = 0; j <= N; j++) u[j][0] = initialConditions(j * h, 0.);

    Matrix<double> m(3, N + 1), d(1, N + 1);
    m[0][0] = 0;
    m[1][0] = leftCondition.b - leftCondition.a / h;
    m[2][0] = leftCondition.a / h;
    for (size_t j = 1; j < N; j++) {
        m[0][j] = sigma;
        m[1][j] = -(1 + 2 * sigma);
        m[2][j] = sigma;
    }
    m[0][N] = -rightCondition.a / h;
    m[1][N] = rightCondition.b + rightCondition.a / h;
    m[2][N] = 0;
    for (size_t k = 0; k < K; k++) {
        for (size_t j = 1; j < N; j++)
            d[0][j] = -u[j][k] - tau * f(j * h, (k + 1) * tau);
        d[0][0] = leftCondition(0, (k + 1) * tau);
        d[0][N] = rightCondition(0, (k + 1) * tau);
        u.copy(TMAsolve(m, d).T(), 0, k + 1);
    }
    return u;
}

Matrix<double> finiteDifferenceMethod(size_t N, double L, double T,
                                      BC leftCondition, BC rightCondition, BC
initialConditions,
                                      double (*f)(double, double),
                                      double theta,
                                      int approximationType = 0) {
    double h = L / N;
    size_t K = T / (0.495 * pow(h, 2.));
```

```
    double tau = T / K;
    double sigma = tau / pow(h, 2.);

    Matrix<double> u(N + 1, K + 1);
    for (size_t j = 0; j <= N; j++) u[j][0] = initialConditions(j * h, 0.);

    Matrix<double> m(3, N + 1), d(1, N + 1);
    m[0][0] = 0;
    m[1][0] = leftCondition.b - leftCondition.a / h;
    if (approximationType == 0 || theta == 0) {
        m[2][0] = leftCondition.a / h;
    } else if (approximationType == 1) {
        m[2][0] = leftCondition.a / h - leftCondition.a * h / (2 * tau * theta);
    }
    for (size_t j = 1; j < N; j++) {
        m[0][j] = sigma * theta;
        m[1][j] = -(1 + 2 * sigma * theta);
        m[2][j] = sigma * theta;
    }
    if (approximationType == 0 || theta == 0) {
        m[0][N] = -rightCondition.a / h;
    } else if (approximationType == 1) {
        m[0][N] = -rightCondition.a / h + rightCondition.a * h / (2 * tau * theta);
    }
    m[1][N] = rightCondition.b + rightCondition.a / h;
    m[2][N] = 0;
    for (size_t k = 0; k < K; k++) {
        for (size_t j = 1; j < N; j++)
            d[0][j] = -(
                u[j][k] +
                tau * theta * f(j * h, (k + 1) * tau) +
                (1 - theta) * (sigma * (u[j + 1][k] - 2 * u[j][k] + u[j - 1][k]) + tau * f(j *
h, k * tau))
            );
        if (approximationType == 0 || theta == 0) {
            d[0][0] = leftCondition(0, (k + 1) * tau);
            d[0][N] = rightCondition(0, (k + 1) * tau);
        } else if (approximationType == 1) {
```

```
            d[0][0] = leftCondition(0, (k + 1) * tau) + d[0][1] * leftCondition.a * h / (2 * tau * theta);
            d[0][N] = rightCondition(0, (k + 1) * tau) - d[0][N - 1] * rightCondition.a * h / (2 * tau * theta);
        }
        u.copy(TMAsolve(m, d).T(), 0, k + 1);
    }
    return u;
}


void print(Matrix<double> A) {
    for (int j = 0; j < A.dim.second; j++) cout << "+-----------";
    cout << "+\n";
    for (int i = 0; i < A.dim.first; i++) {
        cout << "| ";
        for (int j = 0; j < A.dim.second; j++) cout << setw(9) << to_string(A[i][j]) << ((j != A.dim.second - 1) ? " : " : "");
        cout << " |\n";
        for (int j = 0; j < A.dim.second; j++) cout << "+-----------";
        cout << "+\n";
    }
    cout << '\n';
}
```

main.cpp

```
#include "tools.h"
#include <thread>
#include "../UI/bar.h"


double firstFunc  (double x, double t) { return          sin(t); }
double secondFunc (double x, double t) { return         -sin(t); }
double initialFunc(double x, double t) { return             0.; }
double answer     (double x, double t) { return sin(t) * cos(x); }
double f          (double x, double t) { return cos(x) * (sin(t) + cos(t)); }


int main() {
    // параболический тип
    ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
    setlocale(LC_ALL, "rus");
```

```
    thread thread1(ShowGraphics);

    loadFonts();
    Bar<int> bar;
    bar.ValueText.setFont(font);
    bar.setColors(sf::Color(0, 0, 0), sf::Color(250, 120, 0), sf::Color(120, 120, 120));
    bar.setWidth(scw);
    bar.setPosition(0, sch - STANDART_BAR_HEIGHT);
    drawableStuff.push_back(&bar);

    double L = M_PI_2, T = 0.6;
    BC leftCondition     {0., 1., firstFunc  };
    BC rightCondition    {1., 0., secondFunc };
    BC initialConditions {0., 1., initialFunc};

    cout << "какой метод использовать?\n1 - явный\n2 - неявный\n3 - смешанный\n>";
cout.flush();
    int ans; cin >> ans;
    double theta = ans == 1 ? 0. : ans == 2 ? 1. : 0.;
    cout << "с максимальным разибением N = "; cout.flush();
    size_t N; cin >> N;
    Matrix<double> res = finiteDifferenceMethod(N, L, T, leftCondition, rightCondition,
initialConditions, f, theta);
    size_t K = res.dim.second - 1;

    vector<pair<double, double>> g1(N + 1);
    for (int i = 0; i < g1.size(); i++)
        g1[i] = {(i * L) / N, res[i][K]};

    res = finiteDifferenceMethod(N, L, T, leftCondition, rightCondition, initialConditions, f,
theta, 1);
    vector<pair<double, double>> g2(N + 1);
    for (int i = 0; i < g2.size(); i++)
        g2[i] = {(i * L) / N, res[i][K]};

    vector<pair<double, double>> g3(5001);
    for (int i = 0; i < g3.size(); i++) {
        g3[i].first = (i * L) / 5000;
```

```
        g3[i].second = answer((i * L) / 5000, T);

    }

    makeGraphByPoints(g1);

    makeGraphByPoints(g2, sf::Color::Red);

    makeGraphByPoints(g3, sf::Color::Green);


    Scale<int> scale{0, (int)K, (int)K};

    bar.setValue(scale);


    thread thread2([&]{

        cout <<
"--------------------------------------------------------------------------------\n"

             << "|  N  :   K    :      h      :     tau     : sigma :    error1   :    error2
|\n"

             << "+-----+--------+-------------+-------------+-------+-------------
+-------------+\n";


        vector<pair<double, double>> g4(N - 3 + 1);

        vector<pair<double, double>> g5(N - 3 + 1);

        double error1, error2;

        Matrix<double> res;

        for (int n = 3; n <= N; n++) {

            error1 = 0; error2 = 0;

            res = finiteDifferenceMethod(n, L, T, leftCondition, rightCondition,
initialConditions, f, theta);

            size_t K = res.dim.second - 1;


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

                for (int j = 0; j <= K; j++)

                    error1 += max(abs(answer((i * L) / n, (j * T) / K) - res[i][j]) - error1,
0.);

            g4[n - 3] = {L / n, error1};


            res = finiteDifferenceMethod(n, L, T, leftCondition, rightCondition,
initialConditions, f, theta, 1);

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

                for (int j = 0; j <= K; j++)

                    error2 += max(abs(answer((i * L) / n, (j * T) / K) - res[i][j]) - error2,
0.);

            g5[n - 3] = {L / n, error2};
```

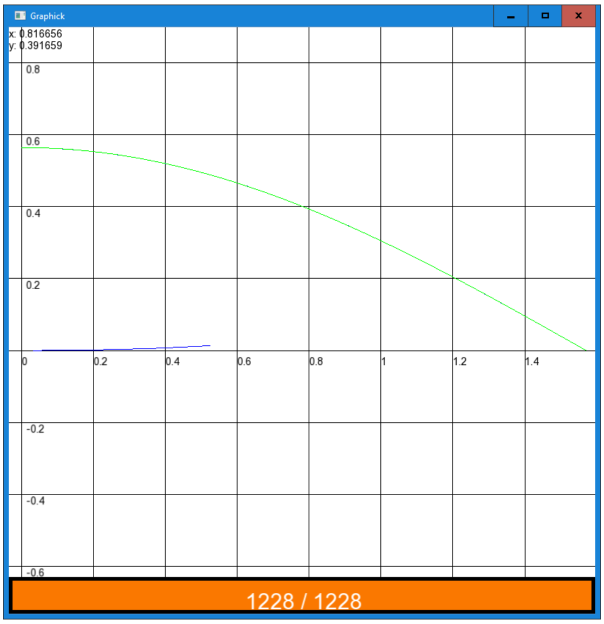
```
            cout << "| " << setw(3) << n << " : " << setw(6) << size_t(T / (0.495 * pow(L / n, 
2.)))
                 << " : " << setw(11) << L / n << " : " << setw(11) << T / (T / (0.495 * pow(L 
/ n, 2.)))
                 << " : " << setw(5) << (T / (T / (0.495 * pow(L / n, 2.)))) / pow(L / n, 2.) 
<< " : "
                 << setw(11) << error1 << " : " << setw(11) << error2 << " |\n"
                 << "+-----+--------+------------+------------+-------+------------
+------------+\n";
        }
        cout << '\n'; cout.flush();
        makeGraphByPoints(g4);
        makeGraphByPoints(g5, sf::Color::Red);
    });
    thread thread3([&]{
        while (window == nullptr || window->isOpen()) {
            if (window->hasFocus() && sf::Mouse::isButtonPressed(sf::Mouse::Left)) {
                if (sf::Mouse::getPosition(*window).y >= bar.getPosition().y) {
                    MouseBuffer = Mouse.getPosition(*window);
                    scale.cur = (double)scale.top * 
(double)sf::Mouse::getPosition(*window).x / (double)scw;
                    normalize(scale);


                    res = finiteDifferenceMethod(N, L, T, leftCondition, rightCondition, 
initialConditions, f, theta);
                    for (int i = 0; i < g1.size(); i++)
                        graph[0][i].y = -res[i][scale.cur];


                    res = finiteDifferenceMethod(N, L, T, leftCondition, rightCondition, 
initialConditions, f, theta, 1);
                    for (int i = 0; i < g1.size(); i++)
                        graph[1][i].y = -res[i][scale.cur];


