МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования «Национальный
исследовательский университет «МИЭТ»

Кафедра: ВМ-1
Дисциплина: Параллельные вычисления в микроэлектронике
01.03.04 «Прикладная математика» (выпускающая кафедра ВМ-1)

Курсовая работа на тему: «Решение линейной начально-краевой задачи для одномерного уравнения теплопроводности»

Выполнил:

студент группы ПМ-41

Алимагадов Курбан Алимагадович

Преподаватель:

профессор, д.ф.-м.н.

Поляков Сергей Владимирович

Дата:

МИЭТ 2019

Содержание

1. Постановка математической задачи

Линейная начально-краевая задача для одномерного уравнения теплопроводности:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(k(x,t)\frac{\partial u}{\partial x}\right) - u, \qquad x \in [0,1], \qquad t > 0, \tag{1}$$

$$k(x,t) = 1 + 10\exp\left[-\frac{(x-0.5)^2}{a^2}\right](1 - e^{-\omega t}), \qquad a = 0.1, \qquad \omega = 20,$$

$$u(x,0) = 0, \qquad u(0,t) = 1 - e^{-\omega t}, \qquad u(1,t) = 0.$$

Решить задачу методом конечных разностей на равномерной сетке с помощью неявной схемы и алгоритма правой параллельной прогонки.

2. Описание численного метода решения

Для численного решения поставленной задачи воспользуемся методом конечных разностей. Введём двумерную координатную сетку по координате и времени:

$$\Omega = \bar{\omega}_x \times \bar{\omega}_t$$

$$\bar{\omega}_x = \left\{x_i = a + ih_x, \qquad i = 0, \dots, N_x, \qquad h_x = \frac{b-a}{N_x}\right\}$$

$$\bar{\omega}_t = \left\{t_j = j\tau, \qquad j = 0, \dots, N_t, \qquad \tau = \frac{t_{max}}{N_x}\right\}$$

Будем использовать неявную схему с весами:

$$\frac{y_i^{j+1} - y_i^j}{\tau} = \sigma\left[\frac{1}{\hbar_x}\left\{k_{i+\frac{1}{2}}\frac{\left(y_{i+1}^{j+1} - y_i^{j+1}\right)}{h_x} - k_{i-\frac{1}{2}}\frac{\left(y_i^{j+1} - y_{i-1}^{j+1}\right)}{h_x}\right\} + f_i^{j+1}\right] + \tag{2}$$

$$+(1-\sigma)\left[\frac{1}{\hbar_x}\left\{k_{i+\frac{1}{2}}\frac{\left(y_{i+1}^{j} - y_i^{j}\right)}{h_x} - k_{i-\frac{1}{2}}\frac{\left(y_i^{j} - y_{i+1}^{j}\right)}{h_x}\right\} + f_i^{j}\right],$$

$$k_{i\pm\frac{1}{2}} = \frac{2k_i k_{i\pm1}}{k_i + k_{i\pm1}}, \qquad \hbar_x = h_{x,}$$

$$i = 1, \dots, N_x - 1, \qquad 0 \le j \le N_{t,}$$

$$y_i^0 = g_0(x), \qquad i = 0, \dots, N_x,$$

$$y_0^{j+1} = g_1(t_{j+1}), \qquad y_{N_x}^{j+1} = g_2(t_{j+1}),$$

где $\sigma = 1$.

Выбор параметров сетки:

$$t_{max} \sim \max(\tau_0, \tau_1) * M, \qquad M = 5, 10, 15, \dots$$

$$g_t = \max_{0 \le i \le N_x} \left| \frac{y_i^{j+1} - y_i^j}{\tau y_i^j} \right| \le \varepsilon_t = 10^{-8}$$

$$\tau^* \le \min(\tau^{(f)}, \tau^{(\sigma)})$$

$$\tau^{(f)} = \frac{1}{L}$$

$$\tau^{(\sigma=1)} = \frac{0{,}5h_x}{\sqrt{\max(k)}}$$

$$\tau \le \tau^* = \min(\tau^{(f)}, \tau^{(\sigma)})$$

$$\tau = \frac{t_{max}}{N_t}$$

$$N_t^{(\sigma=1)} \sim N_x$$

В результате построения разностной схемы возникает алгебраическая задача. Получим систему уравнений с трёхдиагональной матрицей:

$$A_i y_{i-1} - C_i y_i + B_i y_{i+1} = -F_i, \qquad 1 \le i \le N_x - 1 \tag{3}$$

