**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

БАЛТИЙСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ВОЕНМЕХ» им. Д.Ф. УСТИНОВА



Дисциплина Численные методы гидроаэродинамики

практическая работа № 3

Решение двумерного уравнения теплопроводности

с помощью разностной схемы с расщеплением

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил студент | | Анкудинов А.Н. | |
| Фамилия И.О. | | | |
| Группа | А571 | |  |
| Преподаватель | |  | |
|  | | Фамилия И. О. | |

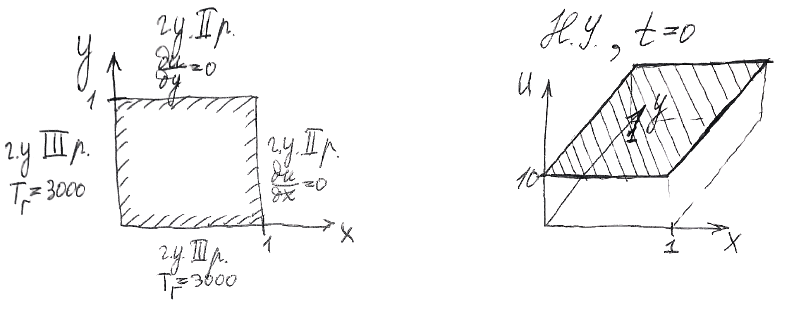
Санкт-Петербург

2020 г.

*Цель работы* - решить уравнение теплопроводности, применив разностную схему с расщеплением

*Исходные данные*

*Вариант 1*

**

*Теоретические сведения*

Уравнение двумерной нестационарной теплопроводности имеет вид:

где *U* – температура, *aT* – коэффициент теплопроводности.

Начальные условия описываются уравнением:

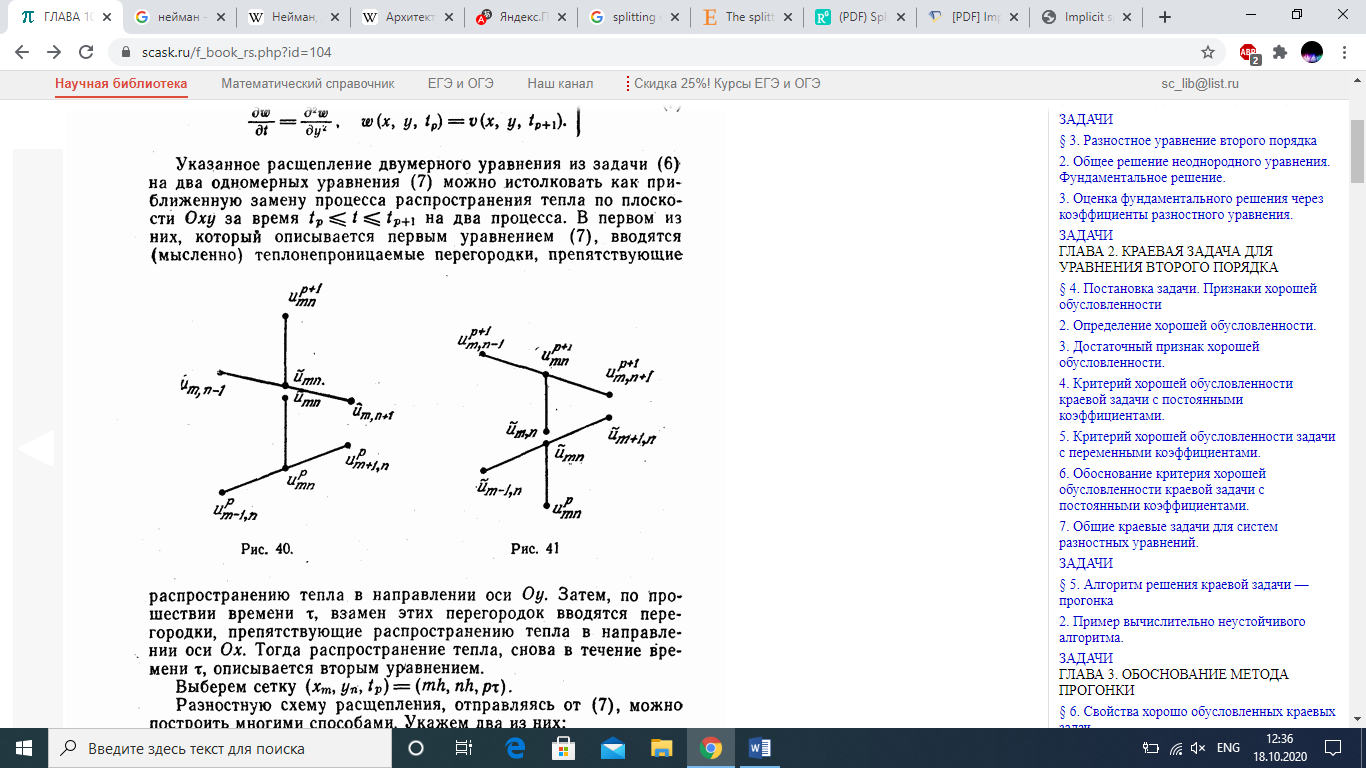
т.е. значения температур по всей поверхности пластинки в начальный момент времени известны.

