Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

ОТЧЕТ

по курсу
“Суперкомпьютерное моделирование и технологии”

Вариант — 3

Работу выполнил:
студент 628 группы
Сазонов Георгий Владимирович

Математическая постановка задачи

В трёхмерной замкнутой области

$$\Omega = [0 \le x \le L_x] \times [0 \le y \le L_y] \times [0 \le z \le L_z]$$

для $0 < t \le T$ требуется найти решение $u(x, y, z, t)$ уравнения в частных производных

$$\frac{\partial^2 u}{\partial t^2} = a^2 \Delta u, \tag{1}$$

с начальными условиями

$$u|_{t=0} = \varphi(x, y, z), \tag{2}$$

$$\left.\frac{\partial u}{\partial t}\right|_{t=0} = 0, \tag{3}$$

при условии, что на границах области заданы либо однородные граничные условия первого рода

$$u(0, y, z, t) = 0, \qquad u(L_x, y, z, t) = 0, \tag{4}$$

$$u(x, 0, z, t) = 0, \qquad u(x, L_y, z, t) = 0, \tag{5}$$

$$u(x, y, 0, t) = 0, \qquad u(x, y, L_z, t) = 0, \tag{6}$$

либо периодические граничные условия

$$u(0, y, z, t) = u(L_x, y, z, t), \qquad u_x(0, y, z, t) = u_x(L_x, y, z, t), \tag{7}$$

$$u(x, 0, z, t) = u(x, L_y, z, t), \qquad u_y(x, 0, z, t) = u_y(x, L_y, z, t), \tag{8}$$

$$u(x, y, 0, t) = u(x, y, L_z, t), \qquad u_z(x, y, 0, t) = u_z(x, y, L_z, t). \tag{9}$$

Конкретная комбинация граничных условий определяется индивидуальным вариантом задания.

Для численного решения задачи введём на Ω сетку $\omega_{h\tau} = \bar{\omega}_h \times \omega_\tau$, где

$$T = T_0, \quad L_x = L_{x0}, \quad L_y = L_{y0}, \quad L_z = L_{z0},$$

$$\bar{\omega}_h = \{(x_i = ih_x,\ y_j = jh_y,\ z_k = kh_z) \mid i, j, k = 0, 1, \ldots, N,\ h_x N = L_x,\ h_y N = L_y,\ h_z N = L_z\},$$

$$\omega_\tau = \{t_n = n\tau \mid n = 0, 1, \ldots, K,\ \tau K = T\}.$$

Через ω_h обозначим множество внутренних, а через γ_h — множество граничных узлов сетки $\bar{\omega}_h$.

Для аппроксимации исходного уравнения (1) с однородными граничными условиями (4)–(6) и начальными условиями (2)–(3) воспользуемся следующей системой уравнений:

$$\frac{u^{n+1}_{ijk} - 2u^n_{ijk} + u^{n-1}_{ijk}}{\tau^2} = a^2 \Delta_h u^n_{ijk}, \quad (x_i, y_j, z_k) \in \omega_h,\ n = 1, 2, \ldots, K-1,$$

где Δ_h — семиточечный разностный аналог оператора Лапласа:

$$\Delta_h u^n_{ijk} = \frac{u^n_{i-1,j,k} - 2u^n_{i,j,k} + u^n_{i+1,j,k}}{h_x^2} + \frac{u^n_{i,j-1,k} - 2u^n_{i,j,k} + u^n_{i,j+1,k}}{h_y^2} + \frac{u^n_{i,j,k-1} - 2u^n_{i,j,k} + u^n_{i,j,k+1}}{h_z^2}.$$

Приведённая выше разностная схема является явной — значения u^{n+1}_{ijk} на $(n+1)$-м временном слое можно явным образом выразить через значения на предыдущих слоях.

