



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

СУПЕРКОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ И ТЕХНОЛОГИИ

**Отчет по заданию №3 «Задача для трёхмерного гиперболического уравнения в прямоугольном параллелепипеде»**

ВАРИАНТ №3

*Богатенкова Анастасия Олеговна  
628 группа*

8 ноября 2022 г.

# Содержание

1 Математическая постановка задачи	2
2 Численный метод решения задачи	3
3 Описание программной реализации	5
4 Графики решений и погрешности	6
5 Результаты расчётов	6
6 Выводы	8

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

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

с начальными условиями

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

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

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

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

либо периодические граничные условия

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

В полученном варианте задания (№3) граничные условия выглядят следующим образом:

$$\begin{aligned} u(0, y, z, t) &= 0, & u(L_x, y, z, t) &= 0 \\ u(x, 0, z, t) &= u(x, L_y, z, t), & u_y(x, 0, z, t) &= u_y(x, L_y, z, t) \\ u(x, y, 0, t) &= 0, & u(x, y, L_z, t) &= 0 \end{aligned} \quad (4)$$

Аналитическое уравнение функции  $u$ :

$$\begin{aligned} u(x, y, z, t) &= \sin\left(\frac{\pi}{L_x}x\right) \cdot \sin\left(\frac{2\pi}{L_y}y\right) \cdot \sin\left(\frac{3\pi}{L_z}z\right) \cdot \cos(a_t t) \\ a_t &= \pi \sqrt{\frac{1}{L_x^2} + \frac{4}{L_y^2} + \frac{9}{L_z^2}} \end{aligned}$$

## 2 Численный метод решения задачи

Введём на  $\Omega$  сетку:  $\omega_{h\tau} = \bar{\omega}_h \times \omega_\tau$ , где

$$T = T_0$$

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

$$\bar{\omega}_h = \{(x_i = ih_x, y_j = jh_y, z_k = kh_z), i, j, k = 0, \dots, N, h_xN = L_x, h_yN = L_y, h_zN = L_z\}$$

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

Через  $\omega_h$  обозначим множество внутренних, а через  $\gamma_h$  – множество граничных узлов сетки  $\bar{\omega}_h$ .

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

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

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

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

Приведённая 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 счёта (т.е. 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:

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

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

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

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

Разностная аппроксимация для периодических граничных условий выглядит следующим образом:

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

$$\begin{aligned} u_{1jk}^{n+1} &= u_{N+1jk}^{n+1} \\ u_{i1k}^{n+1} &= u_{iN+1k}^{n+1} \\ u_{ij1}^{n+1} &= u_{ijN+1}^{n+1} \end{aligned}$$

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

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

Таким образом, алгоритм численного решения задачи выглядит следующим образом:

1. Вычислить значения  $u^0, u^1$  на границе:

$$u^{0,1}|_{\gamma_h} \leftarrow 0, (x_i, y_j, z_k) \in \gamma_h \text{ для условий 1-го рода,}$$

$$u^0|_{\gamma_h} \leftarrow u(x_i, y_j, z_k, 0), (x_i, y_j, z_k) \in \gamma_h,$$

$$u^1|_{\gamma_h} \leftarrow u(x_i, y_j, z_k, \tau), (x_i, y_j, z_k) \in \gamma_h \text{ для периодических условий.}$$

2. Вычислить значения  $u^0$  внутри области:

$$u_{ijk}^0 \leftarrow \varphi(x_i, y_j, z_k), (x_i, y_j, z_k) \in \omega_h$$

3. Вычислить значения  $u^1$  внутри области:

$$u_{ijk}^1 \leftarrow u_{ijk}^0 + \frac{\tau^2}{2} \Delta_h \varphi(x_i, y_j, z_k), (x_i, y_j, z_k) \in \omega_h$$

4. Повторять для  $K - 1$  шагов:

вычисляем значения  $u^{n+1}$  на основе двух предыдущих шагов внутри области:

$$u_{ijk}^{n+1} \leftarrow 2u_{ijk}^n - u_{ijk}^{n-1} + \tau^2 \cdot \Delta_h u^n, (x_i, y_j, z_k) \in \omega_h$$

вычисляем граничные значения  $u^{n+1}$ :

$$u_{ijk}^{n+1} \leftarrow 0, (x_i, y_j, z_k) \in \gamma_h \text{ для условий 1-го рода,}$$

$$u_{ijk}^{n+1} \leftarrow u(x_i, y_j, z_k, (n+1) \cdot \tau), (x_i, y_j, z_k) \in \gamma_h \text{ для периодических условий.}$$

### 3 Описание программной реализации

Реализовано две программы: последовательная и параллельная с использованием MPI. Программа принимает на вход число точек сетки по одной из осей  $N$ , длину отрезка решения по одной из осей  $L$  ( $L_x = L_y = L_z = L$ ), а также имя файла для вывода результатов *out\_file*. Программа выводит в файл число точек  $N$ , число процессов MPI, погрешность решения на последнем временном слое и время работы программы.