Неявная разностная схема для уравнения одномерной нестационарной теплопроводности:

Порядок аппроксимации: по времени *()* – первый, по шагу вдоль оси *x ()* и по шагу вдоль оси *y ()* – второй.

Процесс распространения тепла по плоскости *Oxy* за время от *tn* до *tn+1*приближенно заменяется на два процесса. В первом из них мысленно вводятся теплонепроницаемые перегородки, препятствующие распространению тепла вдоль *Oy*. Затем вводятся теплонепроницаемые перегородки, препятствующие распространению тепла вдоль *Ox*. Каждый из процессов происходит в течение времени .

Промежуточное значение , а затем значение на новом слое находятся по неявной разностной схеме.



*Ход работы*

Необходимо исследовать сеточную сходимость по оси *х*. Для этого будем увеличивать количество разбиений N*х* по оси *x*. Расчёт проведём при T = 0.5 c, , , .

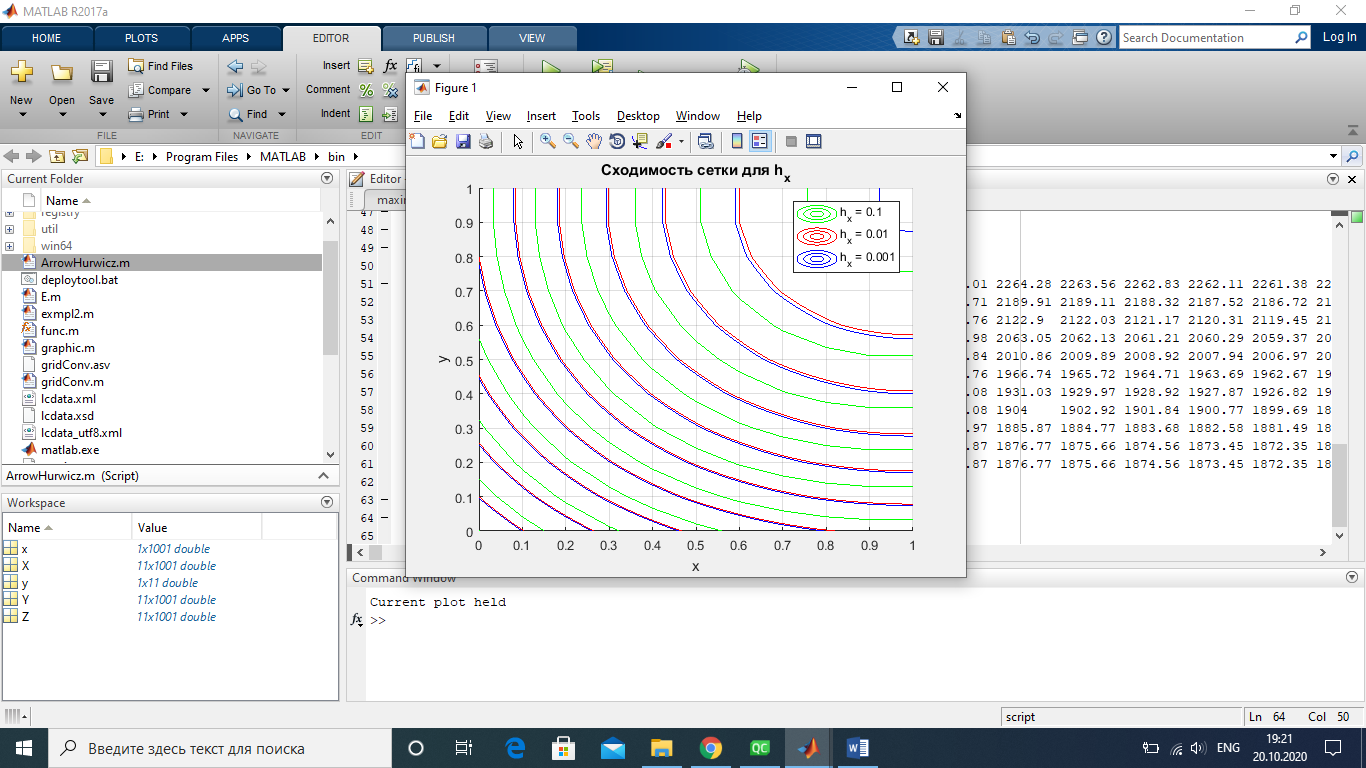


Рисунок 1 – Сеточная сходимость для шага по оси *х*

Видно, что при *hx* = 0.01 и *hx* = 0.001 линии уровня распределения температур практически сливаются - следовательно, брать *hx* < 0.01 не имеет смысла. То есть, при N*х*= 100 достигнута сеточная сходимость.

Теперь исследуем сеточную сходимость по оси *у*. Для этого будем увеличивать количество разбиений N по оси *у*.