Для начала счёта (т. е. для нахождения u^1_{ijk}) должны быть заданы значения u^0_{ijk}, u^1_{ijk}, $(x_i, y_j, z_k) \in \omega_h$. Из условия (2) имеем

$$u^0_{ijk} = \varphi(x_i, y_j, z_k), \quad (x_i, y_j, z_k) \in \omega_h. \tag{10}$$

Простейшая замена начального условия (3) уравнением $(u^1_{ijk} - u^0_{ijk})/\tau = 0$ имеет лишь первый порядок аппроксимации по τ. Аппроксимацию второго порядка по τ и h даёт разностное уравнение

$$\frac{u^1_{ijk} - u^0_{ijk}}{\tau} = \frac{a^2\tau}{2}\Delta_h\varphi(x_i, y_j, z_k),$$

откуда

$$u^1_{ijk} = u^0_{ijk} + \frac{a^2\tau^2}{2}\Delta_h\varphi(x_i, y_j, z_k). \tag{11}$$

Разностная аппроксимация для периодических граничных условий выглядит следующим образом:

$$\begin{aligned}
&u^{n+1}_{0jk} = u^{n+1}_{Njk}, \quad u^{n+1}_{1jk} = u^{n+1}_{N+1,jk}, \\
&u^{n+1}_{i0k} = u^{n+1}_{iNk}, \quad u^{n+1}_{i1k} = u^{n+1}_{i,N+1,k}, \quad i,j,k = 0,1,\ldots,N. \\
&u^{n+1}_{ij0} = u^{n+1}_{ijN}, \quad u^{n+1}_{ij1} = u^{n+1}_{ij,N+1},
\end{aligned}$$

Для вычисления значений $u^0, u^1 \in \gamma_h$ допускается использование аналитического значения u, которое задаётся в программе также для вычисления погрешности решения задачи.

Специфика для Варианта №3

Граничные условия:

- по x: 1-го рода (нулевые)
- по y: периодические
- по z: 1-го рода (нулевые)

Аналитическое решение:

$$u_{\text{analytical}}(x, y, z, t) = \sin\left(\frac{\pi x}{L_x}\right) \cdot \sin\left(\frac{2\pi y}{L_y}\right) \cdot \sin\left(\frac{3\pi z}{L_z}\right) \cdot \cos(a_t t),$$

где

$$a_t = \frac{\pi}{2}\sqrt{\frac{1}{L_x^2} + \frac{4}{L_y^2} + \frac{9}{L_z^2}}, \quad a^2 = \frac{1}{4}.$$

Программная реализация

Реализация задачи состоит из файлов: `main.cpp`, `solution.cpp`, `equation.hpp` и `Makefile`, `run_polus.sh`.

Файл `main.cpp` содержит функцию `main`, которая обрабатывает входные параметры, инициализирует вычисления и сохраняет результаты. В нём также реализованы следующие вспомогательные функции:

- `save_results` — сохраняет в формате CSV либо численное решение, либо аналитическое значение, либо абсолютную погрешность $|u_{\text{численное}} - u_{\text{аналитическое}}|$

- `save_statistics` — записывает в текстовый файл время выполнения, максимальную погрешность за все временные шаги, а также погрешности на первом и последнем шагах.

- `parse_length` - позволяет задавать границы вычислений либо числами, либо как "pi"

Программа принимает на вход пять аргументов:

- N — количество ячеек по каждой пространственной координате (итоговый размер сетки составляет $(N+1)^3$);

- `threads_num` — число потоков OpenMP;

- L_x, L_y, L_z — размеры расчётной области по осям. Каждый из параметров может быть задан либо числом (например, `1.0`), либо строкой `pi`, которая интерпретируется как π

Файл `equation.hpp` содержит объявления используемых типов данных, а также класс `Grid`, описывающий геометрию расчётной сетки. Он хранит значения L_x, L_y, L_z, шаги h_x, h_y, h_z, временной шаг $\tau = 10^{-4}$ и предоставляет метод `index(i,j,k)` для преобразования трёхмерных индексов в линейный адрес одномерного массива. Также в этом файле определена функция `u_analytical`, реализующая точное решение для варианта №3:

$$u(x,y,z,t) = \sin\left(\frac{\pi x}{L_x}\right) \cdot \sin\left(\frac{2\pi y}{L_y}\right) \cdot \sin\left(\frac{3\pi z}{L_z}\right) \cdot \cos(a_t t),$$

где

$$a_t = \frac{\pi}{2}\sqrt{\frac{1}{L_x^2} + \frac{4}{L_y^2} + \frac{9}{L_z^2}}, \quad a^2 = \frac{1}{4}.$$

Значение $a^2 = \frac{1}{4}$ также хранится в классе Grid.

