**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

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



Дисциплина Численные методы гидроаэродинамики

практическая работа № 2

Решение уравнения теплопроводности

с помощью неявной разностной схемы

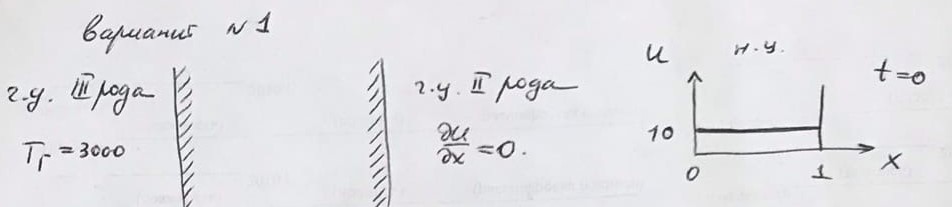
|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил студент | | Анкудинов А.Н. | |
| Фамилия И.О. | | | |
| Группа | А571 | |  |
| Преподаватель | |  | |
|  | | Фамилия И. О. | |

Санкт-Петербург

2020 г.

*Цель работы* - решить уравнение теплопроводности, применив неявную разностную схему.

*Исходные данные*

**

*Теоретические сведения*

Уравнение одномерной нестационарной теплопроводности имеет вид:

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

Начальные условия описываются уравнением:

т.е. значения температур по всей длине стенки в начальный момент времени известны.

Неявная разностная схема для уравнения одномерной нестационарной теплопроводности:

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

Получим уравнение для нахождения температуры в *i-1* узле:

,

где и - прогоночные коэффициенты, вычисляемые по формулам:

,

где - коэффициенты, имеющие следующие формулы для вычисления:

Из граничных условий II рода на правой границе получим уравнение нахождения температуры на правой границе:

*Ход работы*

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

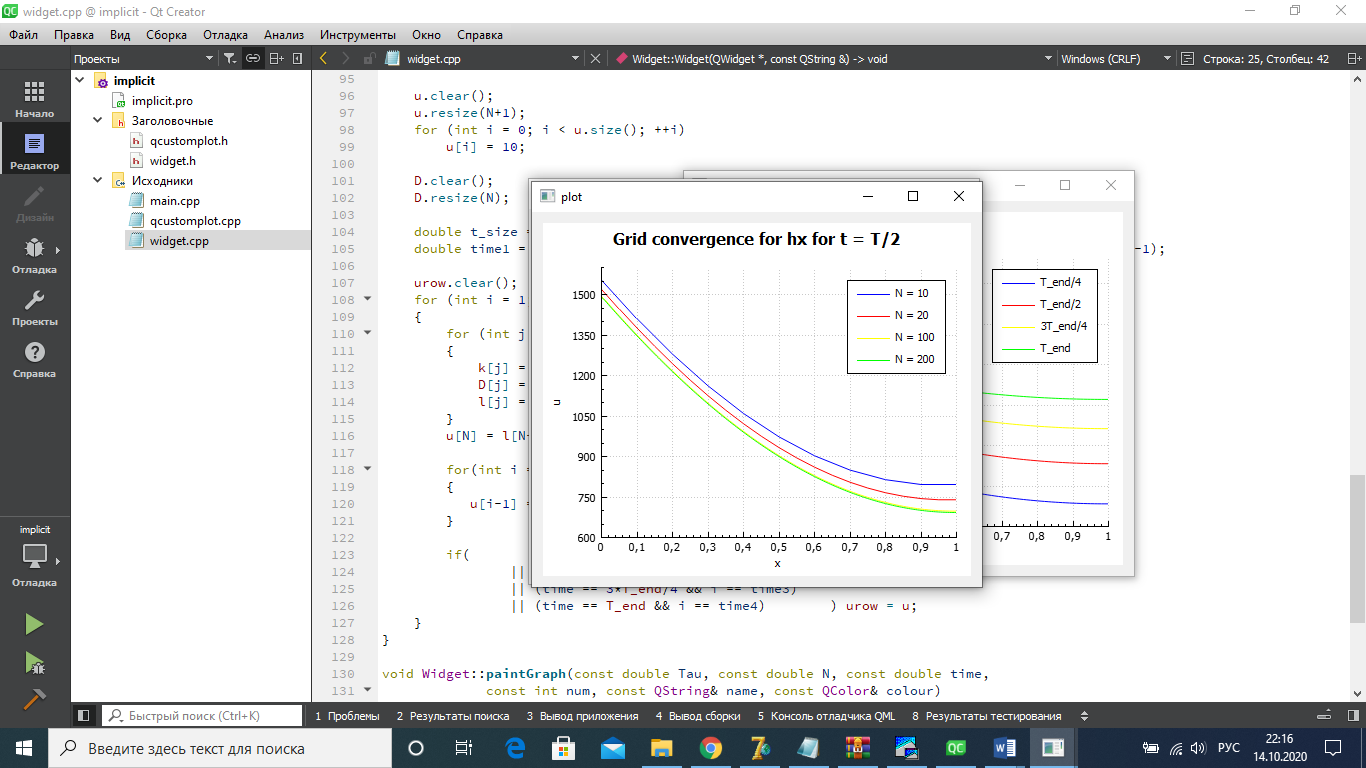


Рисунок 1 – Сеточная сходимость для шага по оси *х*

Видно, что при N = 100 и N = 200 кривая распределения температур практически сливается в одну линию - следовательно, количество разбиений сетки по оси *х* свыше 100 не имеет смысла. То есть, при N = 100 достигнута сеточная сходимость.

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

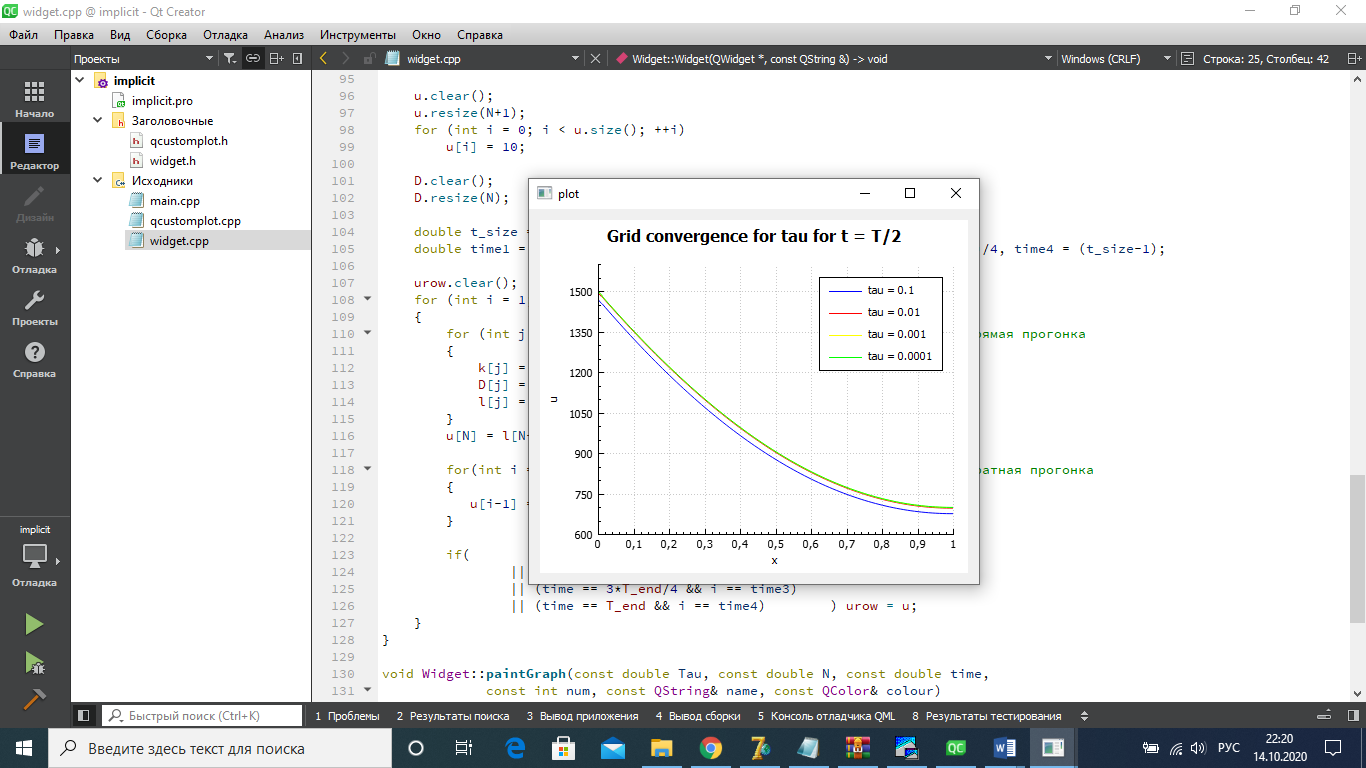


Рисунок 2 – Сеточная сходимость для шага по времени

Видно, что при = 0.01, и при более мелких , кривая распределения температур практически сливается в одну линию - 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.