                    vector<pair<double, double>> g2(5001);
                    for (int i = 0; i < g2.size(); i++) {
                        graph[2][i].y = -answer((i * L) / 5000, T * (double)scale.cur / 
(double)scale.top);
                    }
                }
            }
```

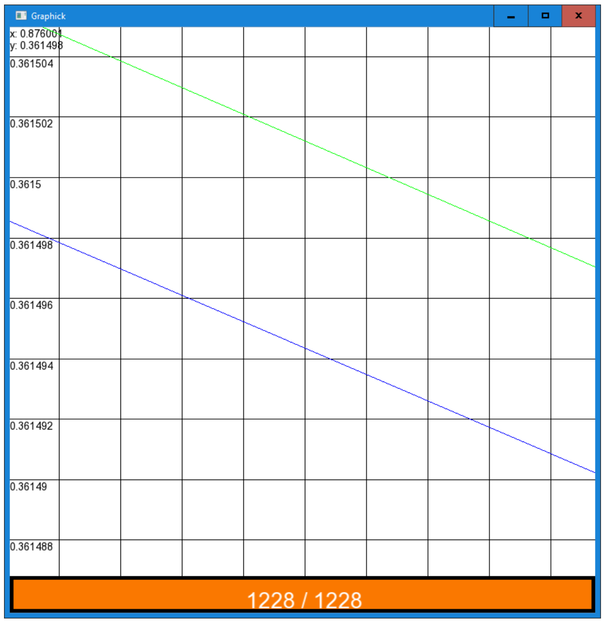
```
        }
    });
    thread1.join();
    thread2.join();
    thread3.join();
    return 0;
}
```

Вывод:

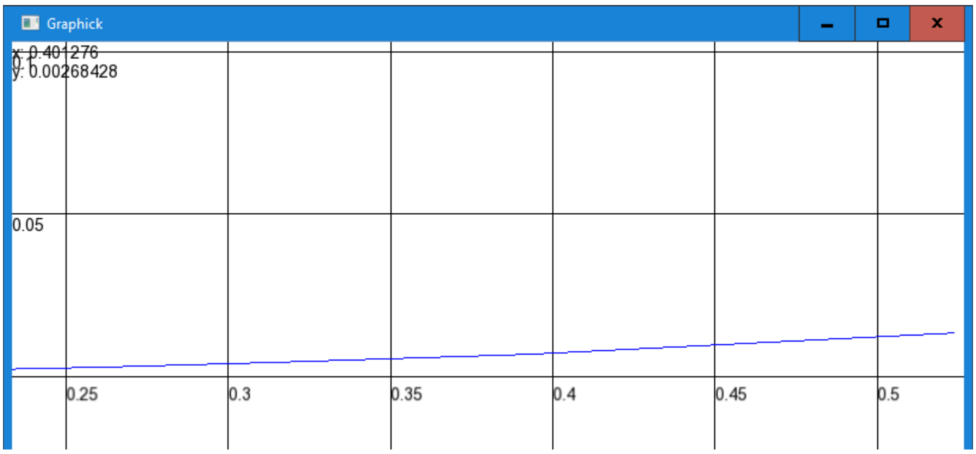
На графиках представлены численное (синий) и аналитические (зелёный) решения при разбиении N = 50

Явный метод

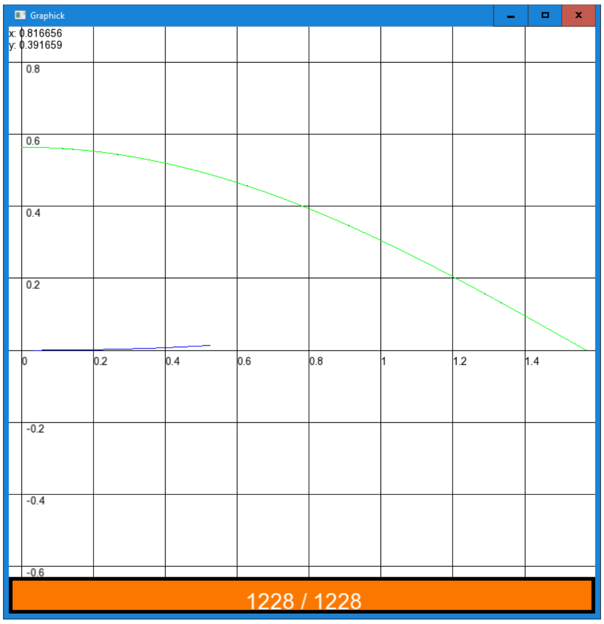


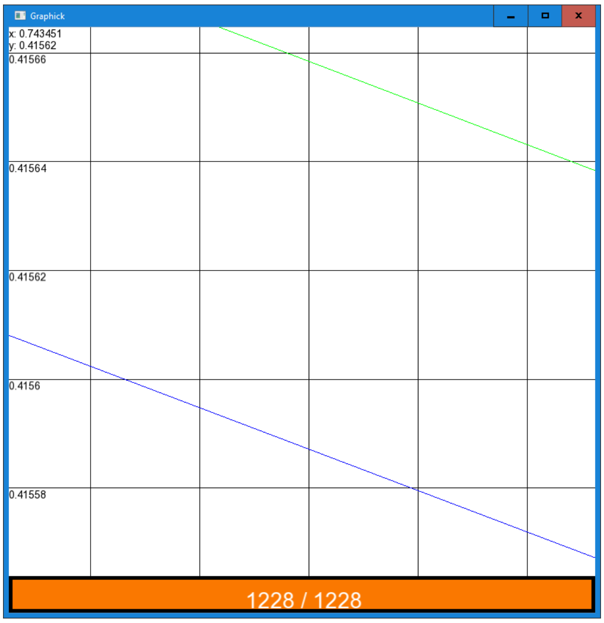


Далее приведён график, отражающий зависимость погрешности от шага разбиения

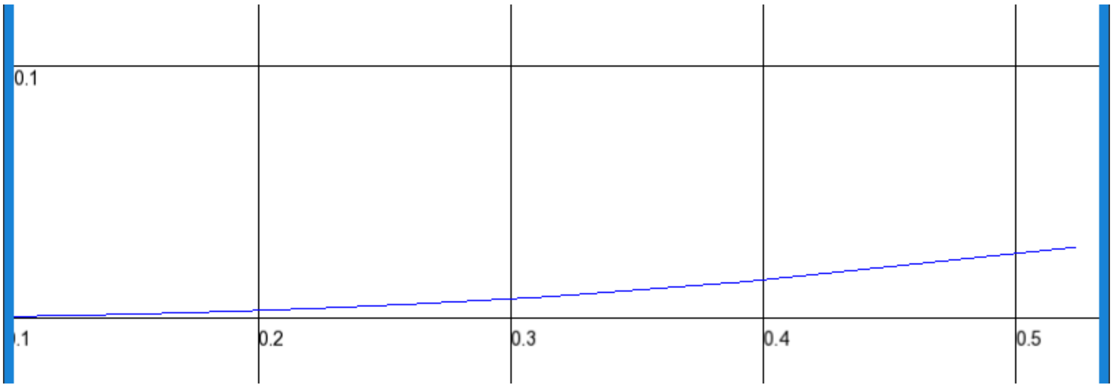


Неявный метод

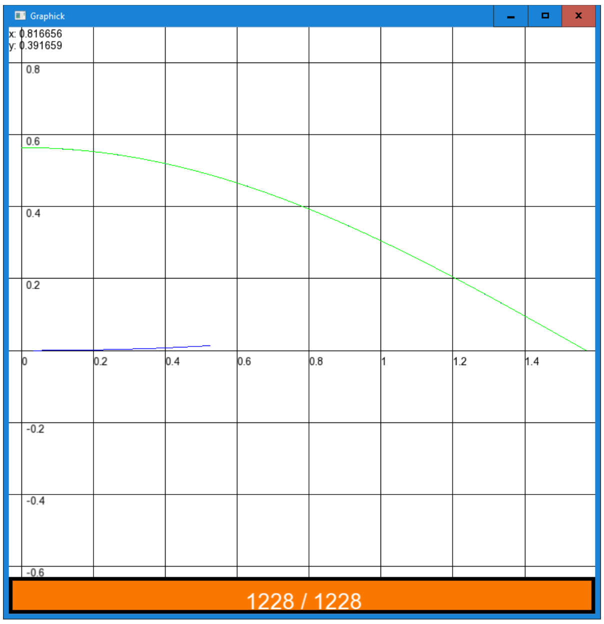


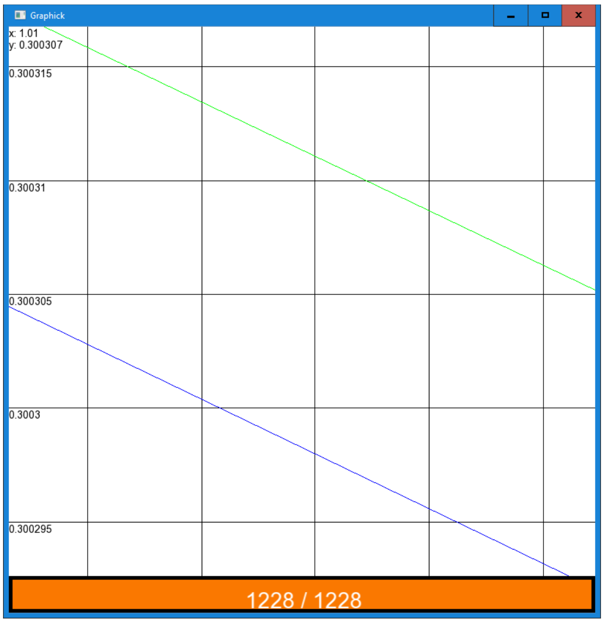


Далее приведён график, отражающий зависимость погрешности от шага разбиения

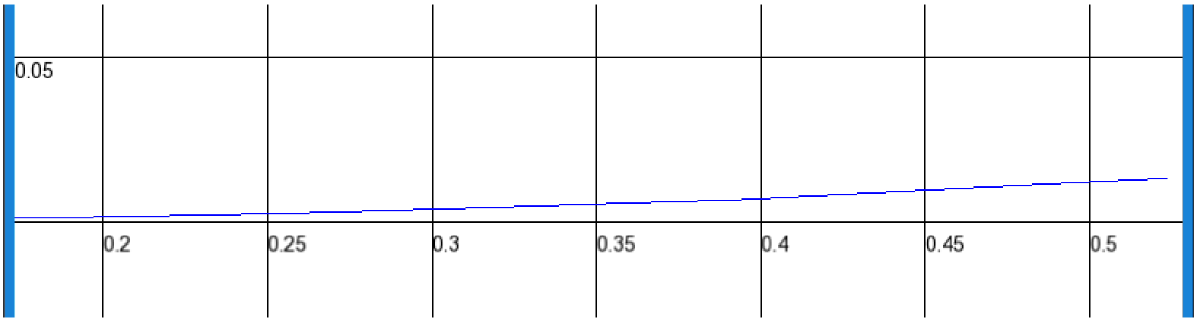


Метод Кранка-Николсона





Далее приведён график, отражающий зависимость погрешности от шага разбиения



Исходя из графика погрешности от длины шага можно сказать, что отклонение от искомого решения явным методом и Кранка-Николсона в целом меньше, чем у неявного.

Лабораторная работа 6

Задание

Используя явную схему крест и неявную схему, решить начально-краевую задачу для дифференциального уравнения гиперболического типа. Аппроксимацию второго начального условия произвести с первым и со вторым порядком. Осуществить реализацию трех вариантов аппроксимации граничных условий, содержащих производные: двухточечная аппроксимация с первым порядком, трехточечная аппроксимация со вторым порядком, двухточечная аппроксимация со вторым порядком. В различные моменты времени вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением $U(x,t)$. Исследовать зависимость погрешности от сеточных параметров t, h.

$$\frac{\partial^2 u}{\partial t^2}=\frac{\partial^2 u}{\partial x^2}+2\frac{\partial u}{\partial x}-2u,\quad \begin{matrix} u(0,t)=\cos(2t) \\ u\left(\frac{\pi}{2},t\right)=0 \end{matrix},\ u(x,0)=\exp(-x)\cos x,\ u_t(x,0)=0.$$

Аналитическое решение: $U(x,t)=\exp(-x)\cos x\cos(2t)$

Теоретические сведения

Постановка задачи

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2}=a^2\dfrac{\partial^2 u}{\partial x^2}+b\dfrac{\partial u}{\partial x}, 0<x<l, t>0; \\ u(0,t)=\varphi_0(t), x=0, t>0; \\ u(l,t)=\varphi_l(t), x=l, t>0; \\ u(x,0)=\psi_1(x), 0\le x\le l, t=0; \\ \dfrac{\partial u(x,0)}{\partial t}=\psi_2(x), 0\le x\le l, t=0, \end{cases}$$

Нанесем на пространственно-временную область $0\le x\le l, 0\le t\le T$ конечноразностную сетку $\omega_{h\tau}=\{x_j=jh,\ j=\overline{0,N};\ t^k=k\tau,\ k=\overline{0,K}\}$ с пространственным шагом $h=\frac{l}{N}$ и шагом по времени $\tau=\frac{T}{K}$.

Аппроксимируем дифференциальные операторы отношением конечных разностей.

$$\left.\frac{\partial^2 u}{\partial t^2}\right|_j^k=\frac{u_j^{k+1}-2u_j^k+u_j^{k-1}}{\tau^2}+O(\tau),\ \left.\frac{\partial u}{\partial x}\right|_j^k=\frac{u_{j+1}^k-u_{j-1}^k}{2h}+O(h^2),\ \left.\frac{\partial^2 u}{\partial x^2}\right|_j^k=\frac{u_{j+1}^k-2u_j^k+u_{j-1}^k}{h^2}+O(h^2),$$

Подставляем в задачу и получаем *явную конечно-разностную схему* для этой задачи в форме

$$\frac{u_j^{k+1}-2u_j^k+u_j^{k-1}}{\tau^2}=a^2\frac{u_{j+1}^k-2u_j^k+u_{j-1}^k}{h^2}+b\frac{u_{j+1}^k-u_{j-1}^k}{2h}+O\left(\tau+h^2\right),\ j=\overline{1,N-1},k=\overline{0,K-1}$$

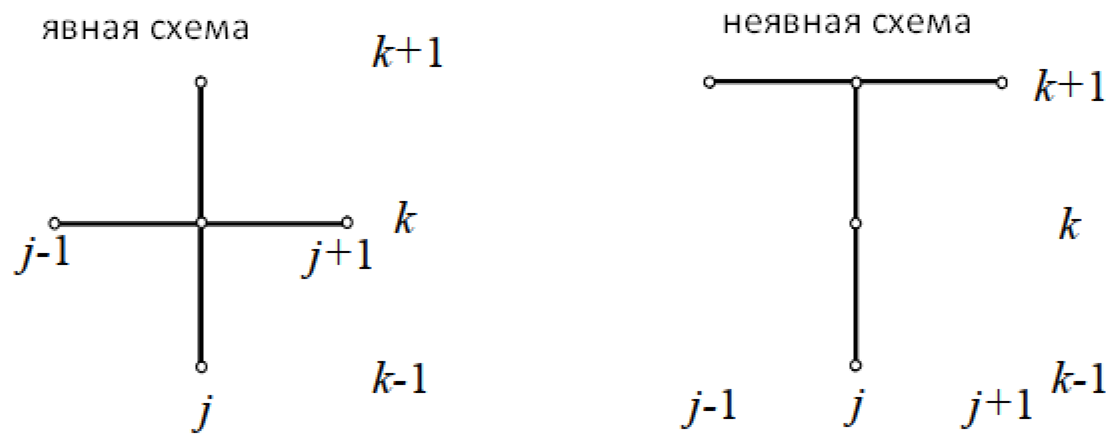
$$u_0^k=\varphi_0\left(t^k\right),u_N^k=\varphi_1\left(t^k\right),k=\overline{0,K};$$

Если в дифференциальный оператор по пространственной переменной аппроксимировать отношением конечных разностей на верхнем временном слое, то после подстановки в задачу получим *неявную конечно-разностную схему* для этой задачи

$$\frac{u_j^{k+1}-2u_j^k+u_j^{k-1}}{\tau^2}=a^2\frac{u_{j+1}^{k+1}-2u_j^{k+1}+u_{j-1}^{k+1}}{h^2}+b\frac{u_{j+1}^{k+1}-u_{j-1}^{k+1}}{2h}+O\left(\tau+h^2\right),\ j=\overline{1,N-1},k=\overline{0,K-1}$$

$$u_0^{k+1}=\varphi_0\left(t^{k+1}\right),u_N^{k+1}=\varphi_1\left(t^{k+1}\right),k=\overline{0,K};$$

Теперь сеточную функцию на верхнем временном слое можно получить из решения СЛАУ с трехдиагональной матрицей.



В обеих схемах необходимо знать значения $u_j^{k-1},u_j^k,j=\overline{1,N-1},k=1,2,\ldots$на нижних временных слоях. Для *k=1* это делается следующим образом:

1. Аппроксимация с точностью $O(\tau)$

$$u_j^1=\psi_1\left(x_j\right)+\psi_2\left(x_j\right)\tau$$

2. Аппроксимация с точностью $O\left(\tau^3\right)$

$$u_j^1 = \psi_1(x_j) + \psi_2(x_j)\tau + \left(a^2 \psi_1''(x) + b\psi_1'(x) + c\psi_1(x) + f(x_j, \tau)\right)\frac{\tau^2}{2}$$

Код:

tools.h