и условиями на границе:

$$-C_0 y_0 + B_0 y_1 = -F_0, \qquad -C_{N_x} y_{N_x} + A_{N_x} y_{N_x - 1} = -F_{N_x} \tag{4}$$

с ограничениями на коэффициенты A, B и C:

$$A_i, B_i \ge 0, \qquad C_i > 0, \qquad i = 0, \dots, N_x \tag{5}$$

$$C_i = A_i + B_i + D_i, \qquad D_i > 0, \qquad i = 0, \dots, N_x. \tag{6}$$

Для решения системы будем пользоваться алгоритмом правой прогонки:

$$\alpha_0 = \frac{B_0}{C_0}, \qquad \beta_0 = \frac{F_0}{C_0},$$

$$\alpha_i = \frac{B_i}{C_i - A_i \alpha_{i-1}}, \qquad \beta_i = \frac{F_i + A_i \beta_{i-1}}{C_i - A_i \alpha_{i-1}}, \qquad i = 1, \dots, N_x,$$

$$y_{N_x} = \beta_{N_x}, \qquad y_i = \alpha_i y_{i+1} + \beta_i, \qquad i = N_x - 1, \dots, 0.$$

3. Описание параллельного алгоритма решения

Введём равномерное разбиение множества номеров узлов сетки $I = \{0,1,\dots,N\}$ на связные подмножества $I_m = \left\{i_1^{(m)},\dots,i_2^{(m)}\right\}$ $(m = 0,\dots,p-1)$.

Процессор с номером m будет обрабатывать $(i_2^{(m)} - i_1^{(m)} + 1)$ точек. Решение на каждом процессоре с номером m $(0 < m < p-1)$ представим в виде линейной комбинации:

$$y_i^m = y_i^{(I,m)} + y_{i_1^{(m)}} y_i^{(III,m)} + y_{i_2^{(m)}} y_i^{(II,m)}, \tag{7}$$

функции $y_i^{(\alpha,m)}$ $(\alpha = I, II, III)$ полностью определены на множестве узлов I_m и представляют собой некоторый базис, а значения искомой функции на границе I_m – $y_{i_1^{(m)}}$ и $y_{i_2^{(m)}}$ – пока неизвестны. Во внутренних узлах I_m функция $y_i^{(I,m)}$ находится из уравнений (3), а функции $y_i^{(II,m)}$, $y_i^{(III,m)}$ из уравнений (3) с нулевой правой частью.

Граничные условия для $y_i^{(\alpha,m)}$ ставятся следующим образом:

$$\begin{aligned} y_{i_1^{(m)}}^{(I,m)} &= 0, \qquad y_{i_2^{(m)}}^{(I,m)} = 0; \\ y_{i_1^{(m)}}^{(II,m)} &= 0, \qquad y_{i_2^{(m)}}^{(II,m)} = 1; \\ y_{i_1^{(m)}}^{(III,m)} &= 1, \qquad y_{i_2^{(m)}}^{(I,m)} = 0. \end{aligned} \tag{8}$$

На нулевом и последнем процессорах можно использовать линейные комбинации:

$$y_i^{(0)} = y_i^{(I,0)} + y_{i_2^{(0)}} y_i^{(II,0)}, \tag{9}$$

$$y_i^{(p-1)} = y_i^{(I,p-1)} + y_{i_1^{(p-1)}} y_i^{(III,p-1)}. \tag{10}$$

Из левого граничного условия исходной задачи для базисных функций нулевого процесса получаем:

$$C_0 y_0^{(I,0)} - B_0 y_1^{(I,0)} = F_0, \qquad y_{i_2^{(0)}}^{(I,0)} = 0; \tag{11}$$

$$C_0 y_0^{(II,0)} - B_0 y_1^{(II,0)} = 0, \qquad y_{i_2^{(0)}}^{(II,0)} = 1. \tag{12}$$

