

**Московский государственный университет  
имени М.В.Ломоносова**

ЗАДАНИЕ ПО КУРСУ

**«Суперкомпьютерное моделирование и технологии»**

**Вариант: 4**

**Студент:** Семеняк Г.А.  
**Группа:** 628

Сентябрь 2025 - Декабрь 2025

Москва 2025

1

---

<sup>1</sup>Код решения лежит на Гитхабе: <https://github.com/twist13227/sctm>

## Содержание

<b>1 Математическая постановка дифференциальной задачи</b>	<b>2</b>
<b>2 Численный метод решения задачи</b>	<b>2</b>
<b>3 Программная реализация (MPI)</b>	<b>4</b>
3.1 Результаты MPI ( $L = 1$ ) . . . . .	6
3.2 Результаты MPI ( $L = \pi$ ) . . . . .	6

## 1 Математическая постановка дифференциальной задачи

В трехмерной замкнутой области

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

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

$$\frac{\partial^2 u}{\partial t^2} = a^2 \Delta u \quad (1)$$

с начальными условиями

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

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

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

$$u(0, y, z, t) = 0, \quad u(L_x, y, z, t) = 0, \quad (4)$$

$$u(x, 0, z, t) = 0, \quad u(x, L_y, z, t) = 0, \quad (5)$$

$$u(x, y, 0, t) = 0, \quad u(x, y, L_z, t) = 0, \quad (6)$$

либо периодические граничные условия

$$u(0, y, z, t) = u(L_x, y, z, t), \quad u_x(0, y, z, t) = u_x(L_x, y, z, t), \quad (7)$$

$$u(x, 0, z, t) = u(x, L_y, z, t), \quad u_y(x, 0, z, t) = u_y(x, L_y, z, t), \quad (8)$$

$$u(x, y, 0, t) = u(x, y, L_z, t), \quad u_z(x, y, 0, t) = u_z(x, y, L_z, t). \quad (9)$$

Конкретная комбинация граничных условий определяется индивидуальным вариантом задания (см. п. 5).

## 2 Численный метод решения задачи

Содержание данного пункта основано на материале книги [2]. Для численного решения задачи введем на  $\Omega$  сетку  $\omega_{h\tau} = \bar{\omega}_h \times \omega_\tau$ , где

$$T = T_0,$$

$$L_x = L_{x_0}, L_y = L_{y_0}, L_z = L_{z_0}$$

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

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

Через  $\omega_h$  обозначим множество внутренних, а через  $\gamma_h$  — 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:

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

Здесь  $\Delta_h$  — семиточечный разностный аналог оператора Лапласа:

$$\Delta_h u^n = \frac{u_{i-1,j,k}^n - 2u_{i,j,k}^n + u_{i+1,j,k}^n}{h^2} + \frac{u_{i,j-1,k}^n - 2u_{i,j,k}^n + u_{i,j+1,k}^n}{h^2} + \frac{u_{i,j,k-1}^n - 2u_{i,j,k}^n + u_{i,j,k+1}^n}{h^2}.$$

Приведенная выше разностная схема является явной — значения  $u_{ijk}^{n+1}$  на  $(n+1)$ -м шаге можно явным образом выразить через значения на предыдущих слоях.

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

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

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

$$\frac{u_{ijk}^1 - u_{ijk}^0}{\tau} = a^2 \frac{\Delta_h \varphi(x_i, y_j, z_k)}{2}, \quad (x_i, y_j, z_k) \in \omega_h. \quad (11)$$

$$u_{ijk}^1 = u_{ijk}^0 + a^2 \frac{\Delta_h \varphi(x_i, y_j, z_k)}{2}. \quad (12)$$

Разностная аппроксимация для периодических граничных условий выглядит следующим образом

$$\begin{aligned} u_{0jk}^{n+1} &= u_{Njk}^{n+1}, & u_{1jk}^{n+1} &= u_{N+1jk}^{n+1}, \\ u_{i0k}^{n+1} &= u_{iNk}^{n+1}, & u_{i1k}^{n+1} &= u_{iN+1k}^{n+1}, \\ u_{ij0}^{n+1} &= u_{ijN}^{n+1}, & u_{ij1}^{n+1} &= u_{ijN+1}^{n+1}, \end{aligned}$$

$i, j, k = 0, 1, \dots, N$ .

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

Вычисления далее проводятся для следующей аналитической функции:

$$u(x, y, z, t) = \sin\left(\frac{3\pi}{L_x}x\right) \cdot \sin\left(\frac{2\pi}{L_y}y\right) \cdot \sin\left(\frac{2\pi}{L_z}z\right) \cdot \cos(a_t t + 4\pi),$$

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

Со следующими граничными условиями:

$$u(0, y, z, t) = 0, \quad u(L_x, y, z, t) = 0, \quad (4)$$

$$u(x, 0, z, t) = u(x, Ly, z, t), \quad u(x, L_y, z, t) = u(x, Ly, z, t), \quad (5)$$

$$u(x, y, 0, t) = u(x, y, Lz, t), \quad u(x, y, 0, t) = u(x, y, Lz, t), \quad (6)$$

### 3 Программная реализация (MPI)

MPI-реализация использует трёхмерную декомпозицию вычислительной области. Сетка процессов автоматически формируется в виде трёхмерного тора с помощью `MPI_Dims_create`, что обеспечивает сбалансированное распределение нагрузки. Каждый процесс хранит локальный блок данных с ghost-слоями толщиной в один узел на границах своей подобласти по всем декомпозированным направлениям.