```
#include <iostream>
#include <cmath>
#include <vector>
#include "TMA.h"
#include <iomanip>
#include "../Graphica.h"
using namespace std;

double dxt0Func  (double x, double t) { return -exp(-x) * (cos(x) + sin(x)) ; }
double ddxt0Func (double x, double t) { return 2 * exp(-x) * sin(x)         ; }

double a, b, c;
void setABC(double x, double y, double z) { a = x; b = y; c = z; }

struct BorderCondition {
    double a, b;
    double (* func)(double, double);
    double operator() (double x, double y) { return func(x, y); }
};
using BC = BorderCondition;

Matrix<double> explicitMethod(size_t N, double L, double T,
                              BC leftCondition, BC rightCondition, BC t0Condition, BC
dt0Condition,
                              double (*f)(double, double)) {
    double h = L / N;
    double h2 = pow(h, 2.);
    size_t K = T / (0.495 * h2);
    double tau = T / K;
    double tau2 = pow(tau, 2.);
    double sigma = tau2 / h2;

    double coef1 = a * a * sigma + b * sigma * h / 2;
```

```
    double coef2 = 2 + c * tau2 - 2 * a * a * sigma;
    double coef3 = a * a * sigma - b * sigma * h / 2;

    double coef6 = -leftCondition.a / h;
    double coef7 = rightCondition.a / h;

    double coef4 = leftCondition.b + coef6;
    double coef5 = rightCondition.b + coef7;

    Matrix<double> u(N + 1, K + 1);
    for (size_t j = 0; j <= N; j++) u[j][0] = t0Condition(j * h, 0.);
    for (size_t j = 0; j <= N; j++)
        u[j][1] = t0Condition(j * h, 0.)
                + tau * dt0Condition(j * h, 0.)
                + (tau2 / 2) * (a * a * ddxt0Func(j * h, 0.)
                              + b * dxt0Func(j * h, 0.)
                              + c * t0Condition(j * h, 0.)
                              + f(j * h, tau))
                ;

    for (size_t k = 1; k < K; k++) {
        for (size_t j = 1; j < N; j++)
            u[j][k + 1] = u[j + 1][k] * coef1
                        + u[j][k]     * coef2
                        + u[j - 1][k] * coef3
                        - u[j][k - 1]
                        + tau2 * f(j * h, k * tau);
        u[0][k + 1] = (leftCondition(0, (k + 1) * tau) + u[1][k + 1] * coef6) / coef4;
        u[N][k + 1] = (rightCondition(0, (k + 1) * tau) + u[N - 1][k + 1] * coef7) / coef5;
    }
    return u;
}

Matrix<double> implicitMethod(size_t N, double L, double T,
                              BC leftCondition, BC rightCondition, BC t0Condition, BC
dt0Condition,
                              double (*f)(double, double)) {
    double h = L / N;
```

```
double h2 = pow(h, 2.);
size_t K = T / (0.495 * h2);
double tau = T / K;
double tau2 = pow(tau, 2.);
double sigma = tau2 / h2;

double coefA = -a * a * sigma + b * sigma * h / 2;
double coefB = 1 + 2 * a * a * sigma - c * tau2;
double coefC = -a * a * sigma - b * sigma * h / 2;

Matrix<double> u(N + 1, K + 1);
for (size_t j = 0; j <= N; j++) u[j][0] = t0Condition(j * h, 0.);
for (size_t j = 0; j <= N; j++)
    u[j][1] = t0Condition(j * h, 0.)
            + tau * dt0Condition(j * h, 0.)
            + (tau2 / 2) * (a * a * ddxt0Func(j * h, 0.)
                          + b * dxt0Func(j * h, 0.)
                          + c * t0Condition(j * h, 0.)
                          + f(j * h, tau))
            ;

Matrix<double> m(3, N + 1), d(1, N + 1);
m[0][0] = 0;
m[1][0] = leftCondition.b - leftCondition.a / h;
m[2][0] = leftCondition.a / h;
for (size_t j = 1; j < N; j++) {
    m[0][j] = coefA;
    m[1][j] = coefB;
    m[2][j] = coefC;
}
m[0][N] = -rightCondition.a / h;
m[1][N] = rightCondition.b + rightCondition.a / h;
m[2][N] = 0;
for (size_t k = 1; k < K; k++) {
    for (size_t j = 1; j < N; j++)
        d[0][j] = 2 * u[j][k] - u[j][k - 1] + tau2 * f(j * h, (k + 1) * tau);
    d[0][0] = leftCondition(0, (k + 1) * tau);
    d[0][N] = rightCondition(0, (k + 1) * tau);
```

```
        u.copy(TMAsolve(m, d).T(), 0, k + 1);
    }
    return u;
}
```

main.cpp

```
#include "tools.h"
#include <thread>
#include "../UI/bar.h"

double f         (double x, double t) { return 0.                           ; }

double leftFunc  (double x, double t) { return cos(2 * t)                   ; }
double rightFunc (double x, double t) { return 0.                           ; }

double t0Func    (double x, double t) { return exp(-x) * cos(x)             ; }
double dt0Func   (double x, double t) { return 0.                           ; }

double answer    (double x, double t) { return exp(-x) * cos(x) * cos(2 * t); }

int main() {
    // гипербалический тип
    ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
    setlocale(LC_ALL, "rus");

    setABC(1, 2, -2);

    loadFonts();
    Bar<int> bar;
    bar.ValueText.setFont(font);
    bar.setColors(sf::Color(0, 0, 0), sf::Color(250, 120, 0), sf::Color(120, 120, 120));
    bar.setWidth(scw);
    bar.setPosition(0, sch - STANDART_BAR_HEIGHT);
    drawableStuff.push_back(&bar);
    thread thread1(ShowGraphics);

    double L = M_PI_2, T = 0.4;
    BC leftCondition  {0., 1., leftFunc };
```

```
    BC rightCondition {0., 1., rightFunc};
    BC t0Condition    {0., 1., t0Func   };
    BC dt0Condition   {1., 0., dt0Func  };

    cout << "Какой метод использовать?\n1 - явный\n2 - неявный\n>"; cout.flush();
    int ans; cin >> ans;
    auto resFunc = ans == 1 ? explicitMethod : implicitMethod;
    cout << "с максимальным разбиением N = "; cout.flush();
    size_t N; cin >> N;
    Matrix<double> res = resFunc(N, L, T, leftCondition, rightCondition, t0Condition,
dt0Condition, f);
    size_t K = res.dim.second - 1;

    vector<pair<double, double>> g1(N + 1);
    for (int i = 0; i < g1.size(); i++)
        g1[i] = {(i * L) / N, res[i][K]};

    vector<pair<double, double>> g2(5001);
    for (int i = 0; i < g2.size(); i++) {
        g2[i].first = (i * L) / 5000;
        g2[i].second = answer((i * L) / 5000, T);
    }
    makeGraphByPoints(g1);
    makeGraphByPoints(g2, sf::Color::Green);

    Scale<int> scale{0, (int)K, (int)K};
    bar.setValue(scale);

    thread thread2([&]{
        cout << "-----------------------------------------------------------------\n"
             << "|  N  :   K    :      h      :     tau     : sigma :    error1   |\n"
             << "+-----+--------+------------+------------+-------+------------+\n";

        vector<pair<double, double>> g3(N - 3 + 1);
        double error1;
        Matrix<double> res;
        for (int n = 3; n <= N; n++) {
            error1 = 0;
```

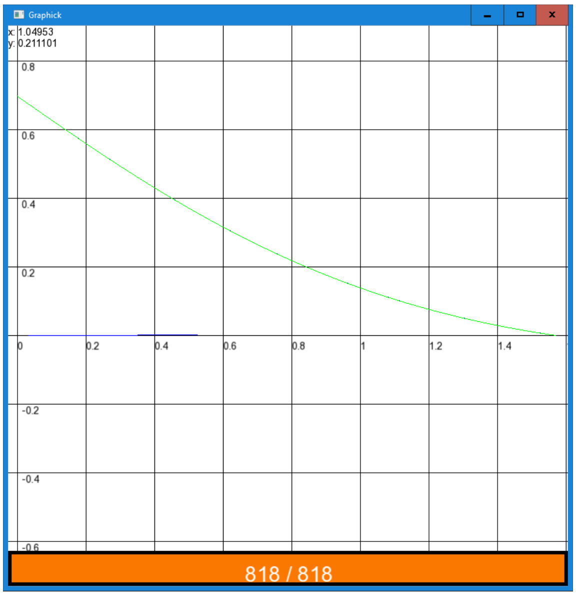
```
            res = resFunc(n, L, T, leftCondition, rightCondition, t0Condition, dt0Condition,
f);
            size_t K = res.dim.second - 1;


            for (int i = 0; i <= n; i++)
                for (int j = 0; j <= K; j++)
                    error1 += max(abs(answer((i * L) / n, (j * T) / K) - res[i][j]) - error1,
0.);
            g3[n - 3] = {L / n, error1};