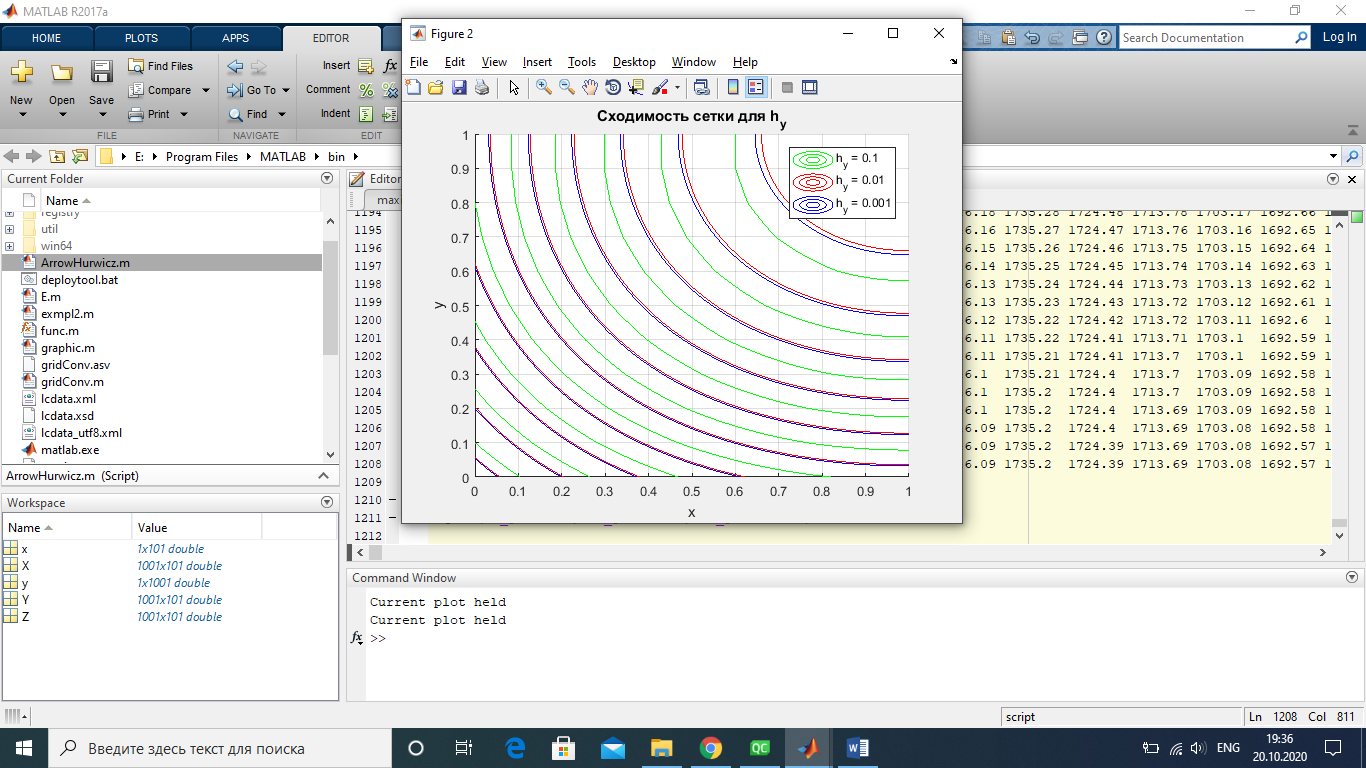


Рисунок 2 – Сеточная сходимость для шага по оси *у*

Результаты исследования сходимости сетки по оси *у* получились такими же, что и для оси *х*: для расчета имеет смысл взять шаг *hу* = 0.01. То есть при N*у* = 100 также достигнута сеточная сходимость.

Теперь исследуем сеточную сходимость для величины шага по времени , уменьшая его величину.