Из правого граничного условия исходной задачи для базисных функций нулевого процесса получаем:

$$y_{i_1^{(p-1)}}^{(I,p-1)} = 0, \qquad C_{N_x} y_{N_x}^{(I,p-1)} - A_{N_x} y_{N_x-1}^{(I,p-1)} = F_{N_x}; \tag{13}$$

$$y_{i_1^{(p-1)}}^{(III,p-1)} = 1, \qquad C_{N_x} y_{N_x}^{(III,p-1)} - A_{N_x} y_{N_x-1}^{(III,p-1)} = 0. \tag{14}$$

Базисные функции при выполнении условий (5) и (6) удовлетворяют принципу максимума:

$$\left\| y^{(I,m)} \right\|_C \le \| D^{-1} F \|_C, \qquad 0 \le y^{(II,m)} \le 1, \qquad 0 \le y^{(III,m)} \le 1, \qquad m = 0, \dots, p-1 \tag{15}$$

$$0 \le y_i^{(II,m)} + y_i^{(III,m)} \le 1, \qquad \text{для всех } i \text{ и } m \tag{16}$$

Эти свойства обеспечивают устойчивость вычислений по формулам (7), (9), (10).

Для нахождения неизвестных значений искомой функции в граничных узлах подобластей I_m запишем исходные уравнения (3) в двух соседних точках, принадлежащих процессорам с номерами m и m+1:

$$A_{i_2^{(m)}} y_{i_2^{(m)}-1} - C_{i_2^{(m)}} y_{i_2^{(m)}} + B_{i_2^{(m)}} y_{i_2^{(m)}+1} = -F_{i_2^{(m)}},$$

$$A_{i_1^{(m+1)}} y_{i_1^{(m+1)}-1} - C_{i_1^{(m+1)}} y_{i_1^{(m+1)}} + B_{i_1^{(m+1)}} y_{i_1^{(m+1)}+1} = -F_{i_1^{(m+1)}}.$$

Учтём, что

$$y_{i_2^{(m)}-1} = y_{i_2^{(m)}-1}^{(I,m)} + y_{i_1^{(m)}} y_{i_2^{(m)}-1}^{(III,m)} + y_{i_2^{(m)}} y_{i_2^{(m)}-1}^{(II,m)}, \qquad y_{i_2^{(m)}+1} = y_{i_1^{(m+1)}},$$

$$y_{i_1^{(m+1)}-1} = y_{i_2^{(m)}}, \qquad y_{i_1^{(m+1)}+1} = y_{i_1^{(m+1)}+1}^{(I,m+1)} + y_{i_1^{(m+1)}} y_{i_1^{(m+1)}+1}^{(III,m+1)} + y_{i_2^{(m+1)}} y_{i_1^{(m+1)}+1}^{(II,m+1)},$$

и получим следующие соотношения:

$$\tilde{A}_{i_2^{(m)}} y_{i_1^{(m)}} - \tilde{C}_{i_2^{(m)}} y_{i_2^{(m)}} + \tilde{B}_{i_2^{(m)}} y_{i_2^{(m+1)}} = -\tilde{F}_{i_2^{(m)}},$$

$$\tilde{A}_{i_1^{(m+1)}} y_{i_2^{(m)}} - \tilde{C}_{i_1^{(m+1)}} y_{i_1^{(m+1)}} + \tilde{B}_{i_1^{(m+1)}} y_{i_2^{(m+1)}} = -\tilde{F}_{i_1^{(m+1)}},$$

С коэффициентами

$$\tilde{A}_{i_2^{(m)}} = A_{i_2^{(m)}} y^{(III,m)}_{i_2^{(m)}-1}, \qquad \tilde{B}_{i_2^{(m)}} = B_{i_2^{(m)}},$$

$$\tilde{C}_{i_2^{(m)}} = C_{i_2^{(m)}} - A_{i_2^{(m)}} y^{(II,m)}_{i_2^{(m)}-1}, \qquad \tilde{F}_{i_2^{(m)}} = F_{i_2^{(m)}} + A_{i_2^{(m)}} y^{(I,m)}_{i_2^{(m)}-1},$$