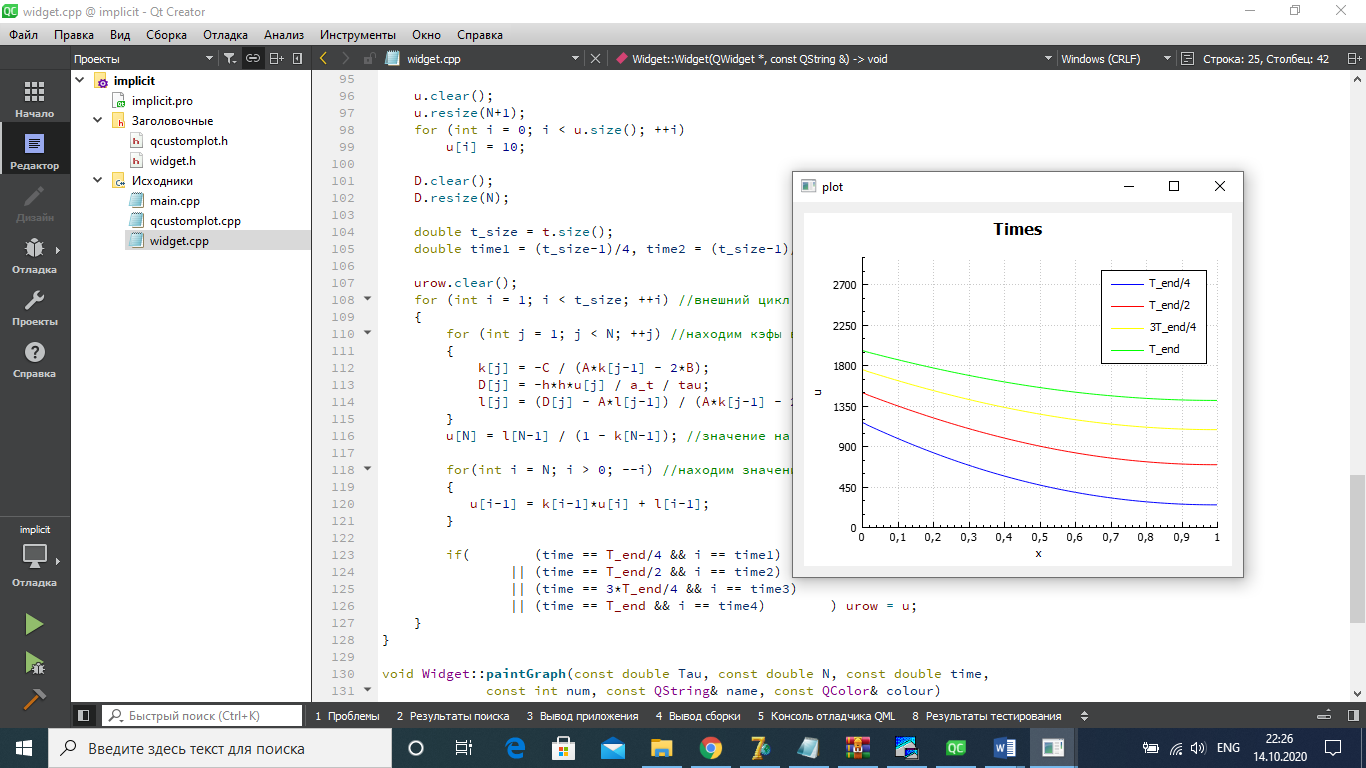


Рисунок 3 – Распределение температур по длине стенки во времени

*Вывод:* можно увидеть, что с течением времени температура по длине стенки становится более однородной. Результат расчета по неявной разностной схеме совпал с результатом расчета по явной схеме.

*Достоинство неявных схем:* безусловная устойчивость.

*Недостатки:* сложность алгоритма, неявная схема не может быть распространена на трехмерный случай.

*Код программы*

*widget.h:*

#ifndef WIDGET\_H

#define WIDGET\_H

#include <QWidget>

#include "qcustomplot.h"

class **Widget** : public QWidget

{

Q\_OBJECT

public:

enum **typeOfWidget** {GRID\_H, GRID\_TAU, TIMES};

**Widget**(QWidget \*parent = nullptr, const typeOfWidget t = GRID\_H);

private:

QVector<double> t{}, x{}, u{}, udef{};

const double a\_t = 1, alpha = 1, T\_g = 3000;

const double t0 = 0, T\_end = 1;

double h, tau;

int picked\_N;

double picked\_tau;

const double x0 = 0, x\_end = 1;

const double A = 1, C = 1;

double B;

QVector<double> k{}, l{}, D{};

QCustomPlot\* plot;

void **setVectors**(const double Tau, const double N, const double time);

void **paintGraph**(const double Tau, const double N, const double time,

const int num, const QString& name, const QColor& colour);

void **title**(const QString& title);

};

#endif // WIDGET\_H

*widget.cpp:*

#include <QVBoxLayout>

#include "widget.h"

Widget::**Widget**(QWidget \*parent, const typeOfWidget t)

: QWidget(parent)

{

setWindowTitle("plot");

setMinimumSize(450, 375);

plot = new QCustomPlot;

plot->xAxis->setLabel("x");

plot->yAxis->setLabel("u");