Граничные условия задаются следующим образом:

- По оси  $x$ : непериодические условия (Дирихле  $u = 0$  на внешних границах  $x = 0$  и  $x = L$ )
- По осям  $y$  и  $z$ : периодические условия

Декомпозиция по периодическим направлениям ( $y$  и  $z$ ) всегда требует обмена ghost-слоями между соседними процессами. Для непериодического направления ( $x$ ) обмен выполняется только если декомпозиция затрагивает эту ось (при числе процессов  $> 1$  по  $x$ ), но значения на внешних границах всегда фиксируются как  $u = 0$ .

## Хранение данных и вычисление индекса

Трёхмерная сеточная функция хранится в одномерном массиве в порядке  $i \rightarrow j \rightarrow k$ . Локальный вектор содержит  $(N_i^{\text{local}} + 2) \cdot (N_j^{\text{local}} + 2) \cdot (N_k^{\text{local}} + 2)$  элементов, где дополнительные слои используются для ghost-элементов. Индексация выполняется по формуле:

$$\text{idx}(i, j, k) = i \cdot (N_j^{\text{local}} + 2) \cdot (N_k^{\text{local}} + 2) + j \cdot (N_k^{\text{local}} + 2) + k,$$

где  $i, j, k$  — локальные индексы в пределах подобласти процесса.

## Границные условия и обмен данными

Для поддержания корректных вычислений лапласиана на границах подобластей используется механизм ghost-слоёв:

- По периодическим осям ( $y, z$ ) выполняется полный обмен границами между соседними процессами
- По непериодической оси ( $x$ ) ghost-слои обмениваются только между внутренними границами процессов, а внешние границы ( $x = 0$  и  $x = L$ ) явно устанавливаются в ноль
- Для обмена используются неблокирующие операции `MPI_Isend` и `MPI_Irecv`, что позволяет перекрыть коммуникации с вычислениями
- Циклический обмен для периодических границ реализуется автоматически через топологию коммуникатора `cart_comm`

На каждом временном шаге выполняется следующая последовательность операций:

1. Обмен ghost-слоями по всем декомпозированным периодическим направлениям
2. Вычисление дискретного лапласиана  $\Delta_h u^n$  на внутренних узлах:

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

3. Обновление решения по схеме второго порядка точности:

$$u^{n+1} = 2u^n - u^{n-1} + a^2 \tau^2 \Delta_h u^n$$

4. Принудительная установка  $u = 0$  на внешних границах по оси  $x$  после каждого шага

Для оценки точности решения на каждом шаге:

- Каждый процесс вычисляет локальную максимальную ошибку по своей подобласти:

$$\varepsilon_{\text{local}} = \max_{i,j,k} |u_{i,j,k}^{\text{calc}} - u_{i,j,k}^{\text{exact}}|$$

- С помощью коллективной операции MPI\_Reduce с операцией MPI\_MAX вычисляется глобальный максимум ошибки
- Результат доступен только на корневом процессе (rank 0) для минимизации накладных расходов

Таким образом, MPI-версия обеспечивает масштабирование по числу процессов и позволяет распределить объём вычислений и памяти по кластерам.

Ниже приведены усреднённые по пяти запускам значения времени, погрешности и ускорения.

### 3.1 Результаты MPI ( $L = 1$ )

Таблица 1: Результаты MPI при  $L = 1$

MPI	$N^3$	Время $T$	Ускорение $S$	Погрешность $\delta$	Var(T)
1	$128^3$	9.814	1.00	7.610e-03	0.653
2	$128^3$	4.015	2.44	7.610e-03	0.119
4	$128^3$	2.097	4.68	7.610e-03	0.00125
8	$128^3$	1.415	6.93	7.610e-03	0.0373
16	$128^3$	0.831	11.81	7.610e-03	0.00385
32	$128^3$	0.562	17.45	7.610e-03	0.00178
1	$256^3$	81.274	1.00	3.808e-03	30.026
2	$256^3$	37.036	2.19	3.808e-03	8.199
4	$256^3$	15.577	5.22	3.808e-03	0.0904
8	$256^3$	9.703	8.37	3.808e-03	1.592
16	$256^3$	6.059	13.41	3.808e-03	0.0196
32	$256^3$	3.024	26.88	3.808e-03	0.0069

### 3.2 Результаты MPI ( $L = \pi$ )

Отметим, что MPI-версия демонстрирует почти линейное ускорение до 16 процессов, после чего влияние коммуникаций начинает снижать эффективность.

Таблица 2: Результаты MPI при  $L = \pi$ 

MPI	$N^3$	Время $T$	Ускорение $S$	Погрешность $\delta$	Var(T)
1	$128^3$	9.250	1.00	8.325e-04	2.279
2	$128^3$	4.769	1.94	8.325e-04	0.532
4	$128^3$	2.066	4.48	8.325e-04	0.00141
8	$128^3$	1.421	6.51	8.325e-04	0.0602
16	$128^3$	1.072	8.63	8.325e-04	0.0477
32	$128^3$	0.580	15.95	8.325e-04	0.000671
1	$256^3$	66.829	1.00	4.164e-04	14.733
2	$256^3$	36.675	1.82	4.164e-04	0.100
4	$256^3$	16.132	4.14	4.164e-04	0.945
8	$256^3$	10.218	6.54	4.164e-04	2.891
16	$256^3$	5.843	11.44	4.164e-04	0.0246
32	$256^3$	3.555	18.80	4.164e-04	0.416