            cout << "| " << setw(3) << n << " : " << setw(6) << size_t(T / (0.495 * pow(L / n,
2.)))
                 << " : " << setw(11) << L / n << " : " << setw(11) << T / (T / (0.495 * pow(L
/ n, 2.)))
                 << " : " << setw(5) << (T / (T / (0.495 * pow(L / n, 2.)))) / pow(L / n, 2.)
<< " : "
                 << setw(11) << error1 << " |\n"
                 << "+-----+--------+------------+------------+-------+-------------+\n";
        }
        cout << '\n'; cout.flush();
        makeGraphByPoints(g3);
    });
    thread thread3([&]{
        while (window == nullptr || window->isOpen()) {
            if (window->hasFocus() && sf::Mouse::isButtonPressed(sf::Mouse::Left)) {
                if (sf::Mouse::getPosition(*window).y >= bar.getPosition().y) {
                    MouseBuffer = Mouse.getPosition(*window);
                    scale.cur = (double)scale.top *
(double)sf::Mouse::getPosition(*window).x / (double)scw;
                    normalize(scale);


                    for (int i = 0; i < g1.size(); i++)
                        graph[0][i].y = -res[i][scale.cur];


                    vector<pair<double, double>> g2(5001);
                    for (int i = 0; i < g2.size(); i++) {
                        graph[1][i].y = -answer((i * L) / 5000, T * (double)scale.cur /
(double)scale.top);
                    }
                }
```

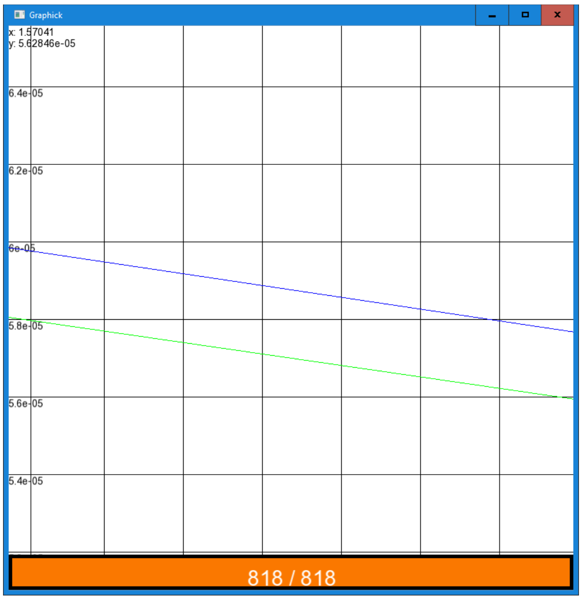
```
            }
        }
    });
    thread1.join();
    thread2.join();
    thread3.join();
    return 0;
}
```

Вывод:

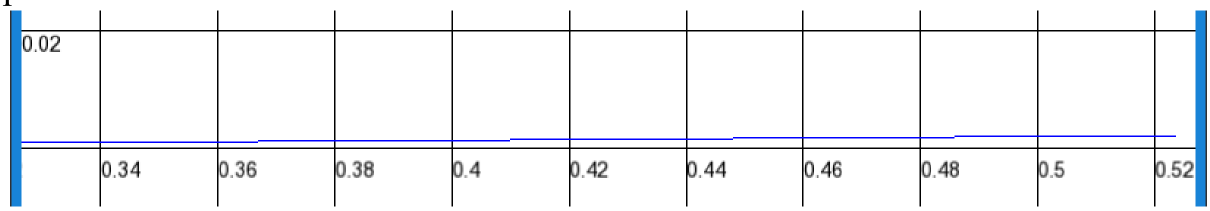
На графиках представлены численное (синий) и аналитические (зелёный) решения при разбиении N = 50

Явный метод

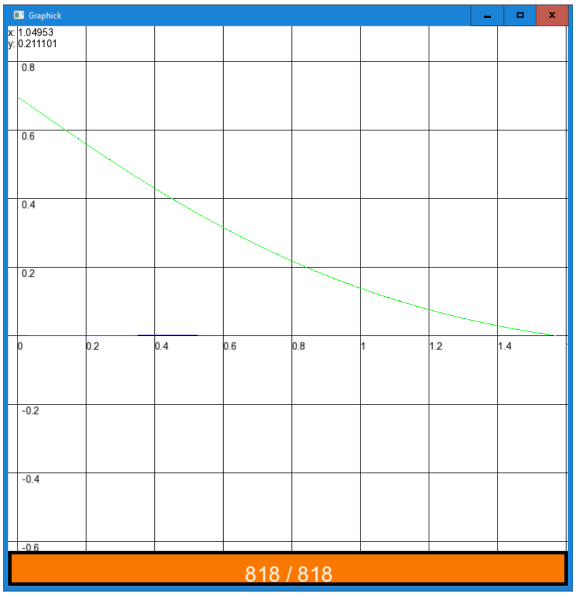


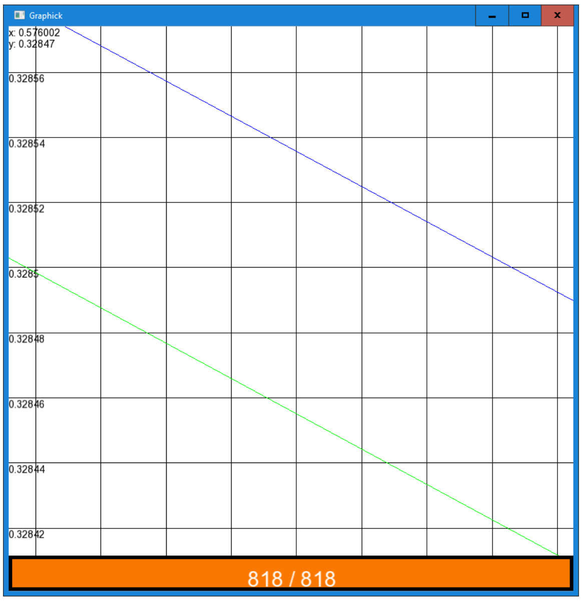


Далее приведён график, отражающий зависимость погрешности от шага разбиения

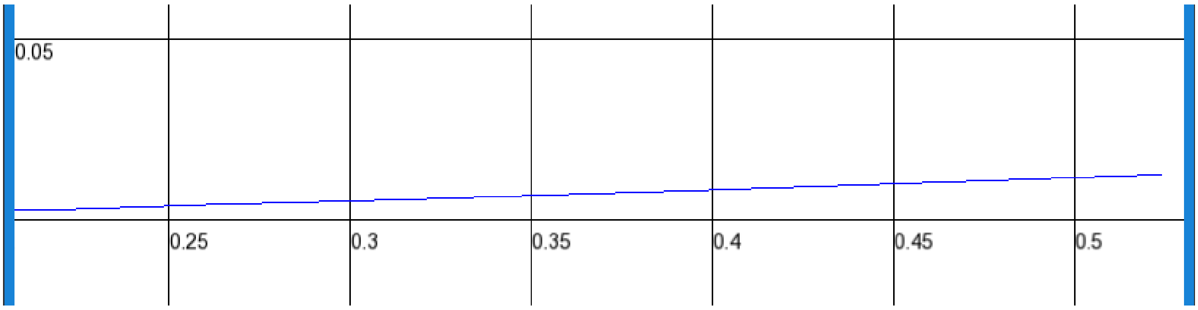


Неявный метод





Далее приведён график, отражающий зависимость погрешности от шага разбиения



Исходя из графика погрешности от длины шага можно сказать, что отклонение от искомого решения явным методом меньше, чем у неявного.

Лабораторная работа 7

Задание

Решить краевую задачу для дифференциального уравнения эллиптического типа. Аппроксимацию уравнения произвести с использованием центрально-разностной схемы. Для решения дискретного аналога применить следующие методы: метод простых итераций (метод Либмана), метод Зейделя, метод простых итераций с верхней релаксацией. Вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением $U(x,y)$. Исследовать зависимость погрешности от сеточных параметров h_x, h_y.

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=-u,\quad \begin{matrix} u(0,y)=0 \\ u\left(\frac{\pi}{2},y\right)=y \end{matrix},\quad u_y(x,0)=\sin x,\ u_y(x,1)-u(x,1)=0.$$

Аналитическое решение: $U(x,y)=y\sin x$.

Теоретические сведения

Постановка задачи

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=f(x,y),(x,y)\in\Omega$$
$$u(x,y)\big|_{\Gamma}=\varphi(x,y),(x,y)\in\Gamma$$

Рассмотрим краевую задачу в прямоугольнике $x\in[0,l_1], y\in[0,l_2]$, на которой наложим сетку $\omega_{h_1,h_2}=\{x_i=ih_1, i=\overline{0,N_1}; y_j=jh_2, j=\overline{0,N_2}\}$.

На этой сетке аппроксимируем дифференциальную задачу во внутренних узлах с помощью отношения конечных разностей по следующей схеме (вводится сеточная функция $u_{ij}=u(x_i,y_j), i=\overline{0,N_1}, j=\overline{0,N_2};$):

$$\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{h_1^2}+\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{h_2^2}+O\left(h_1^2+h_2^2\right)=f(x_i,y_j), i=\overline{1,N_1-1}, j=\overline{1,N_2-1}$$

$$f_{i,j}=f(x_i,y_j)$$

1. Метод Либмана. $u_{i,j}^{(k+1)}=\dfrac{1}{\left(\dfrac{2}{h_1^2}+\dfrac{2}{h_2^2}\right)}\left[u_{i+1,j}^{(k)}+u_{i-1,j}^{(k)}+u_{i,j-1}^{(k)}+u_{i,j+1}^{(k)}-h_1^2h_2^2f_{i,j}\right]$

2. Метод Зейделя. $u_{i,j}^{(k+1)}=\dfrac{1}{\left(\dfrac{2}{h_1^2}+\dfrac{2}{h_2^2}\right)}\left[u_{i+1,j}^{(k)}+u_{i-1,j}^{(k+1)}+u_{i,j-1}^{(k+1)}+u_{i,j+1}^{(k)}-h_1^2h_2^2f_{i,j}\right]$

3. Метод верхней релаксации.

$$u_{i,j}^{(k+1)}=(1-\theta)u_{i,j}^{(k)}+\theta\frac{1}{\left(\frac{2}{h_1^2}+\frac{2}{h_2^2}\right)}\left[u_{i+1,j}^{(k)}+u_{i-1,j}^{(k)}+u_{i,j-1}^{(k)}+u_{i,j+1}^{(k)}-h_1^2h_2^2f_{i,j}\right],\theta\in[1,2]$$

Код:

tools.h

```
#include <iostream>
#include <cmath>
#include <vector>
#include <iomanip>
#include "../../Matrix.h"
#include "../Graphica.h"
using namespace std;

struct BorderCondition {
    double a, b;
    double (* func)(double, double);
    double operator() (double x, double y) { return func(x, y); }
};
using BC = BorderCondition;

int c;
void setC(double x) { c = x; }

namespace IterType {
    enum IterType : unsigned short int {
        simple,
        zaydel,
        relax
    };
};

Matrix<double> Iter(size_t Nx, size_t Ny, double Lx, double Ly, double epsilon,
                    BC leftCondition, BC rightCondition, BC downCondition, BC upCondition,
                    double (*f)(double, double), IterType::IterType flag, double omega = 1.) {
    double hx = Lx / Nx, hy = Ly / Ny;
```

```
    double hx2 = pow(hx, 2), hy2 = pow(hy, 2);

    Matrix<double> res(Nx + 1, Ny + 1, 0.35);

    double coef1 = - downCondition.a / hy;
    double coef2 =   downCondition.b + coef1;

    double coef3 = - leftCondition.a / hx;
    double coef4 =   leftCondition.b + coef3;

    double coef5 = upCondition.a / hy;
    double coef6 = upCondition.b + coef5;

    double coef7 = rightCondition.a / hx;
    double coef8 = rightCondition.b + coef7;

    for (size_t i =  1, j =  0; i <  Nx; i++) res[i][j] = (downCondition (i * hx, j * hy) +
res[i]    [j + 1] * coef1) / coef2;
    for (size_t i =  0, j =  0; j <= Ny; j++) res[i][j] = (leftCondition (i * hx, j * hy) +
res[i + 1][j]     * coef3) / coef4;
    for (size_t i =  1, j = Ny; i <  Nx; i++) res[i][j] = (upCondition   (i * hx, j * hy) +
res[i]    [j - 1] * coef5) / coef6;
    for (size_t i = Nx, j =  0; j <= Ny; j++) res[i][j] = (rightCondition(i * hx, j * hy) +
res[i - 1][j]     * coef7) / coef8;