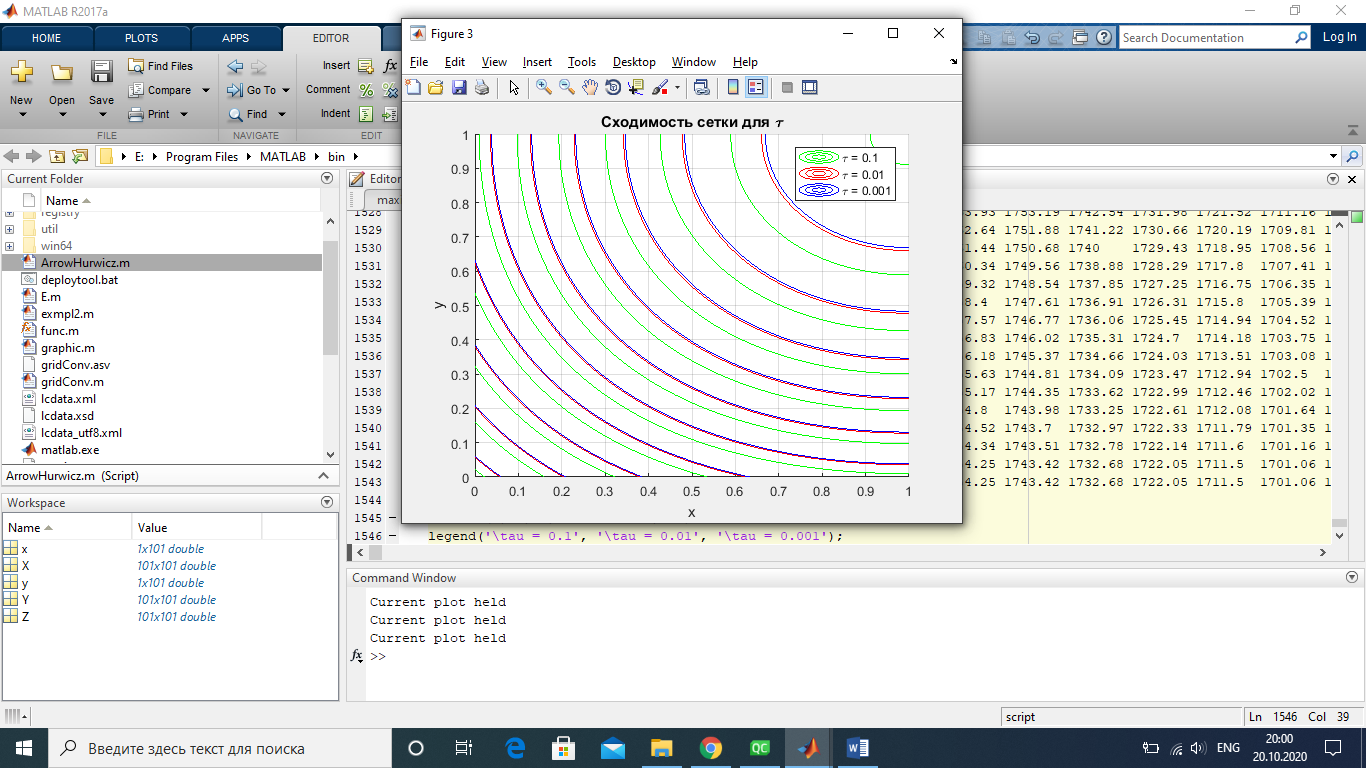
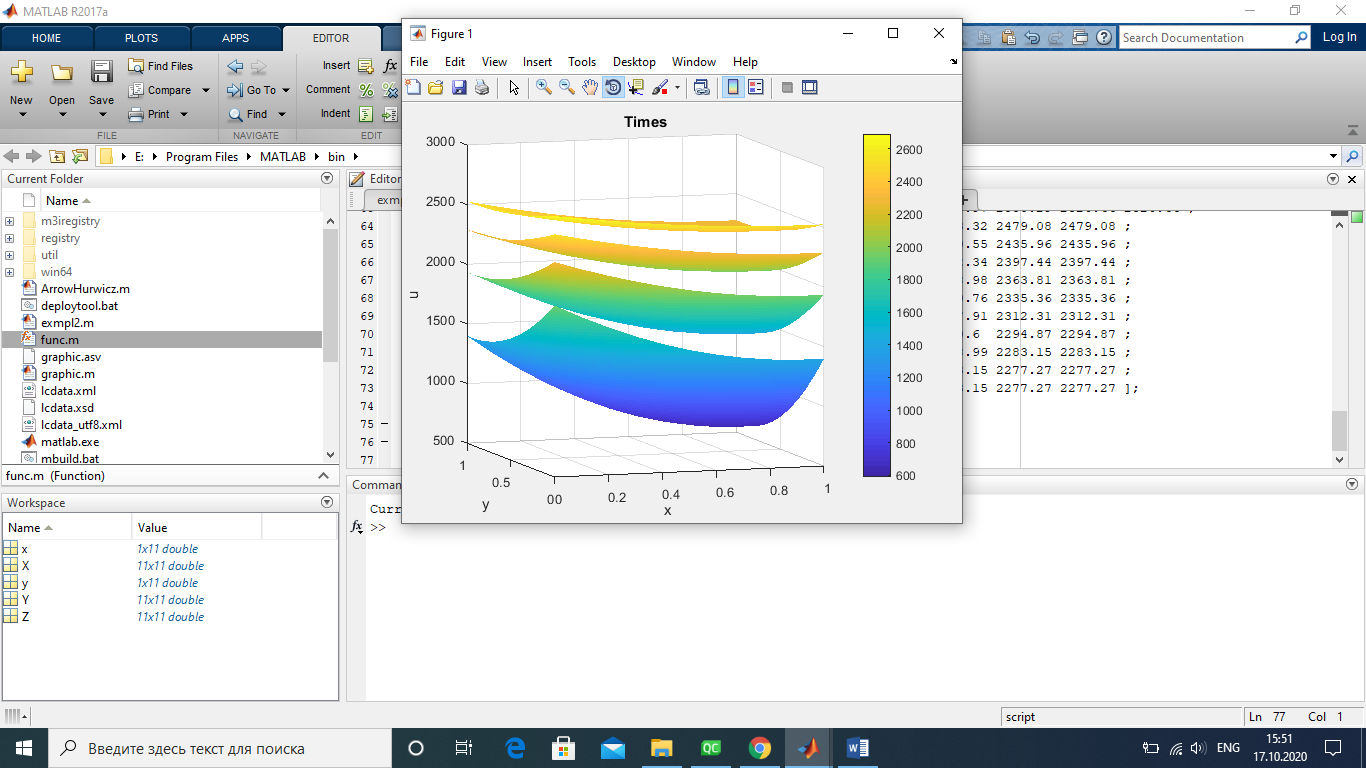


Рисунок 3 – Сеточная сходимость для шага по времени

Видно, что при = 0.01 и при = 0.001 линии уровня распределения температур практически сливаются - следовательно, брать шаг по времени не имеет смысла. То есть, при = 0.01 достигнута сеточная сходимость.