$$\tilde{A}_{i_1^{(m+1)}} = A_{i_1^{(m+1)}}, \qquad \tilde{B}_{i_1^{(m+1)}} = B_{i_1^{(m+1)}} y^{(II,m+1)}_{i_1^{(m+1)}+1},$$

$$\tilde{C}_{i_1^{(m+1)}} = C_{i_1^{(m+1)}} - B_{i_1^{(m+1)}} y^{(III,m+1)}_{i_1^{(m+1)}+1}, \qquad \tilde{F}_{i_1^{(m+1)}} = F_{i_1^{(m+1)}} + B_{i_1^{(m+1)}} y^{(I,m+1)}_{i_1^{(m+1)}+1}.$$

Для каждой пары процессоров, получим следующую систему из $2p-2$ уравнений для $2p-2$ неизвестных

$$\begin{gathered} \tilde{A}_i y_{i-1} - \tilde{C}_i y_i + \tilde{B}_i y_{i+1} = -\tilde{F}_i, \\ i \in \tilde{I} = \left\{ i_2^{(0)}, i_1^{(1)}, i_2^{(1)}, \dots, i_1^{(p-1)} \right\}, \end{gathered} \tag{17}$$

где под индексом $i \pm 1$ понимается переход к соответствующему соседнему элементу из множества $\tilde{I}$.

В граничных узлах $i_2^{(0)}$ и $i_1^{(p-1)}$ уравнения (17) принимают вид (4). Кроме того, в силу свойств (15), (16) коэффициенты новой “короткой” системы уравнений также удовлетворяют условиям принципа максимума вида (5)-(6). Таким образом, решение системы (17) существует и является единственным. Определив его методом обычной прогонки, можно с помощью формул (7), (9), (10) вычислить решение исходной задачи (3)-(4).

Последовательность действий в алгоритме параллельной прогонки:

1) На каждом процессоре с помощью алгоритма скалярной прогонки решаются три задачи для нахождения базисных функций $y_i^{(\alpha,m)}$.
2) Находятся коэффициенты для новой задачи относительно неизвестных $y_{i_1^{(m)}}$, $y_{i_2^{(m)}}$ *(m = 0,…,p – 1).* Эти коэффициенты пересылаются нулевому процессору.
3) Нулевой процессор осуществляет решение короткой системы уравнений (17) и рассылает полученные значения $y_{i_1^{(m)}}$, $y_{i_2^{(m)}}$ соответствующему процессору.

4) Получив эти данные, каждый процессор восстанавливает свою часть искомого решения по формулам (7), (9), (10). На этом процедура решения заканчивается.

4. Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <math.h>
#include "mycom.h"
#include "mynet.h"
#include "myio.h"
#include "myprog.h"
 int np, mp, nl, ier, lp;
 int mp_l, mp_r;
 char pname[MPI_MAX_PROCESSOR_NAME];
 char vname[10] = "ex13b2";
 char sname[20];
 MPI_Status status;
 union_t buf;
 double tick, t1, t2, t3;
 FILE *Fi = NULL;
 FILE *Fo = NULL;
 int nx, ntp, ntm, ntv;
 double xa, xb, xk, x0, r0, q0, u0, u1;
 double k1, k2, tau0, tau1, tmax, epst;
 double tv, u10, omg0, omg1, a = 0.1, omega = 20.0, gt;
```

```
double k(double x, double t);
double k(double x, double t) {
  double s1 = -omega * t;
  double s2 = - (x - 0.5) * (x - 0.5) / (a * a);
  return 1 + 10.0 * exp(s2) * (1 - exp(s1));
}

double f(double x, double t);
double f(double x, double t) {
  return 0.0;
}

double g0(double x);
double g0(double x) {
  return 0.0;
}

double g1(double t);
double g1(double t) {
  return 0.0;
}

double g2(double t);
double g2(double t) {
  double s1 = -omega * t;
  return 1 - exp(s1);
```

```
}

int main(int argc, char *argv[])
{
  int i, j, ii, i1, i2, nc, ncm, ncp, ncx;
  double hx, hx2, tau, gam, s0, s1, s2, s3;
  double *xx, *aa, *bb, *cc, *ff, *y0, *y1, *y2, *y3, *y4, *al;