    Matrix<double> was;
    double coef9 = 2 / hx2 + 2 / hy2;
    do {
        was = res;
        for (size_t i = 1; i < Nx; i++) {
            for (size_t j = 1; j < Ny; j++) {
                if (flag == IterType::simple)
                    res[i][j] = ((was[i + 1][j] + was[i - 1][j]) / hx2 + (was[i][j + 1] +
was[i][j - 1]) / hy2 + c * was[i][j] - hx2 * hy2 * f(i * hx, j * hy)) / coef9;
                else if (flag == IterType::zaydel)
                    res[i][j] = ((was[i + 1][j] + res[i - 1][j]) / hx2 + (was[i][j + 1] +
res[i][j - 1]) / hy2 + c * was[i][j] - hx2 * hy2 * f(i * hx, j * hy)) / coef9;
                else if (flag == IterType::relax)
                    res[i][j] = omega * ((was[i + 1][j] + was[i - 1][j]) / hx2 + (was[i][j +
1] + was[i][j - 1]) / hy2 + c * was[i][j] - hx2 * hy2 * f(i * hx, j * hy)) / coef9
```

```
                                  + (1 - omega) * was[i][j];
            }
        }
        if (flag == IterType::simple) {
            for (size_t i =  1, j =  0; i <  Nx; i++) res[i][j] = (downCondition (i * hx, j * hy) + was[i]    [j + 1] * coef1) / coef2;
            for (size_t i =  0, j =  0; j <= Ny; j++) res[i][j] = (leftCondition (i * hx, j * hy) + was[i + 1][j]     * coef3) / coef4;
            for (size_t i =  1, j = Ny; i <  Nx; i++) res[i][j] = (upCondition   (i * hx, j * hy) + was[i]    [j - 1] * coef5) / coef6;
            for (size_t i = Nx, j =  0; j <= Ny; j++) res[i][j] = (rightCondition(i * hx, j * hy) + was[i - 1][j]     * coef7) / coef8;
        } else {
            for (size_t i =  1, j =  0; i <  Nx; i++) res[i][j] = (downCondition (i * hx, j * hy) + res[i]    [j + 1] * coef1) / coef2;
            for (size_t i =  0, j =  0; j <= Ny; j++) res[i][j] = (leftCondition (i * hx, j * hy) + res[i + 1][j]     * coef3) / coef4;
            for (size_t i =  1, j = Ny; i <  Nx; i++) res[i][j] = (upCondition   (i * hx, j * hy) + res[i]    [j - 1] * coef5) / coef6;
            for (size_t i = Nx, j =  0; j <= Ny; j++) res[i][j] = (rightCondition(i * hx, j * hy) + res[i - 1][j]     * coef7) / coef8;
        }
    } while ((was - res).normL() > epsilon);
    return res;
}
```

main.cpp

```
#include "tools.h"
#include <thread>
#include "../UI/bar.h"

double f         (double x, double y) { return 0.       ; }

double leftFunc  (double x, double y) { return 0.       ; }
double rightFunc (double x, double y) { return y        ; }

double downFunc  (double x, double y) { return sin(x)   ; }
double upFunc    (double x, double y) { return 0.       ; }

double answer    (double x, double y) { return y * sin(x); }
```

```
int main() {
    // элиптический тип
    ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
    setlocale(LC_ALL, "rus");

    setC(1.);

    loadFonts();
    Bar<int> bar;
    bar.ValueText.setFont(font);
    bar.setColors(sf::Color(0, 0, 0), sf::Color(250, 120, 0), sf::Color(120, 120, 120));
    bar.setWidth(scw);
    bar.setPosition(0, sch - STANDART_BAR_HEIGHT);
    drawableStuff.push_back(&bar);
    thread thread1(ShowGraphics);

    double Lx = M_PI_2, Ly = 1.;
    BC leftCondition  {0.,  1., leftFunc };
    BC rightCondition {0.,  1., rightFunc};
    BC downCondition  {1.,  0., downFunc };
    BC upCondition    {1., -1., upFunc   };

    cout << "какой метод использовать?\n1 - Либмана\n2 - Зейделя\n3 - Верхней релаксации\n>"; 
cout.flush();
    int ans; cin >> ans;
    IterType::IterType type = IterType::IterType(ans - 1);
    cout << "epsilon = "; cout.flush();
    double epsilon; cin >> epsilon;

    double omega = 0.;
    if (ans == '3') { cout << "omega = "; cout.flush(); cin >> omega; }

    cout << "с максимальным разбиением Nx Ny = "; cout.flush();
    size_t Nx, Ny; cin >> Nx >> Ny;
    Matrix<double> res = Iter(Nx, Ny, Lx, Ly, epsilon, leftCondition, rightCondition, 
downCondition, upCondition, f, type, omega);
    size_t K = res.dim.second - 1;
```

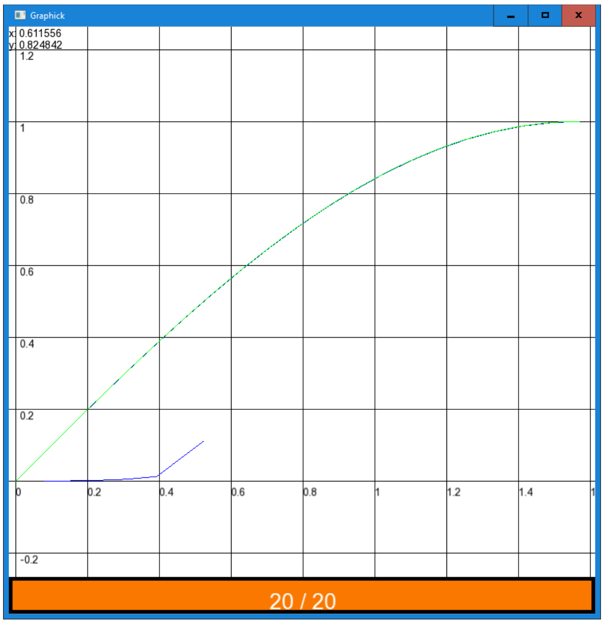
```
    vector<pair<double, double>> g1(Nx + 1);
    for (int i = 0; i < g1.size(); i++)
        g1[i] = {(i * Lx) / Nx, res[i][K]};

    vector<pair<double, double>> g2(5001);
    for (int i = 0; i < g2.size(); i++) {
        g2[i].first = (i * Lx) / 5000;
        g2[i].second = answer((i * Lx) / 5000, Ly);
    }
    makeGraphByPoints(g1);
    makeGraphByPoints(g2, sf::Color::Green);

    Scale<int> scale{0, (int)K, (int)K};
    bar.setValue(scale);

    thread thread2([&]{
        cout << "----------------------------------------------------------\n"
             << "|  Nx  :   Ny  :      hx     :     hy      :    error    |\n"
             << "+-----+--------+------------+------------+-------------+\n";

        vector<pair<double, double>> g3(Nx - 3 + 1);
        double error;
        Matrix<double> res;
        for (int n = 3; n <= min(Nx, Ny); n++) {
            error = 0;
            res = Iter(n, n, Lx, Ly, epsilon / n, leftCondition, rightCondition,
downCondition, upCondition, f, type, omega);

            for (int i = 0; i <= n; i++)
                for (int j = 0; j <= n; j++)
                    error += max(abs(answer((i * Lx) / n, (j * Ly) / n) - res[i][j]) - error,
0.);
            g3[n - 3] = {Lx / n, error};

            cout << "| " << setw(3) << n << " : " << setw(6) << n
                 << " : " << setw(11) << Lx / n << " : " << setw(11) << Ly / n
                 << " : " << setw(11) << error << " |\n"
                 << "+-----+--------+------------+------------+-------------+\n";
```

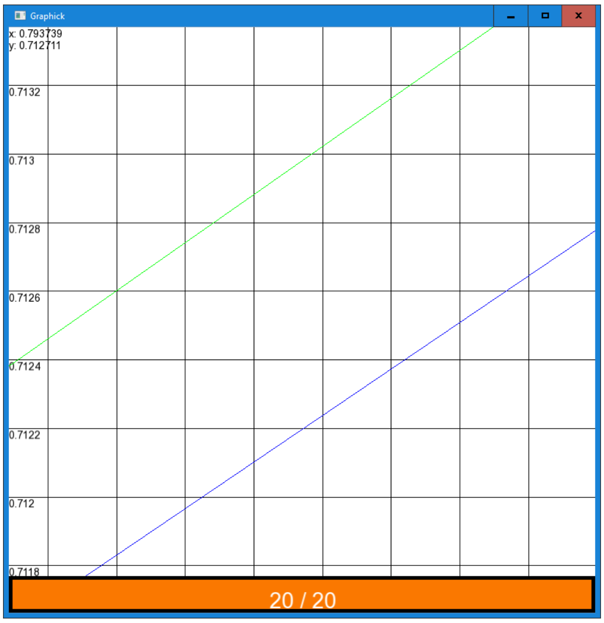
```
        }
        cout << '\n'; cout.flush();
        makeGraphByPoints(g3);
    });
    thread thread3([&]{
        while (window == nullptr || window->isOpen()) {
            if (window->hasFocus() && sf::Mouse::isButtonPressed(sf::Mouse::Left)) {
                if (sf::Mouse::getPosition(*window).y >= bar.getPosition().y) {
                    MouseBuffer = Mouse.getPosition(*window);
                    scale.cur = (double)scale.top *
(double)sf::Mouse::getPosition(*window).x / (double)scw;
                    normalize(scale);

                    for (int i = 0; i < g1.size(); i++)
                        graph[0][i].y = -res[i][scale.cur];

