Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Ивановский государственный энергетический университет имени В. И. Ленина»

Отчет по лабораторной работе

Уравнение теплопроводности

 Выполнил: студент гр. 3-46

Тютюкин Е.С.

Проверила: Чернышева Л.П.

Иваново 2021

Задание

Написать программу на OpenMP, MPI и в последовательном виде, которая вычисляет распространение тепла:

*;*

u(x, 0) = phi (x),u (0, t) = psi1 (t),u(L, t) = psi2 (t)

phi (x) = cos(2x+0.19);

psi1 (t) = 0.932;

psi2 (t) = 0.1798t.

Математическая модель

Для решения данной задачи воспользуемся методом сеток.

Расчетные формулы для метода:

U1k+1 = ψ(tk+1);

Uik+1 = Uik + (a2τ/h2)(Ui-1k - 2Uik + Ui+1k), i = 2, 3, …, n-1;

Unk+1 = ψ(tk+1);

tk+1 = tk + τ;

(a2τ/h2) = r < ½;

Код программы(OpenMP)

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <omp.h>

#include <math.h>

#include <windows.h>

#include <iostream>

using namespace std;

double phi(double x) //Вычисляем значение функции в точке х

{

return cos(2 \* x + 0.19);

}

const double psi1 = 0.932; //Значение в левой краевой точке

double psi2(double t) //Значение в правой краевой точке

{

return 0.1798 \* t;

}

int main()

{

setlocale(LC\_ALL, "RUS");

//Объявление начальных значений

double a = 1.0, L = 300.0, x = 0.0, time = 0.0, tmax = 1.0, tau = 0.0001, h = 0.02;

double r = a \* tau / (h \* h); //Постоянный коэффициент

int n = (int)(L / h) + 1;

double\* u = new double[n];

double\* un = new double[n];

double tn = omp\_get\_wtime(); //Начальное время

u[0] = psi1; //Начальное значение в левой краевой точке

x += h; //Шаг по координате

for (int i = 1; i < n - 1; i++) //Задаем начальные значения

{

u[i] = phi(x);

x += h;

}

u[n - 1] = psi2(time); //Начальное значение в правой краевой точке

do {

printf(" \r Время расчёта %f", time); // Вывод текущего значения переменной time для отладки

#pragma omp parallel //Открываем параллельную секцию

{

#pragma omp for schedule(dynamic, 1)

for (int i = 1; i < n - 1; i++) //Вычисляем новые значения в точках

{

un[i] = u[i] + r \* (u[i - 1] - 2.0 \* u[i] + u[i + 1]);

}

#pragma omp for schedule(dynamic, 1)

for (int i = 1; i < n - 1; i++) //Переприсваиваем новые значения в массив u

{

u[i] = un[i];

}

}

time += tau; //Шаг по времени

u[0] = psi1;

u[n - 1] = psi2(time);

} while (time <= tmax);

double tk = omp\_get\_wtime(); //Конечное время

FILE\* f;

f = fopen("Temp\_res\_m.dat", "w");

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

fprintf(f, "u[%d] = %f\n", i, u[i]); //Вывод значений

fprintf(f, "\nВремя вычислений: %f\n", tk - tn); //Время выполнения

fclose(f);

delete[] u;

delete[] un;

system("pause");

return 0;

}

Код программы(MPI)

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <mpi.h>

#include <math.h>

double phi(double x) //Вычисляем значение функции в точке х

{

return cos(2 \* x + 0.19);

}

const double psi1 = 0.932; //Значение в левой краевой точке

double psi2(double t) //Значение в правой краевой точке

{

return 0.1798 \* t;

}

int main(int arc, char\*\* argv) {

//Определение начальных значений переменных

int size, rank, msgtag = 12;

double a = 1.0, L = 300.0, x = 0.0, time = 0.0, tmax = 1.0, tau = 0.0001, h = 0.02;

double r = a \* tau / (h \* h), upr, uu;

int N = (int)(L / h) + 1;

int n, l = 0, i, j;

//шапочка

if (MPI\_Init(&arc, &argv) != MPI\_SUCCESS) return 1;

if (MPI\_Comm\_size(MPI\_COMM\_WORLD, &size) != MPI\_SUCCESS) {

MPI\_Finalize();

return 2;

}

if (MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank) != MPI\_SUCCESS) {

MPI\_Finalize();

return 3;

}

MPI\_Status status;

//Распределение отрезков по процессам

int l1 = N / size;

int l2 = N % size;

int\* kol = new int[size];

for (i = 0; i < size; i++) kol[i] = l1;

if (l2) {

if (rank == size - 1) l1++;

l2--;

kol[size - 1]++;

}

if (l2) {

if (rank < l2) l1++;

for (i = 0; i < l2; i++) kol[i]++;

}

n = l1;

if ((!rank) || (rank == size - 1)) n++; else n += 2;

double\* u = new double[n];

if (rank) {

int sum = 0;

for (i = 0; i < rank; i++) sum += kol[i];

l = sum;

sum--;

x = h \* sum; //Вычисляем смещение для каждого промежутка

}

//Присваиваем начальные значения

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

u[i] = phi(x);

x += h;

}

if (!rank) u[0] = psi1;

if (rank == size - 1) u[n - 1] = psi2(time);

double tn = MPI\_Wtime(); //Начальное время

do {

upr = u[0]; //Запоминаем предыдущее значение

for (i = 1; i < n - 1; i++) {

uu = u[i];

u[i] += r \* (upr - 2.0 \* u[i] + u[i + 1]); //Вычисление новых значений в точках

upr = uu; //Запоминаем предыдущее значение

}

//Начальные значения в краевых точках

if (!rank) u[0] = psi1;

if (rank == size - 1) u[n - 1] = psi2(time + tau);