Параллельная версия программы выполнена следующим образом:

1. Сетка разбивается на блоки, каждый из которых достаётся процессу для вычисления функции;
2. Каждый процесс вычисляет своих соседей (номера процессов-соседей и прямоугольники-границы блоков);
3. Каждый процесс заполняет свой блок начальными значениями  $u_0$ ,  $u_1$  с использованием аналитической функции и формул предыдущей главы.
4. В цикле происходит вычисление функции на очередном временном слое, при этом происходят пересылки прямоугольников-границ блоков для вычисления оператора Лапласа.
5. В конце вычислений с помощью *MPI\_Reduce* вычисляется погрешность вычисления функции на последнем временном слое как максимум из локальных ошибок каждого из процессов.

Разбиение сетки на блоки происходит рекурсивно по следующему алгоритму:

- разбиение начинается с оси  $X$  и целой сетки, изначальный *size* равен числу процессов;
- если текущее количество областей *size* = 1, возвращается текущий параллелепипед;
- если *size* нечётно, то по текущей оси область делится на две части: одна со стороной, делённой на *size*, и оставшаяся часть. Первая часть достаётся 1 процессу, для оставшейся выполняется следующий пункт;
- по выбранной оси область делится пополам, алгоритм запускается рекурсивно для подобластей переходя на следующую ось ( $X \rightarrow Y$ ;  $Y \rightarrow Z$ ;  $Z \rightarrow X$ ).

## 4 Графики решений и погрешности

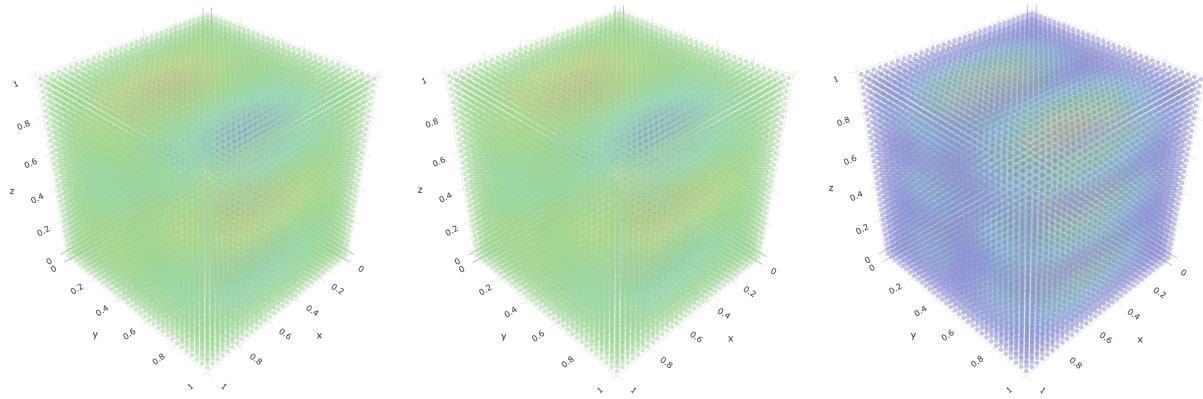


Рис. 1: Графики аналитической функции, вычисленной функции и погрешности вычисления для  $L_x = L_y = L_z = 1$