  MyNetInit(&argc,&argv,&np,&mp,&nl,pname,&tick);

  fprintf(stderr,"Netsize: %d, process: %d, system: %s,
tick=%12le\n",np,mp,pname,tick);
  sleep(1);

  sprintf(sname,"%s.p%02d",vname,mp);
  ier = fopen_m(&Fo,sname,"wt");
  if (ier!=0) mpierr("Protocol file not opened",1);

  if (mp==0) {
    sprintf(sname,"%s.d",vname);
    ier = fopen_m(&Fi,sname,"rt");
    if (ier!=0) mpierr("Data file not opened",2);
    fscanf(Fi,"xa=%le\n",&xa);
    fscanf(Fi,"xb=%le\n",&xb);
    fscanf(Fi,"xk=%le\n",&xk);
    fscanf(Fi,"x0=%le\n",&x0);
    fscanf(Fi,"r0=%le\n",&r0);
    fscanf(Fi,"q0=%le\n",&q0);
```

```
  fscanf(Fi,"u0=%le\n",&u0);
  fscanf(Fi,"u1=%le\n",&u1);
  fscanf(Fi,"k1=%le\n",&k1);
  fscanf(Fi,"k2=%le\n",&k2);
  fscanf(Fi,"tau0=%le\n",&tau0);
  fscanf(Fi,"tau1=%le\n",&tau1);
  fscanf(Fi,"tmax=%le\n",&tmax);
  fscanf(Fi,"epst=%le\n",&epst);
  fscanf(Fi,"nx=%d\n",&nx);
  fscanf(Fi,"ntp=%d\n",&ntp);
  fscanf(Fi,"ntm=%d\n",&ntm);
  fscanf(Fi,"lp=%d\n",&lp);
  fclose_m(&Fi);
  if (argc>1) sscanf(argv[1],"%d",&nx);
  if (argc>2) sscanf(argv[2],"%d",&ntp);
  if (argc>3) sscanf(argv[3],"%d",&ntm);
}

if (np>1) {
  if (mp==0) {
    buf.ddata[0]  = xa;
    buf.ddata[1]  = xb;
    buf.ddata[2]  = xk;
    buf.ddata[3]  = x0;
    buf.ddata[4]  = r0;
    buf.ddata[5]  = q0;
    buf.ddata[6]  = u0;
```

```
  buf.ddata[7]  = u1;
  buf.ddata[8]  = k1;
  buf.ddata[9]  = k2;
  buf.ddata[10] = tau0;
  buf.ddata[11] = tau1;
  buf.ddata[12] = tmax;
  buf.ddata[13] = epst;
  buf.idata[28] = nx;
  buf.idata[29] = ntp;
  buf.idata[30] = ntm;
  buf.idata[31] = lp;
}
MPI_Bcast(buf.ddata,16,MPI_DOUBLE,0,MPI_COMM_WORLD);
if (mp>0) {
  xa   = buf.ddata[0];
  xb   = buf.ddata[1];
  xk   = buf.ddata[2];
  x0   = buf.ddata[3];
  r0   = buf.ddata[4];
  q0   = buf.ddata[5];
  u0   = buf.ddata[6];
  u1   = buf.ddata[7];
  k1   = buf.ddata[8];
  k2   = buf.ddata[9];
  tau0 = buf.ddata[10];
  tau1 = buf.ddata[11];
  tmax = buf.ddata[12];
```

```
   epst = buf.ddata[13];
   nx   = buf.idata[28];
   ntp  = buf.idata[29];
   ntm  = buf.idata[30];
   lp   = buf.idata[31];
  }
}

fprintf(Fo,"Netsize: %d, process: %d, system: %s, tick=%12le\n",
  np,mp,pname,tick);

fprintf(Fo,"xa=%le xb=%le xk=%le x0=%le r0=%le\n",xa,xb,xk,x0,r0);
fprintf(Fo,"q0=%le u0=%le u1=%le k1=%le k2=%le\n",q0,u0,u1,k1,k2);
fprintf(Fo,"tau0=%le tau1=%le tmax=%le epst=%le\n",tau0,tau1,tmax,epst);
fprintf(Fo,"nx=%d ntp=%d ntm=%d lp=%d\n",nx,ntp,ntm,lp);