Теперь проведём расчёты для T = 0.25 c, T = 0.5 c, T = 0.75 c, T = 1 c.



*t = 3T/4*

*t = T/4*

*t = T/2*

*t = T*

Рисунок 4 – Распределение температур по поверхности пластинки во времени

*Вывод:* можно увидеть, что с течением времени температура по поверхности пластинки становится более однородной.

*Достоинство разностной схемы с расщеплением:* безусловная устойчивость.

*Недостатки:* не может быть распространена на трехмерный случай, порядок аппроксимации по времени – первый.

*Код программы*

#include <iostream>

#include <vector>

#include <cmath>

using namespace std;

const int a\_t = 1, alpha = 1, T\_g = 3000, T = 1, A = 1, C = 1;

int Ntau;

double hy, hx, tau, Bx, By;

vector<double> y, x, time, kx, Dx, lx, ky, Dy, ly;

vector<vector<double>> u;

enum **computationType** {GRID, TIMES};

void **setVectors**(const int Nx, const int Ny, const double tau, const computationType type);

int **main**()

{

// Исследуем сходимости

// Исследуем сходимость по hx

// cout << "\n\n\nNx = 10:\n";

// setVectors(10, 10, 0.01, GRID);

// cout << "\n\n\nNx = 100:\n";

// setVectors(100, 10, 0.01, GRID);

// cout << "\n\n\nNx = 1000:\n";

// setVectors(1000, 10, 0.01, GRID);

const int pickNx = 100;

// Исследуем теперь сходимость по hy

// cout << "\n\n\nNy = 10:\n";

// setVectors(pickNx, 10, 0.01, GRID);

// cout << "\n\n\nNy = 100:\n";

// setVectors(pickNx, 100, 0.01, GRID);

// cout << "\n\n\nNy = 1000:\n";

// setVectors(pickNx, 1000, 0.01, GRID);

const int pickNy = 100;

// Исследуем теперь сходимость по tau

// cout << "\n\n\ntau = 0.1:\n";

// setVectors(pickNx, pickNy, 0.1, GRID);

// cout << "\n\n\ntau = 0.01:\n";

// setVectors(pickNx, pickNy, 0.01, GRID);

// cout << "\n\n\ntau = 0.001:\n";

// setVectors(pickNx, pickNy, 0.001, GRID);

const double pickTau = 0.01;

//После исследования сходимости можем посчитать для разных времен

setVectors(pickNx, pickNy, pickTau, TIMES);

return 0;

}

void **setVectors**(const int Nx, const int Ny, const double tau, const computationType type)

{

hy = (double)1 / Ny;

y.clear(); y.resize(Ny+1);

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

y[j] = j\*hy;

hx = (double)1 / Nx;

x.clear(); x.resize(Nx+1);

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

x[i] = i\*hx;

u.clear(); u.resize(Nx+1);

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

u[i].resize(Ny+1);

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

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

u[i][j] = 10;

Ntau = ceil(T / tau); //возникает баг: Ntau = 1/0.01 = 99, а не 100 - пришлось округлять до большего

time.clear(); time.resize(Ntau+1);

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

time[i] = i\*tau;

//Коэффициенты для прогонки вдоль x

Bx = 1 + hx\*hx/(2\*a\_t\*tau);

kx.clear(); kx.resize(Nx);

kx[0] = a\_t/hx / (alpha + a\_t/hx);

lx.clear(); lx.resize(Nx);

lx[0] = alpha\*T\_g / (alpha + a\_t/hx);

Dx.clear(); Dx.resize(Nx);