                    for (int i = 0; i < g2.size(); i++)
                        graph[1][i].y = -answer((i * Lx) / 5000, Ly * (double)scale.cur /
(double)scale.top);
                }
            }
        }
    });
    thread1.join();
    thread2.join();
    thread3.join();
    return 0;
}
```

Вывод:

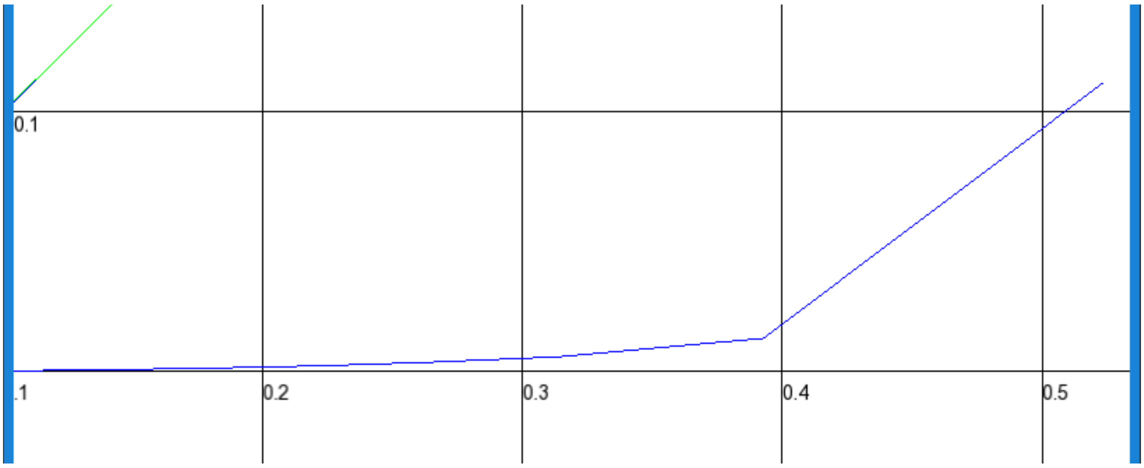
На графиках представлены численное (синий) и аналитические (зелёный) решения при разбиении N = 20 и epsilon = 0.000001

Метод Либмана

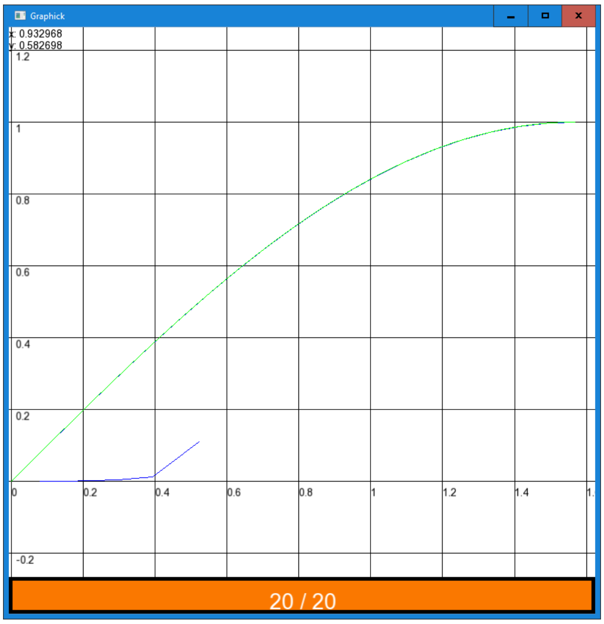


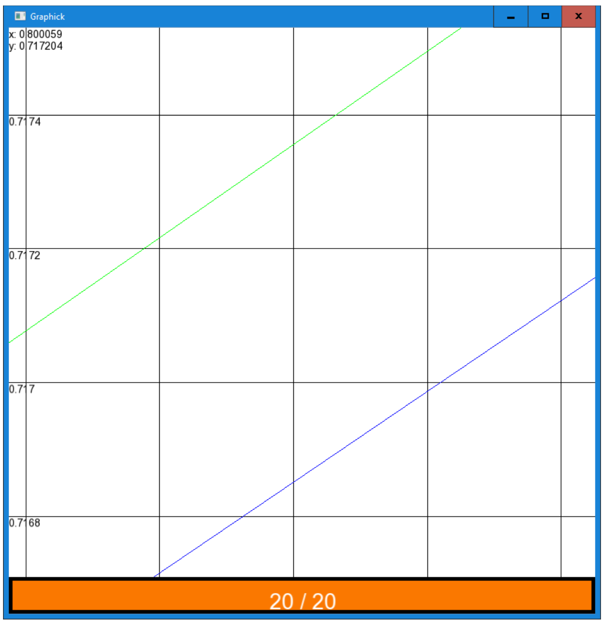


Далее приведён график, отражающий зависимость погрешности от шага разбиения

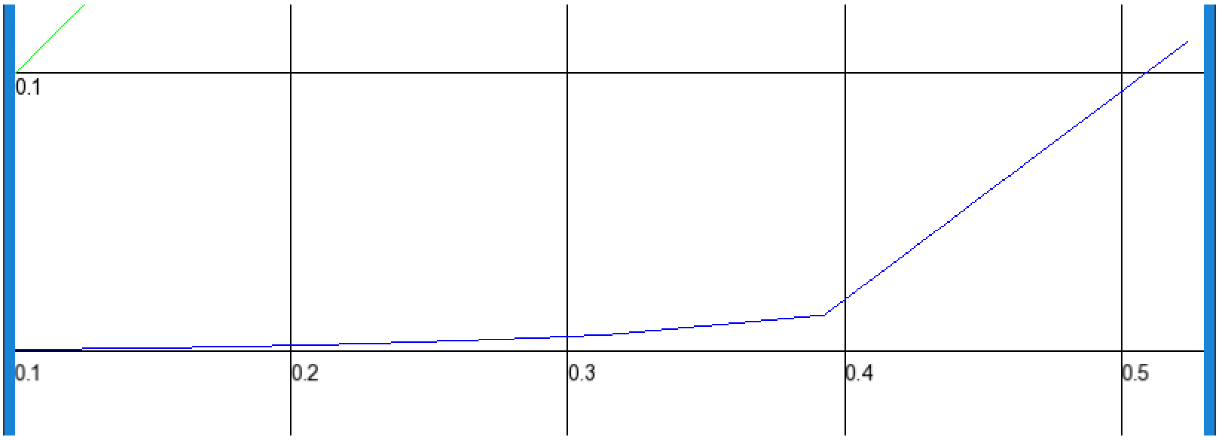


Метод Зейделя

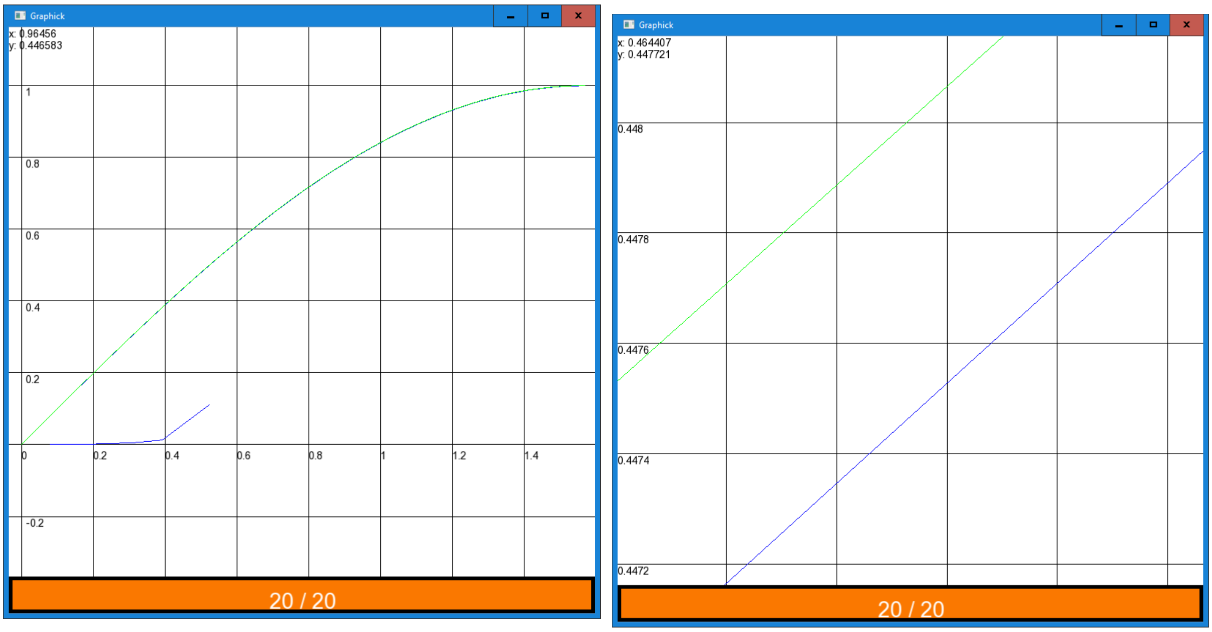




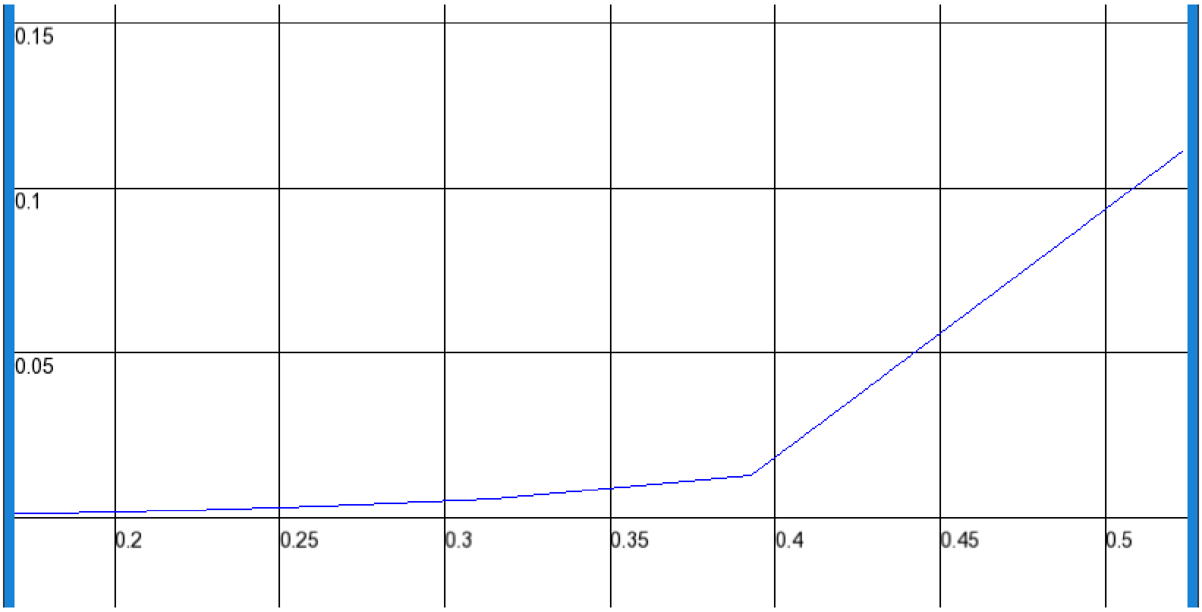
Далее приведён график, отражающий зависимость погрешности от шага разбиения



Метод верхей релаксации при omega = 1.1



Далее приведён график, отражающий зависимость погрешности от шага разбиения



Из результатов выполненной лабораторной работы можно заключить, что все рассмотренные методы примерно в равной степени точны, однако количество итераций, за которое эта точность достигается, у каждого из них отличается. Самым быстрым оказался метод Зейделя, за ним следует метод простых итераций с верхней релаксацией, а самым медленным оказался метод простых итераций.

Лабораторная работа 8

Задание

Используя схемы переменных направлений и дробных шагов, решить двумерную начально-краевую задачу для дифференциального уравнения параболического типа. В различные моменты времени вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением $U(x,t)$. Исследовать зависимость погрешности от сеточных параметров t, h_x, h_y.

$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} + a\frac{\partial^2 u}{\partial y^2}, a>0,$ $u(0,y,t) = \sinh(y)\exp(-3at),$ $u_x\left(\frac{\pi}{4}, y, t\right) = -2\sinh(y)\exp(-3at),$

$u_y(x,0,t) = \cos(2x)\exp(-3at),$
$u(x, \ln 2, t) = \frac{3}{4}\cos(2x)\exp(-3at),$ $u(x,y,0) = \cos(2x)\sinh(y)$

Аналитическое решение: $U(x,y,t) = \cos(2x)\sinh(y)\exp(-3at)$

Теоретические сведения

Постановка задачи

$$\widetilde{G}_T = \widetilde{G} \times [0,T]; t \in [0,T], \widetilde{G} = G + \Gamma, G = l_1 \times l_2$$
$$\frac{\partial u}{\partial t} = a\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + f(x,y,t), x \in (0,l_1), y \in (0,l_2), t>0$$
$$u(x,0,t) = \varphi_1(x,t), x \in [0,l_1], y=0, t>0$$
$$u_x(x,l_2,t) = \varphi_2(x,t), x \in [0,l_1], y=l_2, t>0$$
$$u_y(0,y,t) = \varphi_3(y,t), x=0, y \in [0,l_2], t>0$$
$$u(l_1,y,t) = \varphi_4(y,t), x=l_1, y \in [0,l_2], t>0$$
$$u(x,y,0) = \psi(x,y), x \in [0,l_1], y \in [0,l_2], t>0$$

Введем пространственно-временную сетку с шагами соответственно по переменным :

$$\omega^{\tau}_{h_1,h_2} = \{x_i = i h_1, i = \overline{0,I}; x_j = j h_2, j = \overline{0,J}; t^k = k\tau, k = 0,1,2,\ldots\}$$

и на этой сетке будем аппроксимировать дифференциальную задачу.

Метод переменных направлений:

$$\frac{u_{ij}^{k+1/2}-u_{ij}^{k}}{\tau/2}=\frac{a}{h_1^2}\left(u_{i+1\,j}^{k+1/2}-2u_{ij}^{k+1/2}+u_{i-1\,j}^{k+1/2}\right)+\frac{a}{h_2^2}\left(u_{i\,j+1}^{k}-2u_{ij}^{k}+u_{i\,j-1}^{k}\right)+f_{ij}^{k+1/2}$$

$$\frac{u_{ij}^{k+1}-u_{ij}^{k+1/2}}{\tau/2}=\frac{a}{h_1^2}\left(u_{i+1\,j}^{k+1/2}-2u_{ij}^{k+1/2}+u_{i-1\,j}^{k+1/2}\right)+\frac{a}{h_2^2}\left(u_{i\,j+1}^{k+1}-2u_{ij}^{k+1}+u_{i\,j-1}^{k+1}\right)+f_{ij}^{k+1/2}$$

Метод дробных шагов:

$$\frac{u_{ij}^{k+1/2}-u_{ij}^{k}}{\tau/2}=\frac{a}{h_1^2}\left(u_{i+1\,j}^{k+1/2}-2u_{ij}^{k+1/2}+u_{i-1\,j}^{k+1/2}\right)+\frac{f_{ij}^{k}}{2}$$

$$\frac{u_{ij}^{k+1}-u_{ij}^{k+1/2}}{\tau/2}=\frac{a}{h_2^2}\left(u_{i\,j+1}^{k+1}-2u_{ij}^{k+1}+u_{i\,j-1}^{k+1}\right)+\frac{f_{ij}^{k+1}}{2}$$

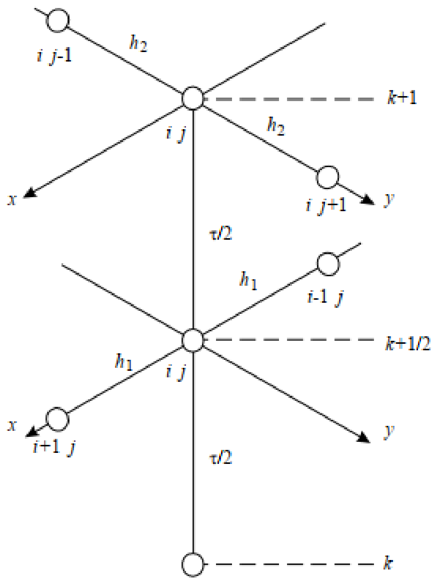


Рис.5.8. Шаблон схемы метода дробных шагов

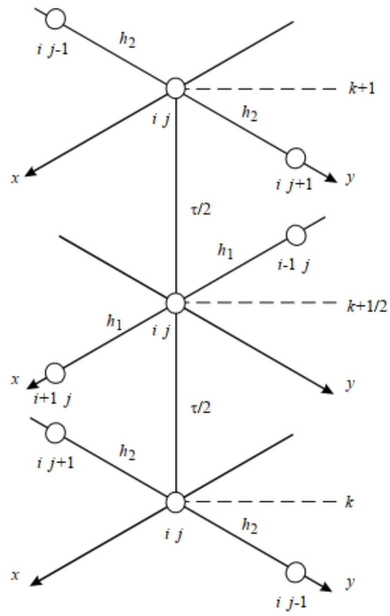


Рис. 5.7 Шаблон схемы метода переменных направлений

Код:

tools.h