t1 = MPI_Wtime();

u10 = u1 - u0;
hx = (xb-xa)/nx; hx2 = hx * hx;
tau = 0.5 * hx / sqrt(11.0);
tau = dmin(tau,1.0/q0); gam = tau / hx2;
s0 = dmin(tmax/tau,1000000000.0);
ntm = imin(ntm,(int)s0);

fprintf(Fo,"u10=%le\n",u10);
fprintf(Fo,"hx=%le tau=%le ntm=%d\n",hx,tau,ntm);
```

```
if (mp == 0) fprintf(stderr,"nx=%d hx=%le tau=%le ntm=%d\n",nx,hx,tau,ntm);

if (mp ==    0) mp_l = -1; else mp_l = mp - 1;
if (mp == np-1) mp_r = -1; else mp_r = mp + 1;

MyRange(np,mp,0,nx,&i1,&i2,&nc);
ncm = nc-1; ncp = 2*(np-1); ncx = imax(nc,ncp);

fprintf(Fo,"i1=%d i2=%d nc=%d\n",i1,i2,nc);

xx = (double*)(malloc(sizeof(double)*nc));
y0 = (double*)(malloc(sizeof(double)*nc));
y1 = (double*)(malloc(sizeof(double)*nc));

aa = (double*)(malloc(sizeof(double)*nc));
bb = (double*)(malloc(sizeof(double)*nc));
cc = (double*)(malloc(sizeof(double)*nc));
ff = (double*)(malloc(sizeof(double)*nc));
al = (double*)(malloc(sizeof(double)*ncx));

if (np>1) {
  y2 = (double*)(malloc(sizeof(double)*nc));
  y3 = (double*)(malloc(sizeof(double)*nc));
  y4 = (double*)(malloc(sizeof(double)*9*ncp));
}

for (i=0; i<nc; i++) xx[i] = xa + hx * (i1 + i);
```

```
 for (i=0; i<nc; i++) {
  ii = i1 + i;

  if ((ii==0) || (ii==nx)) {
   aa[i] = 0.0; bb[i] = 0.0; cc[i] = 1.0;
  }
  else {
   s0 = k(xx[i],0.0); s1 = k(xx[i]-hx,0.0); s2 = k(xx[i]+hx,0.0);
   aa[i] = gam * 2.0 * s0 * s1 / (s0 + s1);
   bb[i] = gam * 2.0 * s0 * s2 / (s0 + s2);
   cc[i] = 1.0 + aa[i] + bb[i];
  }
 }

 ntv = 0; tv = 0.0; gt = 1.0;

 for (i=0; i<nc; i++) y1[i] = g0(xx[i]);

// Time loop:

 do {
  ntv++; tv += tau;

  for (i=0; i<nc; i++) y0[i] = y1[i];

  for (i=0; i<nc; i++) {
```

```
    ii = i1 + i;

    if (ii==0)       ff[i] = g1(tv);
    else if (ii==nx) ff[i] = g2(tv);
    else             ff[i] = y0[i] - tau * y0[i];
  }

  if (np<2) ier = prog_right(nc,aa,bb,cc,ff,al,y1);
  else      ier = prog_rightpm(np,mp,nc,ntv,aa,bb,cc,ff,al,y1,y2,y3,y4);

  if (ier!=0) mpierr("Bad solution",1);

  if (ntv % ntp == 0) {
    gt = 0.0;
    for (i=0; i<nc; i++) {
      s0 = (y1[i]/y0[i]-1.0);
      gt = dmax(gt,dabs(s0));
    }
    gt = gt / tau;
    if (np>1) {
      s0=gt;
MPI_Allreduce(&s0,&gt,1,MPI_DOUBLE,MPI_MAX,MPI_COMM_WORLD);
    }
    if (mp == 0) {
      t2 = MPI_Wtime() - t1;
      fprintf(stderr,"ntv=%d tv=%le gt=%le tcpu=%le\n",ntv,tv,gt,t2);
    }
  }
```

