

# **Отчёт по заданию курса**

## **«Суперкомпьютерное моделирование и технологии»**

**Моисеев Дмитрий Александрович**

Октябрь 2025 – Декабрь 2025

### **Содержание**

<b>1</b>	<b>Математическая постановка задачи</b>	<b>2</b>
<b>2</b>	<b>Численный метод решения</b>	<b>2</b>
<b>3</b>	<b>Программная реализация (OpenMP)</b>	<b>3</b>
<b>4</b>	<b>Результаты OpenMP-версии</b>	<b>4</b>
<b>5</b>	<b>Заключение OpenMP</b>	<b>4</b>

# 1 Математическая постановка задачи

В трёхмерной замкнутой области

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

для  $t \in (0, T]$  требуется найти решение волнового уравнения:

$$\frac{\partial^2 u}{\partial t^2} = a^2 \Delta u.$$

Начальные условия:

$$u(x, y, z, 0) = \varphi(x, y, z), \quad \frac{\partial u}{\partial t}(x, y, z, 0) = 0.$$

Граничные условия (вариант 3:  $x$  — 1-й род,  $y$  — периодические,  $z$  — 1-й род).

$$\begin{aligned} u(0, y, z, t) &= u(L_x, y, z, t) = 0, \\ u(x, 0, z, t) &= u(x, L_y, z, t), \\ u(x, y, 0, t) &= u(x, y, L_z, t) = 0. \end{aligned}$$

Аналитическое решение:

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

где

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

## 2 Численный метод решения

Используется равномерная сетка:

$$x_i = ih_x, \quad y_j = jh_y, \quad z_k = kh_z, \quad t_n = n\tau,$$

где  $h_x = L_x/N$ ,  $h_y = L_y/N$ ,  $h_z = L_z/N$ .

Разностная схема:

$$\frac{u_{i,j,k}^{n+1} - 2u_{i,j,k}^n + u_{i,j,k}^{n-1}}{\tau^2} = a^2 \Delta_h u_{i,j,k}^n,$$

где семиточечный аналог оператора Лапласа:

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

Для начальных слоёв:

$$u_{i,j,k}^0 = \varphi(x_i, y_j, z_k), \quad u_{i,j,k}^1 = u_{i,j,k}^0 + \frac{a^2 \tau^2}{2} \Delta_h \varphi(x_i, y_j, z_k).$$

### 3 Программная реализация (OpenMP)

Программа написана на C++11. Алгоритм решения задачи выглядит так:

1. **Задание геометрии и параметров:** выбрать  $N$ ,  $L_x = L_y = L_z = L$ ,  $T$ ,  $N_t$ , скорость  $a$  (для  $a^2 = 1/4$  берём  $a = 0.5$ ); шаги  $h_x = L_x/N$  (аналогично  $h_y, h_z$ ),  $\tau = T/N_t$ . Проверить условие устойчивости (CFL) При увеличении  $N$  пропорционально увеличивать  $N_t$  для сохранения CFL:

$$a \tau \sqrt{\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2}} \leq 1.$$

2. **Выделение памяти:** создать три массива  $(N+1)^3$  двойной точности для слоёв времени в кольцевом буфере  $u[3]$ ; индексация  $\text{idx}(i, j, k, N) = i(N+1)^2 + j(N+1) + k$ .
3. **Инициализация  $u^0$ :**  $u_{ijk}^0 = u_{\text{ан}}(x_i, y_j, z_k, 0)$ . Применить граничные условия (см. п. 5b).
4. **Инициализация  $u^1$  (2-й порядок по времени):**

$$u_{ijk}^1 = u_{ijk}^0 + \frac{a^2 \tau^2}{2} \Delta_h u_{\text{ан}}(x_i, y_j, z_k, 0),$$

где  $\Delta_h$  — семиточечный лапласиан. Применить граничные условия.

5. **Основной цикл по времени** для  $n = 1, \dots, N_t - 1$ :

- (a) **Обновление внутренних узлов**  $1 \leq i, j, k \leq N - 1$  (явная схема):

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

- (b) **Граничные условия (вариант 3):**

по  $y$  — *периодика* (призрачные слои):  $u_{i,0,k} = u_{i,N-1,k}$ ,  $u_{i,N,k} = u_{i,1,k}$ ;

по  $x$  и  $z$  — *Дирихле*  $u = 0$  на  $i \in \{0, N\}$  и  $k \in \{0, N\}$ .

Порядок: сначала периодика по  $y$ , затем обнуление по  $x, z$ .

- (c) **Диагностика:** вычислить погрешности

6. **Вывод результатов:** время работы, значения погрешностей.

Распараллеливание (OpenMP). Основные тройные циклы распараллелены:

```
// обновление слоя
#pragma omp parallel for collapse(3)
for (int i=1; i<N; ++i) for (int j=1; j<N; ++j) for (int k=1; k<N; ++k) { ... }
// метрики ошибок
#pragma omp parallel for collapse(3) reduction(max:err_inf) { ... }
```

Число потоков задаётся переменной окружения OMP\_NUM\_THREADS.

Вычисление погрешности:

$$\varepsilon = \max_{i,j,k} |u_{i,j,k}^n - u_{\text{аналит}}(x_i, y_j, z_k, t_n)|.$$

## 4 Результаты OpenMP-версии

### Вариант 3

Эксперименты проводились на вычислительном кластере IBM Polus. Размеры сетки:  $N = 128, 256$ .  $L = 1, T = 0.001, N_t = 20$  Время выполнения усреднялось по 5 запускам для каждой конфигурации.

№ потоков	$N^3$	$T$ (сек)	$S$ (ускорение)	$\delta$ (погрешность)
1	$128^3$	4.635001	1.00	1e-04
2	$128^3$	2.522893	1.84	1e-04
4	$128^3$	1.240660	3.74	1e-04
8	$128^3$	0.928058	4.99	1e-04
16	$128^3$	0.813130	5.70	1e-04
32	$128^3$	0.554325	8.36	1e-04
1	$256^3$	32.45046	1.00	2e-04
2	$256^3$	17.041407	1.90	2e-04
4	$256^3$	10.245951	3.17	2e-04
8	$256^3$	6.239249	5.20	2e-04
16	$256^3$	4.918441	6.60	2e-04
32	$256^3$	3.356135	9.67	2e-04

## 5 Заключение OpenMP

Увеличение количества потоков приводит к существенному сокращению времени вычислений. Полученное ускорение близко к линейному до 8 потоков, что подтверждает эффективность распараллеливания. Отклонение от идеального ускорения объясняется накладными расходами на управление потоками и ограничениями подсистемы памяти (особенно заметными при 16–32 потоках).

Для более крупной задачи ( $256^3$ ) масштабирование лучше на всём диапазоне потоков: ускорение достигает порядка  $S \approx 16$  на 32 потоках (эффективность около 50%), а на 2–8 потоках наблюдается умеренное суперлинейное ускорение за счёт улучшения кэш-локальности.

Отдельно стоит обратить внимание, что оценка погрешности в норме  $L^\infty$  практически не зависит от числа потоков, но при переходе к более тонкой сетке может выглядеть больше. Это связано с эффектом экстремума (большее число узлов повышает вероятность «поймать» худшую точку) и накоплением фазовой ошибки.