//Коэффициенты для прогонки вдоль y

By = 1 + hy\*hy/(2\*a\_t\*tau);

ky.clear(); ky.resize(Ny);

ky[0] = a\_t/hy / (alpha + a\_t/hy);

ly.clear(); ly.resize(Ny);

ly[0] = alpha\*T\_g / (alpha + a\_t/hy);

Dy.clear(); Dy.resize(Ny);

//Вычисление

double time1 = Ntau/4, time2 = Ntau/2, time3 = 3\*Ntau/4, time4 = Ntau;

for (int t = 1; t <= Ntau; ++t) //внешний цикл - шагаем по времени

{

for(int j = 0; j <= Ny; ++j) // I этап: шагаем по y - для каждого y прогоняем вдоль x

{

for (int i = 1; i < Nx; ++i) //находим k, D, l во внутренних узлах - прямая прогонка - по x

{

kx[i] = -C / (A\*kx[i-1] - 2\*Bx);

Dx[i] = -hx\*hx\*u[i][j] / a\_t / tau;

lx[i] = (Dx[i] - A\*lx[i-1]) / (A\*kx[i-1] - 2\*Bx);

}

u[Nx][j] = lx[Nx-1] / (1 - kx[Nx-1]); //значение на правой границе

for(int i = Nx; i > 0; --i) //находим значения u справа налево - обратная прогонка - по x

{

u[i-1][j] = kx[i-1]\*u[i][j] + lx[i-1]; //u вдоль x по каждому конкретному y

}

} // здесь мы получили u промежуточное - учли распространение тепла только вдоль x

for(int i = 0; i <= Nx; ++i) // II этап: шагаем по x - для каждого x прогоняем вдоль y (на I этапе найдя промежуточное u)

{

for (int j = 1; j < Ny; ++j) //находим k, D, l во внутренних узлах - прямая прогонка - по y

{

ky[j] = -C / (A\*ky[j-1] - 2\*By);

Dy[j] = -hy\*hy\*u[i][j] / a\_t / tau;

ly[j] = (Dy[j] - A\*ly[j-1]) / (A\*ky[j-1] - 2\*By);

}

u[i][Ny] = ly[Ny-1] / (1 - ky[Ny-1]); //значение на правой границе

for(int j = Ny; j > 0; --j) //находим значения u справа налево - обратная прогонка - по y

{

u[i][j-1] = ky[j-1]\*u[i][j] + ly[j-1]; //u вдоль y по каждому конкретному x

}

}//здесь мы получили u(n+1) - итоговые значения u на новом временном слое

bool condition;

if(type == GRID)

condition = (t == time2); //условие для расчета сеточной сходимости - вывод только в моменты t = T/2

else if(type == TIMES)

condition = (t == time1 || t == time2 || t == time3 || t == time4); //условие для вывода в 4 момента времени

if(condition) //вывод значений в нужные моменты времени

{

cout << "\nt = " << time[t] << ":\n";

//Информация в консоль о значениях u в узлах сетки: [X строк на Y столбцов] и [Y строк на X столбцов]

// for(int i = 0; i <= Nx; ++i) //сетка [X строк на Y столбцов]

// {

// cout << "x = " << x[i] << ":\t";

// for(int j = 0; j <= Ny; ++j)

// cout << u[i][j] << "\t"; //вектор значений u вдоль y для конкретного x[i]

// cout << endl;

// }

// cout << endl;

// for(int j = 0; j <= Ny; ++j) //сетка [Y строк на X столбцов]

// {

// cout << "y = " << y[j] << ":\t";

// for(int i = 0; i <= Nx; ++i)

// cout << u[i][j] << "\t"; //вектор значений u вдоль x для конкретного y[j]

// cout << endl;

// }

//массив Z размерности [Y строк на X столбцов] - копирую в matlab для построения 3d-графика

cout << endl;

cout << "Z = [\t";

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

{

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

{

cout << u[i][j] << "\t"; //вектор значений u вдоль x для конкретного y[j]

}

if(j != Ny) cout << ";";

else cout << "];";

cout << endl;

}

}

}

}