//Ox Range

plot->xAxis->setRange(0, 1);

plot->xAxis->setAutoTickStep(false);

plot->xAxis->setTickStep(0.1);

if(t == GRID\_H)

{

paintGraph(0.01, 10, T\_end/2, 0, "N = 10", Qt::blue);

paintGraph(0.01, 20, T\_end/2, 1, "N = 20", Qt::red);

paintGraph(0.01, 100, T\_end/2, 2, "N = 100", Qt::yellow);

paintGraph(0.01, 200, T\_end/2, 3, "N = 200", Qt::green);

plot->yAxis->setRange(600, 1600);

title("Grid convergence for hx for t = T/2");

}

else if(t == GRID\_TAU)

{

picked\_N = 100;

paintGraph(0.1, picked\_N, T\_end/2, 0, "tau = 0.1", Qt::blue);

paintGraph(0.01, picked\_N, T\_end/2, 1, "tau = 0.01", Qt::red);

paintGraph(0.001, picked\_N, T\_end/2, 2, "tau = 0.001", Qt::yellow);

paintGraph(0.0001, picked\_N, T\_end/2, 3, "tau = 0.0001", Qt::green);

plot->yAxis->setRange(600, 1600);

title("Grid convergence for tau for t = T/2");

}

else if(t == TIMES)

{

picked\_N = 100;

picked\_tau = 0.01;

paintGraph(picked\_tau, picked\_N, T\_end/4, 0, "T\_end/4", Qt::blue);

paintGraph(picked\_tau, picked\_N, T\_end/2, 1, "T\_end/2", Qt::red);

paintGraph(picked\_tau, picked\_N, 3\*T\_end/4, 2, "3T\_end/4", Qt::yellow);

paintGraph(picked\_tau, picked\_N, T\_end, 3, "T\_end", Qt::green);

plot->yAxis->setRange(0, 3000);

title("Times");

}

plot->legend->setVisible(true);

plot->replot();

QVBoxLayout\* l = new QVBoxLayout(this);

l->addWidget(plot);

setLayout(l);

}

void Widget::**setVectors**(const double Tau, const double N, const double time)

{

h = x\_end/N;

tau = Tau;

B = 1 + h\*h/(2\*a\_t\*tau);

t.clear();

t = {t0};

for (int i = 1; t[i-1] < T\_end; ++i)

t.push\_back(t0 + tau\*i);

x.clear();

x.resize(N+1);

x[0] = x0;

for (int i = 1; i < x.size(); ++i)

x[i] = x[i-1] + h;

// x = {x0};

// for (int i = 1; x[i-1] < x\_end; ++i)

// x.push\_back(x0 + h\*i);

k.clear();

k.resize(N);

k[0] = a\_t/h / (alpha + a\_t/h);

l.clear();

l.resize(N);

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

u.clear();

u.resize(N+1);

for (int i = 0; i < u.size(); ++i)

u[i] = 10;

D.clear();

D.resize(N);

double t\_size = t.size();

double time1 = (t\_size-1)/4, time2 = (t\_size-1)/2, time3 = 3\*(t\_size-1)/4, time4 = (t\_size-1);

udef.clear();

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

{

for (int j = 1; j < N; ++j) //находим кэфы во внутренних узлах - прямая прогонка

{

k[j] = -C / (A\*k[j-1] - 2\*B);

D[j] = -h\*h\*u[j] / a\_t / tau;

l[j] = (D[j] - A\*l[j-1]) / (A\*k[j-1] - 2\*B);

}

u[N] = l[N-1] / (1 - k[N-1]); //значение на правой границе

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

{

u[i-1] = k[i-1]\*u[i] + l[i-1];

}

if( (time == T\_end/4 && i == time1)

|| (time == T\_end/2 && i == time2)

|| (time == 3\*T\_end/4 && i == time3)

|| (time == T\_end && i == time4) ) udef = u;

}

}

void Widget::**paintGraph**(const double Tau, const double N, const double time,

const int num, const QString& name, const QColor& colour)

{

setVectors(Tau, N, time);

plot->addGraph();

plot->graph(num)->setName(name);

plot->graph(num)->setData(x, udef);

plot->graph(num)->setPen(QPen(colour));

}

void Widget::**title**(const QString& title)

{

plot->plotLayout()->insertRow(0);

plot->plotLayout()->addElement(0, 0, new QCPPlotTitle(plot, title));

}

*main.cpp:*

#include "widget.h"

#include <QApplication>

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

{

QApplication a(*argc*, argv);

Widget grid\_h(nullptr, "grid h");

grid\_h.show();

Widget grid\_tau(nullptr, "grid tau");

grid\_tau.show();

Widget times(nullptr, "times");

times.show();

return a.exec();

}