```
  if (lp>0) {
    fprintf(Fo,"ntv=%d tv=%le gt=%le\n",ntv,tv,gt);
    for (i=0; i<nc; i++)
      fprintf(Fo,"i=%8d x=%12le y1=%12le\n",(i1+i),xx[i],y1[i]);
  }
 } while ((ntv<ntm) && (gt>epst));

 t1 = MPI_Wtime() - t1;

 sprintf(sname,"%s_%02d.dat",vname,np);
 OutFun1DP(sname,np,mp,nc,xx,y1);

 fprintf(Fo,"ntv=%d tv=%le gt=%le time=%le\n",ntv,tv,gt,t1);
 if (mp == 0) fprintf(stderr,"ntv=%d tv=%le gt=%le tcpu=%le\n",ntv,tv,gt,t1);

 ier = fclose_m(&Fo);

 MPI_Finalize();
 return 0;
}
```

5. Оценка точности решения

Проведена оценка точности решения в зависимости от шага сетки в момент времени t = 0.25. Воспользуемся тем, что для неявной схемы:

$$\tau^{(\sigma=1)} = \frac{0{,}5h_x}{\sqrt{\max(k)}}.$$

Следовательно:

$$\|y_h^\tau - u_h^\tau\|_c \le c(h^2+\tau),$$

$$\left\|y_{\frac{h}{2}}^\tau - u_{\frac{h}{2}}^\tau\right\|_c \le c\left(\frac{h^2}{4}+\frac{\tau}{2}\right),$$

$$\left\|y_{\frac{h}{4}}^\tau - u_{\frac{h}{4}}^\tau\right\|_c \le c\left(\frac{h^2}{16}+\frac{\tau}{4}\right),$$

Отсюда получим, что:

$$\Delta_1 = \left\|y_h^\tau - y_{\frac{h}{2}}^\tau\right\|_c \le c\left(h^2\left(1+\frac{1}{4}\right)+\tau\left(1+\frac{1}{2}\right)\right),$$

$$\Delta_2 = \left\|y_{\frac{h}{2}}^\tau - y_{\frac{h}{4}}^\tau\right\|_c \le c\left(\frac{h^2}{4}\left(1+\frac{1}{4}\right)+\frac{\tau}{2}\left(1+\frac{1}{2}\right)\right),$$

$$\Delta = \frac{\Delta_1}{\Delta_2} = 2\frac{h^2\left(1+\frac{1}{4}\right)+\tau\left(1+\frac{1}{2}\right)}{\frac{h^2}{2}\left(1+\frac{1}{4}\right)+\tau\left(1+\frac{1}{2}\right)} = 2+\frac{\frac{h^2}{2}\left(1+\frac{1}{4}\right)}{\frac{h^2}{2}\left(1+\frac{1}{4}\right)+\tau\left(1+\frac{1}{2}\right)}.$$

Так как $\tau \sim h$, то

$$\lim_{h\to 0}\frac{\frac{h^2}{2}\left(1+\frac{1}{4}\right)}{\frac{h^2}{2}\left(1+\frac{1}{4}\right)+\tau\left(1+\frac{1}{2}\right)} = 0,$$

следовательно, при маленьких значениях $h, \Delta \approx 2$.

В рамках проделанной работы было найдено численное решение задачи с разбиениями по координате x: 500 отрезков, 1000 отрезков, 2000 отрезков.

По полученному решению были найдены Δ_1, Δ_2 и Δ в момент времени t = 0.25.

Для удобства работы с большими массивами данных вычисления проводились в среде Matlab.

Код программы на языке Matlab:

```
function [i, x, y] = ReadTxt(fId,count)
for j = 1:count
i(j) = fscanf(fId,'i=       %f');
x(j) = fscanf(fId,'  x=%f');
y(j) = fscanf(fId,' y1=%f\n');
end
fclose(fId);
end
```