Основной вычислительный алгоритм реализован в файле `solution.cpp` и основан на явной конечно-разностной схеме второго порядка точности:

- `laplace_operator` — вычисляет семиточечный аналог оператора Лапласа $\Delta_h u$. Для оси y применяется периодическая индексация (в соответствии с граничными условиями (8)), тогда как для осей x и z используются обычные соседние узлы, поскольку заданы однородные условия первого рода (формулы (4) и (6));

- `init` — вычисляет начальные слои u^0 и u^1 по формулам (10) и (12):

$$u^0_{ijk} = \varphi(x_i, y_j, z_k), \quad u^1_{ijk} = u^0_{ijk} + \frac{a^2\tau^2}{2}\Delta_h\varphi(x_i, y_j, z_k),$$

  с учётом $a^2 = 1/4$. Граничные узлы устанавливаются в соответствии с типом условий: нулевые значения для x и z, периодическое продолжение для y;

- `run_algo` — выполняет 19 итераций (всего 20 временных шагов), обновляя решение по схеме

$$u^{n+1}_{ijk} = 2u^n_{ijk} - u^{n-1}_{ijk} + a^2\tau^2\Delta_h u^n_{ijk}.$$

  После каждого шага вычисляется максимальная абсолютная погрешность относительно аналитического решения;

- `solve_equation` — управляет ходом вычислений: устанавливает число потоков OpenMP, отключает динамическое распределение, инициализирует данные, запускает алгоритм и измеряет время выполнения с помощью `omp_get_wtime()`.

Параллелизация реализована с использованием технологии OpenMP. Все циклы по сетке снабжены директивой `#pragma omp parallel for`, а при вычислении максимальной погрешности используется редукция `reduction(max : ...)`. На текущем этапе реализована только OpenMP-версия; гибридная MPI+OpenMP реализация с блочным разбиением области предполагается в дальнейшем.

Файл `Makefile` обеспечивает сборку программы под суперкомпьютер IBM Polus. Запуск вычислительных экспериментов осуществляется через внешний скрипт `run_polus.sh` и включает:

- размеры сетки $N = 128, 256, 512$;
- два набора геометрических параметров: $L_x = L_y = L_z = 1$ и $L_x = L_y = L_z = \pi$;
- число потоков: 1, 2, 4, 8, 16, 32.

Результаты расчётов

Таблица 1: Результаты расчётов для сетки с $L = 1$ (OpenMP)

Число OpenMP нитей	Число точек сетки по одной оси	Время решения, с	Ускорение	Погрешность
1	128	7,06	1	3,42e-4
2	128	3,67	1,92	3,42e-4
4	128	1,95	3,62	3,42e-4
8	128	1,05	6,72	3,42e-4
16	128	0,87	8,11	3,42e-4
32	128	0,67	10,54	3,42e-4
1	256	53,5	1	6,78e-4
2	256	27,56	1,94	6,78e-4
4	256	14,63	3,66	6,78e-4
8	256	7,64	7	6,78e-4
16	256	5,94	9,01	6,78e-4
32	256	4,37	12,24	6,78e-4
1	512	414,19	1	1,3e-3
2	512	212,76	1,95	1,3e-3
4	512	111,06	3,73	1,3e-3
8	512	56,88	7,28	1,3e-3
16	512	41,69	9,93	1,3e-3
32	512	32,81	12,62	1,3e-3

Таблица 2: Результаты расчётов для сетки с $L = \pi$ (OpenMP)

Число OpenMP нитей	Число точек сетки по одной оси	Время решения, с	Ускорение	Погрешность
1	128	7,06	1	3,48e-5
2	128	3,68	1,92	3,48e-5
4	128	1,96	3,60	3,48e-5
8	128	1,05	6,72	3,48e-5
16	128	0,82	8,61	3,48e-5
32	128	0,66	10,7	3,48e-5
1	256	53,57	1	6,96e-5
2	256	27,61	1,94	6,96e-5
4	256	14,46	3,70	6,96e-5
8	256	7,68	6,98	6,96e-5
16	256	5,85	9,16	6,96e-5
32	256	4,38	12,23	6,96e-5
1	512	414,25	1	1,38e-4
2	512	213,07	1,94	1,38e-4
4	512	110,38	3,75	1,38e-4
8	512	57,67	7,18	1,38e-4
16	512	41,80	9,91	1,38e-4
32	512	32,71	12,66	1,38e-4