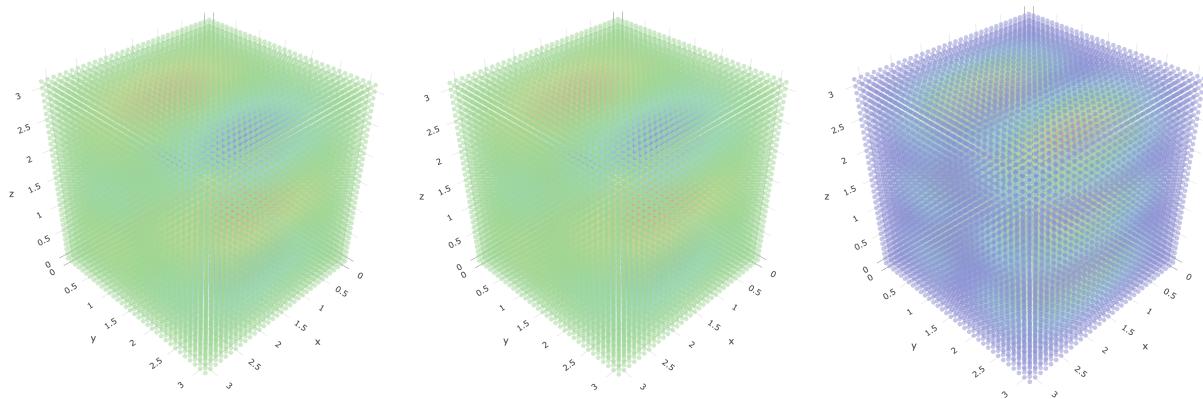


Рис. 2: Графики аналитической функции, вычисленной функции и погрешности вычисления для  $L_x = L_y = L_z = \pi$

## 5 Результаты расчётов

В таблицах 1, 2 и на графиках 3, 4 представлены результаты расчётов на суперкомпьютере Polus для  $L_x = L_y = L_z = L$ ,  $T = 1$ ,  $K = 1000$ . Для каждого набора  $p$ ,  $N$ ,  $L$  время работы программы усреднялось по 10 запускам. В таблицах представлены усреднённые значения времени и ускорения.

Число MPI-процессов	Число точек сетки $N^3$	Время решения $T$	Ускорение $S$	Погрешность $\delta$
1	$128^3$	5.739	1.000	$9.3 \cdot 10^{-6}$
10	$128^3$	0.761	7.544	$9.3 \cdot 10^{-6}$
20	$128^3$	0.447	12.831	$9.3 \cdot 10^{-6}$
40	$128^3$	0.374	15.329	$9.3 \cdot 10^{-6}$
1	$256^3$	46.286	1.000	$2.1 \cdot 10^{-6}$
10	$256^3$	5.103	9.071	$2.1 \cdot 10^{-6}$
20	$256^3$	2.863	16.166	$2.1 \cdot 10^{-6}$
40	$256^3$	1.668	27.748	$2.1 \cdot 10^{-6}$
1	$512^3$	375.106	1.000	$2.9 \cdot 10^{-7}$
10	$512^3$	40.720	9.212	$2.9 \cdot 10^{-7}$
20	$512^3$	21.517	17.433	$2.9 \cdot 10^{-7}$
40	$512^3$	11.474	32.690	$2.9 \cdot 10^{-7}$

Таблица 1: Результаты для Polus при  $L = 1$

Число MPI-процессов $p$	Число точек сетки $N^3$	Время решения $T$	Ускорение $S$	Погрешность $\delta$
1	$128^3$	5.757	1.000	$9.8 \cdot 10^{-7}$
10	$128^3$	0.776	7.415	$9.8 \cdot 10^{-7}$
20	$128^3$	0.494	11.657	$9.8 \cdot 10^{-7}$
40	$128^3$	0.366	15.711	$9.8 \cdot 10^{-7}$
1	$256^3$	46.297	1.000	$2.4 \cdot 10^{-7}$
10	$256^3$	5.204	8.896	$2.4 \cdot 10^{-7}$
20	$256^3$	2.930	15.801	$2.4 \cdot 10^{-7}$
40	$256^3$	1.639	28.245	$2.4 \cdot 10^{-7}$
1	$512^3$	372.260	1.000	$5.8 \cdot 10^{-8}$
10	$512^3$	40.734	9.139	$5.8 \cdot 10^{-8}$
20	$512^3$	21.860	17.030	$5.8 \cdot 10^{-8}$
40	$512^3$	11.612	32.058	$5.8 \cdot 10^{-8}$