```
fId = fopen('E:\Документы\ДЗ\Параллельные вычисления в
микроэлектронике\500.txt', 'rt');
[i20, x20, y20] = ReadTxt(fId,501);
fId = fopen('E:\Документы\ДЗ\Параллельные вычисления в
микроэлектронике\1000.txt', 'rt');
[i40, x40, y40] = ReadTxt(fId,1001);
fId = fopen('E:\Документы\ДЗ\Параллельные вычисления в
микроэлектронике\2000.txt', 'rt');
[i80, x80, y80] = ReadTxt(fId,2001);
%%
x1 = x20;
x2 = x40(1:2:end);
x3 = x80(1:4:end);

y1 = y20;
y2 = y40(1:2:end);
y3 = y80(1:4:end);

delta1 = max(abs(y2 - y1));
delta2 = max(abs(y3 - y2));

delta = delta1 / delta2

log2(delta)
```

В результате расчётов были получены значения, приведённые в таблице № 1.

Таблица №1 Максимум разницы решений в общих узлах на сетках h, h/2, h/4 и Δ в момент времени t = 0.25.

$\Delta_1 = \left\| y_h^\tau - y_{\frac{h}{2}}^\tau \right\|$	$\Delta_2 = \left\| y_{\frac{h}{2}}^\tau - y_{\frac{h}{4}}^\tau \right\|$	$\Delta = \frac{\Delta_1}{\Delta_2}$
8.930000000001437e-05	4.480000000001150e-05	1.993303571428381

Отсюда находим что,

$$\log_2 \Delta = 0.995161443063481.$$

Таким образом, вычисленная оценка сходимости соответствует теоретической.

6. Оценка эффективности распараллеливания

При исходной задачи с помощью скалярного алгоритма прогонки приходится выполнять $Q_1 = C_1 N$ обобщённых арифметических операций. При решении задачи по изложенному параллельному алгоритму производится $Q_p = 3C_2 \frac{N}{p} + 2C_1 p$ операций. Константы C1 и C2 не зависят от N и p и близки по величине ($\frac{C_2}{C_1} \sim 1.2$).

Получим следующие формулы для оценки ускорения и эффективности

$$S_p^T = \frac{Q_1}{Q_p} = \frac{p}{\frac{3C_2}{C_1} + \frac{2p^2}{N}},$$

$$E_p^T = \frac{S_p^T}{p} * 100\% = \frac{1}{\frac{3C_2}{C_1} + \frac{2p^2}{N}} * 100\%.$$

При проведении вычислительного эксперимента и оценке ускорения и эффективности по времени используются следующие формулы:

$$S_p = \frac{t_1}{t_p},$$

$$E_p = \frac{S_p}{p} * 100\%$$

Расчёты проводились с разбиением области на N = 2000 отрезков для p = 1, 2, 4, 8, 16 процессоров.

Таблица №2 Время выполнения в зависимости от числа процессоров

Кол-во процессоров	1	2	4	8	16
Время выполнения расчётов	3.950744e+01	2.316421e+01	1.752563e+01	1.635500e+01	1.645126e+01

Результаты полученные экспериментально и значения теоретических оценок приведены в таблице № 3.

Таблица №3 Оценка эффективности распараллеливания

Кол-во процессоров	2	4	8	16
$\boldsymbol{S_p^T}$	0.555	1.106	2.183	4.149
$\boldsymbol{E_p^T}$	27.747	27.655	27.296	25.934
$\boldsymbol{S_p}$	1.706	2.254	2.416	2.401
$\boldsymbol{E_p}$	85.277	56.357	30.195	15.009

Проанализировав данные таблицы, можно сделать вывод, что при наибольшем ускорении $S_8 = 2.416$ эффективность нашей параллельной программы достигает величины $E_8 = 30.195$, когда количество процессоров равно 8. Рассчитанная эффективность близка по значению к теоретической оценке $E_8 \approx E_8^T = 27.296$. Дальнейшее увеличение числа процессоров при данном разбиении не имеет смысла, так как видно что $S_{16} < S_8$ и E_{16} почти в 2 раза меньше E_8. Также рассматривая теоретические оценки, можно заметить, что $E_{16}^T < E_8^T$.

7. Заключение

В рамках работы была решена линейная начально-краевая задача для одномерного уравнения теплопроводности методом правой параллельной прогонки. По результатам расчётов были найдены оценки сходимости численного решения, ускорения и эффективности расчётов в зависимости от числа процессоров. Найденные значения соответствует теоретическим оценкам.