```
#include <iomanip>

#include "TMA.h"

using namespace std;


struct BorderCondition {

    double a, b;

    double (* func)(double, double, double);

    double operator() (double x, double y, double t) { return func(x, y, t); }

};
```

```
using BC = BorderCondition;


double a = 1.;

void setA(double x) { a = x; }


vector<vector<vector<double>>> Method(size_t Nx, size_t Ny, size_t Nt, double Lx, double Ly, double Lt,
                                      BC leftCondition, BC rightCondition, BC downCondition, BC upCondition, BC initCondition,
                                      double (*f)(double, double, double), int type) {
    double hx = Lx / Nx, hy = Ly / Ny, tau = Lt / Nt;
    double hx2 = pow(hx, 2), hy2 = pow(hy, 2), ht_2 = tau / 2.;
    double sigmaX = a * tau / hx2, sigmaY = a * tau / hy2;


    vector<vector<vector<double>>> res(Nt + 1, vector<vector<double>>(Nx + 1, vector<double>(Ny + 1)));
    for (size_t i = 0; i < Nx + 1; i++)
        for (size_t j = 0; j < Ny + 1; j++)
            res[0][i][j] = initCondition(hx * i, hy * j, 0);


    Matrix<double> mx(3, Nx + 1), dx(1, Nx + 1), my(3, Ny + 1), dy(1, Ny + 1), u(Nx + 1, Ny + 1), r;


    double coef1 = - downCondition.a / hy;
    double coef2 =   downCondition.b + coef1;


    double coef3 = - leftCondition.a / hx;
    double coef4 =   leftCondition.b + coef3;


    double coef5 = upCondition.a / hy;
    double coef6 = upCondition.b + coef5;


    double coef7 = rightCondition.a / hx;
    double coef8 = rightCondition.b + coef7;


    // k + 1/2
    mx[0][0] = 0;
    mx[1][0] = leftCondition.b - leftCondition.a / hx;
    mx[2][0] = leftCondition.a / hx;
```

```
    for (size_t i = 1; i < Nx; i++) {
        mx[0][i] = sigmaX;
        mx[1][i] = -(2 + 2 * sigmaX);
        mx[2][i] = sigmaX;
    }
    mx[0][Nx] = -rightCondition.a / hx;
    mx[1][Nx] = rightCondition.b + rightCondition.a / hx;
    mx[2][Nx] = 0;
    // k + 1
    my[0][0] = 0;
    my[1][0] = downCondition.b - downCondition.a / hy;
    my[2][0] = downCondition.a / hy;
    for (size_t j = 1; j < Ny; j++) {
        my[0][j] = sigmaY;
        my[1][j] = -(2 + 2 * sigmaY);
        my[2][j] = sigmaY;
    }
    my[0][Ny] = -upCondition.a / hy;
    my[1][Ny] = upCondition.b + upCondition.a / hy;
    my[2][Ny] = 0;

    for (size_t k = 0; k < Nt; k++) {
        u.copy(Matrix(res[k]));
        // k + 1/2
        for (size_t j = 1; j < Ny; j++) {
            for (size_t i = 1; i < Nx; i++)
                dx[0][i] = -2 * res[k][i][j]
                           -sigmaY * (res[k][i][j + 1] - 2 * res[k][i][j] + res[k][i][j - 1])
* (1 - type)
                           -tau * f(i * hx, j * hy, (k + 0.5) * tau);
            dx[0][0] = leftCondition(0., j * hy, (k + 0.5) * tau);
            dx[0][Nx] = rightCondition(Lx, j * hy, (k + 0.5) * tau);
            u.copy(TMAsolve(mx, dx).T(), 0, j);
        }
        for (size_t j = 0; j <= Ny; j++) {
            res[k + 1][0] [j] = u[0] [j];
            res[k + 1][Nx][j] = u[Nx][j];
        }
```

```
        // k + 1
        for (size_t i = 1; i < Nx; i++) {
            for (size_t j = 1; j < Ny; j++)
                dy[0][j] = -2 * u[i][j]
                           -sigmaX * (u[i + 1][j] - 2 * u[i][j] + u[i - 1][j]) * (1 - type)
                           -tau * f(i * hx, j * hy, (k + 0.5) * tau);
            dy[0][0] = downCondition(i * hx, 0., (k + 0.5) * tau);
            dy[0][Ny] = upCondition(i * hx, Ly, (k + 0.5) * tau);


            r = TMAsolve(my, dy);
            for (size_t j = 0; j <= Ny; j++)
                res[k + 1][i][j] = r[0][j];
        }
        res[k + 1][0] [0]  = (leftCondition (0., 0., (k + 0.5) * tau) + res[k + 1][1]     [0]
* coef3) / coef4;
        res[k + 1][0] [Ny] = (leftCondition (0., Ly, (k + 0.5) * tau) + res[k + 1][1]     [Ny]
* coef3) / coef4;
        res[k + 1][Nx][0]  = (rightCondition(Lx, 0., (k + 0.5) * tau) + res[k + 1][Nx - 1][0]
* coef7) / coef8;
        res[k + 1][Nx][Ny] = (rightCondition(Lx, Ly, (k + 0.5) * tau) + res[k + 1][Nx - 1][Ny]
* coef7) / coef8;
    }
    return res;
}


vector<vector<vector<double>>> AlternatingDirectionMethod(size_t Nx, size_t Ny, size_t Nt,
double Lx, double Ly, double Lt,
                                    BC leftCondition, BC rightCondition, BC downCondition, BC
upCondition, BC initCondition,
                                    double (*f)(double, double, double)) {
    return Method(Nx, Ny, Nt, Lx, Ly, Lt, leftCondition, rightCondition, downCondition,
upCondition, initCondition, f, 0);
}
vector<vector<vector<double>>> FractionalStepsMethod(size_t Nx, size_t Ny, size_t Nt, double
Lx, double Ly, double Lt,
                                    BC leftCondition, BC rightCondition, BC downCondition, BC
upCondition, BC initCondition,
                                    double (*f)(double, double, double)) {
    return Method(Nx, Ny, Nt, Lx, Ly, Lt, leftCondition, rightCondition, downCondition,
upCondition, initCondition, f, 1);
}
```

main.cpp

```
#include "tools.h"

#include <thread>

#include "../Graphica.h"

#include "../UI/bar.h"


double f         (double x, double y, double t) { return 0.
; }


double leftFunc  (double x, double y, double t) { return              sinh(y) * exp(-3 * a *
t); }

double rightFunc (double x, double y, double t) { return -2 *         sinh(y) * exp(-3 * a *
t); }


double downFunc  (double x, double y, double t) { return cos(2 * x) *           exp(-3 * a *
t); }

double upFunc    (double x, double y, double t) { return cos(2 * x) * 0.75 *    exp(-3 * a *
t); }


double initFunc  (double x, double y, double t) { return cos(2 * x) * sinh(y)
; }


double answer    (double x, double y, double t) { return cos(2 * x) * sinh(y) * exp(-3 * a *
t); }


int main() {

    // параболический тип

    ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);

    setlocale(LC_ALL, "rus");


    setA(1.);


    loadFonts();


    Bar<int> barY;

    barY.ValueText.setFont(font);

    barY.setColors(sf::Color(0, 0, 0), sf::Color(250, 120, 0), sf::Color(120, 120, 120));

    barY.setWidth(scw);

    barY.setPosition(0, sch - STANDART_BAR_HEIGHT * 2);

    drawableStuff.push_back(&barY);
```

```
    Bar<int> barT;
    barT.ValueText.setFont(font);
    barT.setColors(sf::Color(0, 0, 0), sf::Color(250, 120, 0), sf::Color(120, 120, 120));
    barT.setWidth(scw);
    barT.setPosition(0, sch - STANDART_BAR_HEIGHT);
    drawableStuff.push_back(&barT);

    PlacedText textY, textT;
    textY.setString("y = "); textT.setString("t = ");
    textY.setPosition(barY.ValueText.getPosition() - sf::Vector2f(textY.getSize().x, 0.f));
    textT.setPosition(barT.ValueText.getPosition() - sf::Vector2f(textT.getSize().x, 0.f));
    drawableStuff.push_back(&textY); drawableStuff.push_back(&textT);

    thread thread1(ShowGraphics);

    double Lx = M_PI_4, Ly = log(2.), Lt = 0.5;
    BC leftCondition    {0., 1., leftFunc };
    BC rightCondition   {1., 0., rightFunc};
    BC downCondition    {1., 0., downFunc };
    BC upCondition      {0., 1., upFunc   };
    BC initCondition    {0., 1., initFunc };

    cout << "какой метод использовать?\n1 - переменных направлений\n2 - дробных шагов\n>";
cout.flush();
    int ans; cin >> ans;

    auto resFunc = ans == 1 ? AlternatingDirectionMethod : FractionalStepsMethod;

    cout << "с максимальным разбиением Nx Ny Nt = "; cout.flush();
    size_t Nx, Ny, Nt; cin >> Nx >> Ny >> Nt;
    vector<vector<vector<double>>> res = resFunc(Nx, Ny, Nt, Lx, Ly, Lt, leftCondition,
rightCondition, downCondition, upCondition, initCondition, f);

    vector<vector<vector<double>>> answerM(Nt + 1, vector<vector<double>>(Nx + 1,
vector<double>(Ny + 1)));
    double error = 0.;
    for (size_t i = 0; i < Nt + 1; i++)
        for (size_t j = 0; j < Nx + 1; j++)
```

```
            for (size_t k = 0; k < Ny + 1; k++) {
                answerM[i][j][k] = answer((j * Lx) / Nx, (k * Ly) / Ny, (i * Lt) / Nt);
                error = max(error, abs(answerM[i][j][k] - res[i][j][k]));
            }

    cout << "error = " << error << '\n'; cout.flush();

    vector<pair<double, double>> g1(Nx + 1);
    for (int i = 0; i < g1.size(); i++)
        g1[i] = {(i * Lx) / Nx, res[Nt][i][Ny]};

    vector<pair<double, double>> g2(5001);
    for (int i = 0; i < g2.size(); i++) {
        g2[i].first = (i * Lx) / 5000;
        g2[i].second = answer((i * Lx) / 5000, Ly, Lt);
    }
    makeGraphByPoints(g1);
    makeGraphByPoints(g2, sf::Color::Green);