Таблица 2: Результаты для Polus при  $L = \pi$

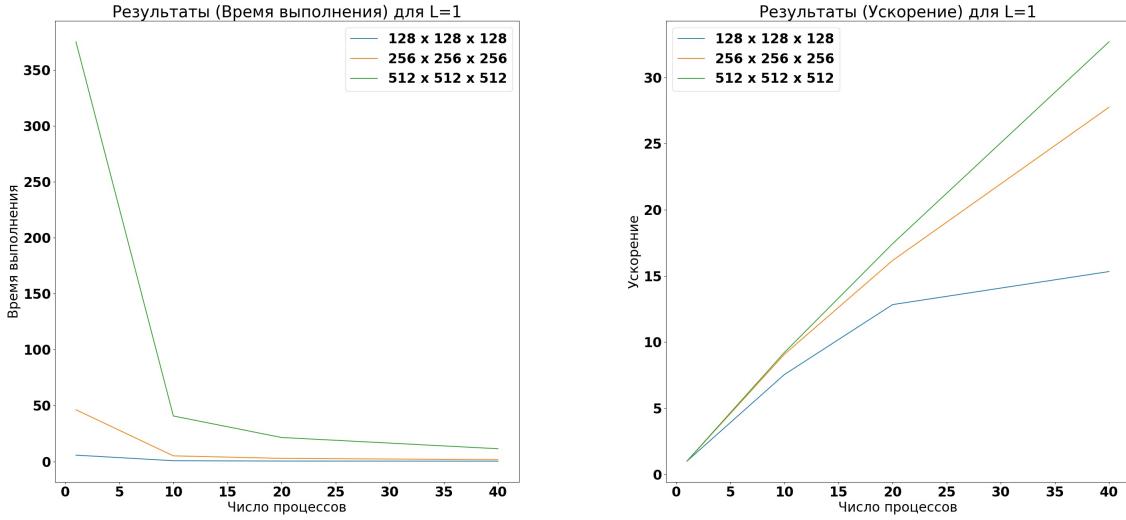


Рис. 3: Графики времени работы и ускорения для Polus при  $L = 1$

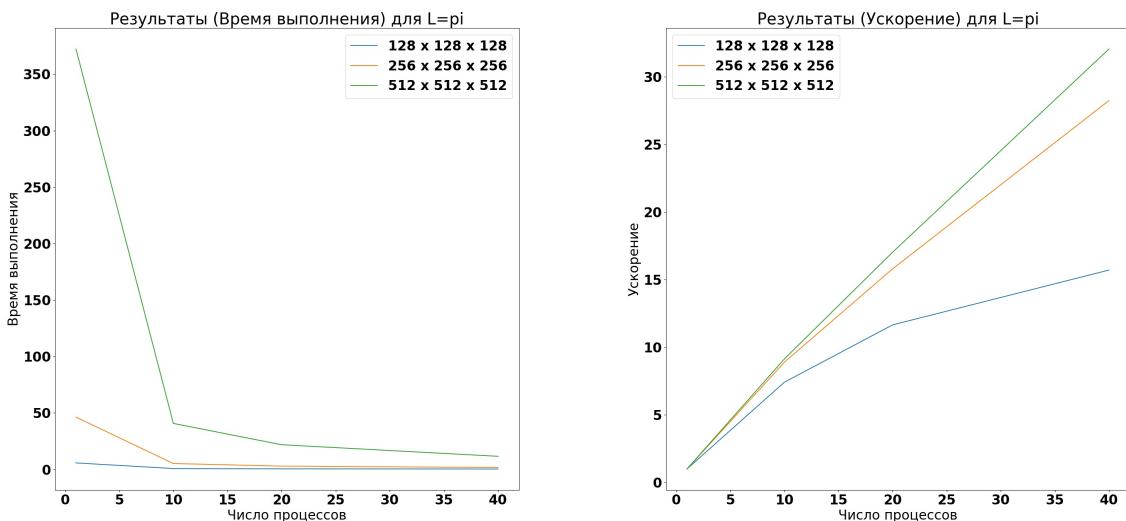


Рис. 4: Графики времени работы и ускорения для Polus при  $L = \pi$

## 6 Выводы

Задача для трёхмерного гиперболического уравнения в прямоугольном параллелепипеде отлично подходит для распараллеливания. В результате получены программные средства, решающие поставленную задачу средствами MPI.

Результаты расчётов показали, что чем выше число точек сетки, тем лучше

показатели распараллеливания программы, так как накладные расходы, связанные с пересылкой данных между процессами становятся незначительными по сравнению со сложностью задачи, решаемой каждым из процессов. При этом, если зафиксировать размер сетки, с ростом числа процессов ускорение будет всё больше отличаться от идеального. Из этого можно сделать вывод о том, что параллельный алгоритм решения данной задачи может существенно снизить временные затраты на подсчет значений функции, а также позволяет вычислять значения с большей точностью.