//Обмен данными между процессами

if (rank & 1) {

MPI\_Ssend(&u[1], 1, MPI\_DOUBLE, rank - 1, msgtag, MPI\_COMM\_WORLD);

MPI\_Recv(&u[0], 1, MPI\_DOUBLE, rank - 1, msgtag, MPI\_COMM\_WORLD, &status);

if (rank != size - 1) {

MPI\_Ssend(&u[n - 2], 1, MPI\_DOUBLE, rank + 1, msgtag, MPI\_COMM\_WORLD);

MPI\_Recv(&u[n - 1], 1, MPI\_DOUBLE, rank + 1, msgtag, MPI\_COMM\_WORLD, &status);

}

}

else {

if (rank != size - 1) {

MPI\_Recv(&u[n - 1], 1, MPI\_DOUBLE, rank + 1, msgtag, MPI\_COMM\_WORLD, &status);

MPI\_Ssend(&u[n - 2], 1, MPI\_DOUBLE, rank + 1, msgtag, MPI\_COMM\_WORLD);

}

if (rank) {

MPI\_Recv(&u[0], 1, MPI\_DOUBLE, rank - 1, msgtag, MPI\_COMM\_WORLD, &status);

MPI\_Ssend(&u[1], 1, MPI\_DOUBLE, rank - 1, msgtag, MPI\_COMM\_WORLD);

}

}

time += tau; //Шаг по времени

} while (time <= tmax);

double tk = MPI\_Wtime(); //Конечное время

//вывод

int in, ik;

if (!rank) in = 0; else in = 1;

if (rank == size - 1) ik = n; else ik = n - 1;

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

{

if (i == rank)

{

FILE\* f;

f = fopen("Temperature.txt", "a");

for (j = in; j < ik; j++, l++) {

fprintf(f, "u[%d] = %f\n", l, u[j]);

}

fclose(f);

}

MPI\_Barrier(MPI\_COMM\_WORLD);

}

if (!rank) {

FILE\* f;

f = fopen("Temperature.txt", "a");

fprintf(f, "\n¬рем¤ вычислений: %f", tk - tn);

fclose(f);

}

delete[]u;

delete[]kol;

//тапочки

MPI\_Finalize();

return 0;

}

Код последовательной программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <math.h>

#include <windows.h>

#include <iostream>

using namespace std;

double phi(double x) //Вычисляем значение функции в точке х

{

return cos(2 \* x + 0.19);

}

const double psi1 = 0.932; //Значение в левой краевой точке

double psi2(double t) //Значение в правой краевой точке

{

return 0.1798 \* t;

}

int main()

{

setlocale(LC\_ALL, "RUS");

//Объявление начальных значений

double a = 1.0, L = 300.0, x = 0.0, time = 0.0, tmax = 1.0, tau = 0.0001, h = 0.02;

double r = a \* tau / (h \* h); //Постоянный коэффициент

int n = (int)(L / h) + 1;

double\* u = new double[n];

double\* un = new double[n];

double tn = omp\_get\_wtime(); //Начальное время

u[0] = psi1; //Начальное значение в левой краевой точке

x += h; //Шаг по координате

for (int i = 1; i < n - 1; i++) //Задаем начальные значения

{

u[i] = phi(x);

x += h;

}

u[n - 1] = psi2(time); //Начальное значение в правой краевой точке

do {

printf(" \r Время расчёта %f", time); // Вывод текущего значения переменной time для отладки

{

for (int i = 1; i < n - 1; i++) //Вычисляем новые значения в точках

{

un[i] = u[i] + r \* (u[i - 1] - 2.0 \* u[i] + u[i + 1]);

}

for (int i = 1; i < n - 1; i++) //Переприсваиваем новые значения в массив u

{

u[i] = un[i];

}

}

time += tau; //Шаг по времени

u[0] = psi1;

u[n - 1] = psi2(time);

} while (time <= tmax);

double tk = omp\_get\_wtime(); //Конечное время

FILE\* f;

f = fopen("Temp\_res\_m.dat", "w");

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

fprintf(f, "u[%d] = %f\n", i, u[i]); //Вывод значений

fprintf(f, "\nВремя вычислений: %f\n", tk - tn); //Время выполнения

fclose(f);

delete[] u;

delete[] un;

system("pause");

return 0;

}

Результаты

u[0] = 0.932000

u[1] = 0.919071

u[2] = 0.906147

u[3] = 0.893229

u[4] = 0.880323

u[5] = 0.867432

u[6] = 0.854559

u[7] = 0.841709

u[8] = 0.828885

u[9] = 0.816091

…

u[14991] = 0.165632

u[14992] = 0.166968

u[14993] = 0.168359

u[14994] = 0.169807

u[14995] = 0.171315

u[14996] = 0.172884

u[14997] = 0.174517

u[14998] = 0.176215

u[14999] = 0.177982

u[15000] = 0.179818

Время вычислений MPI: 0.355327

Время вычислений последовательных: 17,581615

Время вычислений OpenMP: 17,205643

|  |  |  |
| --- | --- | --- |
|  | **Время выполнения** | **Ускорение** |
| MPI | 0.355327 | 49,5239 |
| Open MP | 17,205643 | 1,0218 |
| Последовательная | 17,581615 | 1 |

Вывод:

была написана программа для решения уравнения теплопроводности с помощью явной схемы. Наиболее быстрые вычисления были выполнены с помощью технологий MPI.