    Scale<int> scaleT{0, (int)Nt, (int)Nt};
    barT.setValue(scaleT);
    Scale<int> scaleY{0, (int)Ny, (int)Ny};
    barY.setValue(scaleY);

    thread thread2([&]{
        cout << "--------------------------------------------------------------\n"
             << "|  N  :     hx      :     hy      :     tau     :    error    |\n"
             << "+-----+-------------+-------------+-------------+-------------+\n";

        vector<pair<double, double>> g3(50 - 3 + 1);
        double error;
        vector<vector<vector<double>>> res;
        for (int n = 3; n <= 50; n++) {
            error = 0;
            res = resFunc(n, n, n, Lx, Ly, Lt, leftCondition, rightCondition, downCondition,
upCondition, initCondition, f);

            for (size_t i = 0; i < n + 1; i++)
```

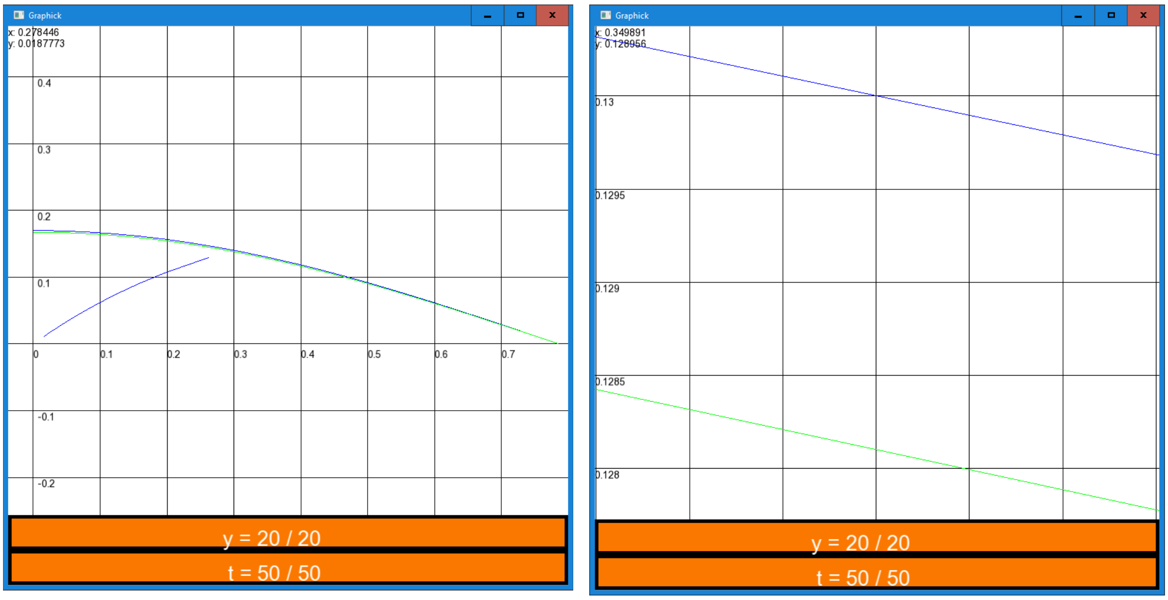
```
                for (size_t j = 0; j < n + 1; j++)
                    for (size_t k = 0; k < n + 1; k++)
                        error = max(error, abs(answer((j * Lx) / n, (k * Ly) / n, (i * Lt) /
n) - res[i][j][k]));
            g3[n - 3] = {Lx / n, error};


            cout << "| " << setw(3) << n << " : " << setw(11) << Lx / n << " : " << setw(11)
                 << Ly / n << " : " << setw(11) << Lt / n << " : " << setw(11) << error << "
|\n"
                 << "+-----+-------------+-------------+-------------+-------------+\n";
        }
        cout << '\n'; cout.flush();
        makeGraphByPoints(g3);
    });
    thread thread3([&]{
        while (window == nullptr || window->isOpen()) {
            if (window->hasFocus() && sf::Mouse::isButtonPressed(sf::Mouse::Left)) {
                if (sf::Mouse::getPosition(*window).y >= barT.getPosition().y ||
sf::Mouse::getPosition(*window).y >= barY.getPosition().y) {
                    MouseBuffer = Mouse.getPosition(*window);
                    if (sf::Mouse::getPosition(*window).y >= barT.getPosition().y) {
                        scaleT.cur = (double)scaleT.top *
(double)sf::Mouse::getPosition(*window).x / (double)scw;
                        normalize(scaleT);
                    } else {
                        scaleY.cur = (double)scaleY.top *
(double)sf::Mouse::getPosition(*window).x / (double)scw;
                        normalize(scaleY);
                    }
                    textY.setPosition(barY.ValueText.getPosition() -
sf::Vector2f(textY.getSize().x, 0.f));
                    textT.setPosition(barT.ValueText.getPosition() -
sf::Vector2f(textT.getSize().x, 0.f));


                    for (int i = 0; i < g1.size(); i++)
                        graph[0][i].y = -res[scaleT.cur][i][scaleY.cur];


                    for (int i = 0; i < g2.size(); i++)
                        graph[1][i].y = -answer((i * Lx) / 5000, Ly * (double)scaleY.cur /
(double)scaleY.top, Lt * (double)scaleT.cur / (double)scaleT.top);
                }
```

```
            }
        }
    });
    thread1.join();
    thread2.join();
    thread3.join();
    return 0;
}
```

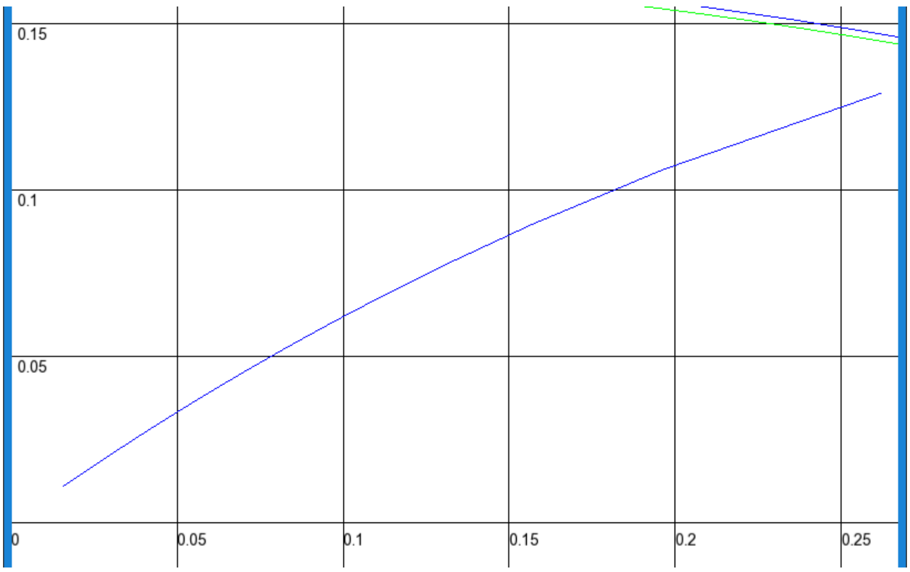
Вывод:

На графиках представлены численное (синий) и аналитические (зелёный) решения при разбиении N = 50

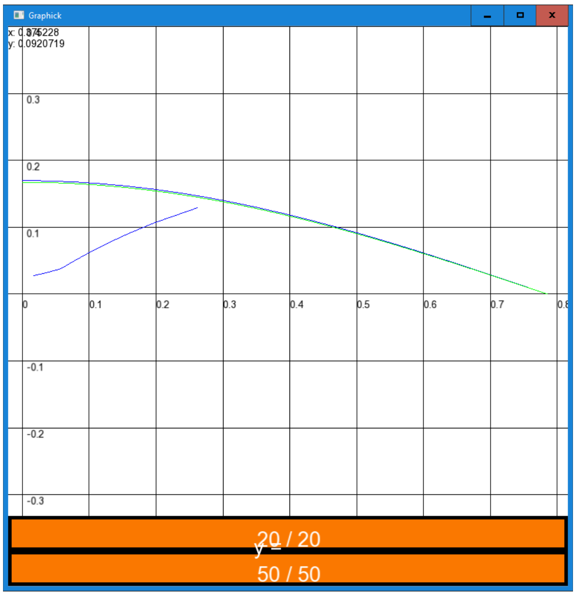
Метод переменных направлений

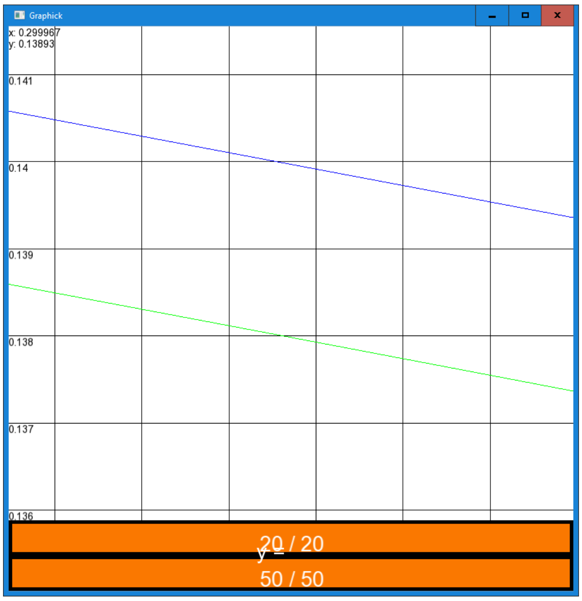


Далее приведён график, отражающий зависимость погрешности от шага разбиения

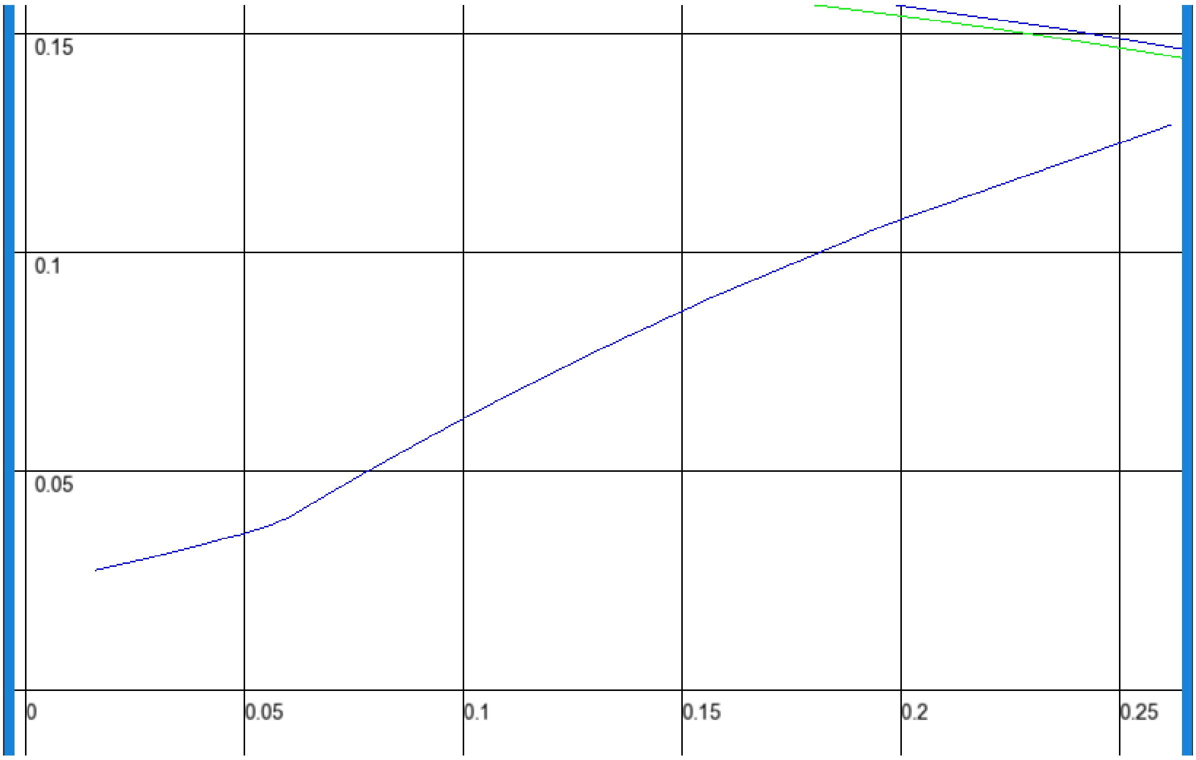


Метод дробных шагов





Далее приведён график, отражающий зависимость погрешности от шага разбиения



С увеличением шага сетки погрешность растет линейно.