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:**

Содержание

[1. Постановка математической задачи 3](#_Toc27923488)

[2. Описание численного метода решения 3](#_Toc27923489)

[3. Описание параллельного алгоритма решения 5](#_Toc27923490)

[4. Текст программы 8](#_Toc27923491)

[5. Оценка точности решения 17](#_Toc27923492)

[6. Оценка эффективности распараллеливания 20](#_Toc27923493)

[7. Заключение 21](#_Toc27923494)

## Постановка математической задачи

Линейная начально-краевая задача для одномерного уравнения теплопроводности:

|  |  |
| --- | --- |
|  | (1) |

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

## Описание численного метода решения

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

Будем использовать неявную схему с весами:

|  |  |
| --- | --- |
|  | (2) |

где

Выбор параметров сетки:

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

|  |  |
| --- | --- |
|  | (3) |

и условиями на границе:

|  |  |
| --- | --- |
|  | (4) |

с ограничениями на коэффициенты A, B и C:

|  |  |
| --- | --- |
|  | (5) |
|  | (6) |

Для решения системы будем пользоваться алгоритмом правой прогонки:

## Описание параллельного алгоритма решения

Введём равномерное разбиение множества номеров узлов сетки на связные подмножества .

Процессор с номером m будет обрабатывать точек. Решение на каждом процессоре с номером m представим в виде линейной комбинации:

|  |  |
| --- | --- |
|  | (7) |

функции полностью определены на множестве узлов *Im* и представляют собой некоторый базис, а значения искомой функции на границе *Im* – и – пока неизвестны. Во внутренних узлах *Im* функция находится из уравнений (3), а функции , из уравнений (3) с нулевой правой частью.

Граничные условия для ставятся следующим образом:

|  |  |
| --- | --- |
|  | (8) |

На нулевом и последнем процессорах можно использовать линейные комбинации:

|  |  |
| --- | --- |
|  | (9) |
|  | (10) |

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

|  |  |
| --- | --- |
|  | (11) |
|  | (12) |

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

|  |  |
| --- | --- |
|  | (13) |
|  | (14) |

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

|  |  |
| --- | --- |
|  | (15) |
|  | (16) |

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

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

|  |  |
| --- | --- |
|  |  |
|  |  |

Учтём, что

|  |  |
| --- | --- |
|  |  |

и получим следующие соотношения:

|  |  |
| --- | --- |
|  |  |
|  |  |

С коэффициентами

Для каждой пары процессоров, получим следующую систему из уравнений для неизвестных

|  |  |
| --- | --- |
|  | (17) |

где под индексом понимается переход к соответствующему соседнему элементу из множества

В граничных узлах и 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 *.*
2. Находятся коэффициенты для новой задачи относительно неизвестных *, (m = 0,…,p – 1).* Эти коэффициенты пересылаются нулевому процессору.
3. Нулевой процессор осуществляет решение короткой системы уравнений (17) и рассылает полученные значения *,*  соответствующему процессору.
4. Получив эти данные, каждый процессор восстанавливает свою часть искомого решения по формулам (7), (9), (10). На этом процедура решения заканчивается.

## 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;

}

## Оценка точности решения

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

Следовательно:

Отсюда получим, что:

Так как , то

следовательно, при маленьких значениях .

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 , и 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 в момент времени t = 0.25.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 8.930000000001437e-05 | 4.480000000001150e-05 | 1.993303571428381 |

Отсюда находим что,

Таким образом, вычисленная оценка сходимости соответствует теоретической.

## Оценка эффективности распараллеливания

При исходной задачи с помощью скалярного алгоритма прогонки приходится выполнять обобщённых арифметических операций. При решении задачи по изложенному параллельному алгоритму производится операций. Константы C1 и C2 не зависят от N и p и близки по величине ().

Получим следующие формулы для оценки ускорения и эффективности

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

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 |
|  | 0.555 | 1.106 | 2.183 | 4.149 |
|  | 27.747 | 27.655 | 27.296 | 25.934 |
|  | 1.706 | 2.254 | 2.416 | 2.401 |
|  | 85.277 | 56.357 | 30.195 | 15.009 |

Проанализировав данные таблицы, можно сделать вывод, что при наибольшем ускорении эффективность нашей параллельной программы достигает величины , когда количество процессоров равно 8. Рассчитанная эффективность близка по значению к теоретической оценке . Дальнейшее увеличение числа процессоров при данном разбиении не имеет смысла, так как видно что и почти в 2 раза меньше . Также рассматривая теоретические оценки, можно заметить, что .

## 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.