



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

**ОТЧЕТ О ВЫПОЛНЕНИИ
ЗАДАНИЯ ПО КУРСУ
«Суперкомпьютерное моделирование
и технологии»**

ВАРИАНТ 6

Выполнила:
Мирова Елизавета Сергеевна
группа 616, кафедра МС

Москва, 2025

Содержание

1	Математическая постановка задачи	3
2	Численный метод решения задачи	4
2.1	Метод фиктивных областей	4
2.2	Разностная схема решения задачи	5
2.3	Метод решения системы линейных алгебраических уравнений	6
3	Описание проделанной работы	8
3.1	Математические выкладки	8
3.1.1	Решение основной задачи	8
3.1.2	Алгоритм двумерного разбиения прямоугольника	9
3.2	Программы	10
3.2.1	Последовательная программа	10
3.2.2	OpenMP-программа	10
3.2.3	MPI-программа	10
3.2.4	MPI+OpenMP-программа	10
3.2.5	MPI+CUDA-программа	10
4	Результаты	11
5	О результатах	14
5.1	Команды компиляции программ	14
5.2	Оценка корректности	14
5.3	Анализ результатов	14

1 Математическая постановка задачи

В области $D \subset \mathbb{R}^2$, ограниченной кусочно-гладким контуром γ , рассматривается дифференциальное уравнение Пуассона

$$-\Delta u = f(x, y), \quad (x, y) \in D, \quad (1)$$

в котором оператор Лапласа

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}.$$

Функция $f(x, y)$ считается известной. Для выделения единственного решения уравнение дополняется граничными условиями Дирихле:

$$u(x, y) = 0, \quad (x, y) \in \gamma. \quad (2)$$

Требуется найти функцию $u(x, y)$, удовлетворяющую уравнению (1) в области D и краевому условию (2) на ее границе, если область D задана следующим образом:

$$D = \{(x, y) \in \mathbb{R}^2 : |x| + |y| < 2, \quad y < 1\},$$

а функция $f(x, y)$ равна единице в каждой точке области D .

2 Численный метод решения задачи

2.1 Метод фиктивных областей

Для приближенного решения задачи (1),(2) предлагается воспользоваться методом фиктивных областей. Пусть область D принадлежит прямоугольнику $\Pi = \{(x, y) : A_1 < x < B_1, A_2 < y < B_2\}$. Обозначим через $\bar{D}, \bar{\Pi}$ замыкание области D и прямоугольника Π соответственно, через Γ – границу прямоугольника. Разность множеств $\hat{D} = \Pi \setminus D$ называется фиктивной областью.

Выберем и зафиксируем малое $\varepsilon > 0$. В прямоугольнике Π рассматривается задача Дирихле

$$-\frac{\partial}{\partial x} \left(k(x, y) \frac{\partial v}{\partial x} \right) - \frac{\partial}{\partial y} \left(k(x, y) \frac{\partial v}{\partial y} \right) = F(x, y), \quad (x, y) \in \Pi \setminus \gamma, \quad (3)$$

с кусочно-постоянным коэффициентом

$$k(x, y) = \begin{cases} 1, & (x, y) \in D, \\ 1/\varepsilon, & (x, y) \in \hat{D}, \end{cases} \quad (4)$$

и правой частью

$$F(x, y) = \begin{cases} f(x, y), & (x, y) \in D, \\ 0, & (x, y) \in \hat{D}. \end{cases} \quad (5)$$

Требуется найти непрерывную в $\bar{\Pi}$ функцию $v(x, y)$, удовлетворяющую дифференциальному уравнению (3) всюду в $\Pi \setminus \gamma$, равную нулю на границе Γ прямоугольника, и такую, чтобы вектор потока

$$\mathbf{W}(x, y) = -k(x, y) \left(\frac{\partial v}{\partial x}, \frac{\partial v}{\partial y} \right)$$

имел непрерывную нормальную компоненту на общей части криволинейной границы области D и прямоугольника Π . Последнее означает, что в каждой точке $(x_0, y_0) \in \gamma \cap \Pi$ должно выполняться равенство

$$\lim_{\substack{(x,y) \rightarrow (x_0,y_0) \\ (x,y) \in D}} \langle \mathbf{W}(x, y), \mathbf{n}(x_0, y_0) \rangle = \lim_{\substack{(x,y) \rightarrow (x_0,y_0) \\ (x,y) \in \hat{D}}} \langle \mathbf{W}(x, y), \mathbf{n}(x_0, y_0) \rangle, \quad (6)$$

где $\mathbf{n}(x, y)$ – вектор единичной нормали к границе γ в точке (x, y) , определенный всюду или почти всюду на кривой.

Известно, что функция $v(x, y)$ равномерно приближает решение $u(x, y)$ задачи (1),(2) в области D , а именно,

$$\max_{(x,y) \in D} |v(x, y) - u(x, y)| < C\varepsilon, \quad C > 0. \quad (7)$$

В частности, $|v(x, y)| < C\varepsilon$ во всех точках кривой γ . Этот результат позволяет получить исковую функцию $u(x, y)$ с любой наперед заданной точностью $\varepsilon > 0$, решая задачу (3),(6) вместо задачи (1),(2). Тем самым, задача Дирихле в криволинейной области приближенно заменяется задачей Дирихле в прямоугольнике с кусочно-постоянным коэффициентом $k(x, y)$.

2.2 Разностная схема решения задачи

Краевую задачу (3),(6) предлагается решать численно методом конечных разностей. В замыкании прямоугольника Π определяется равномерная прямоугольная сетка $\bar{\omega}_h = \bar{\omega}_1 \times \bar{\omega}_2$, где

$$\bar{\omega}_1 = \{x_i = A_1 + ih_1, i = \overline{0, M}\}, \quad \bar{\omega}_2 = \{y_j = A_2 + jh_2, j = \overline{0, N}\}.$$

Здесь $h_1 = (B_1 - A_1)/M$, $h_2 = (B_2 - A_2)/N$. Через ω_h обозначим множество внутренних узлов сетки $\bar{\omega}_h$, т.е. множество узлов сетки прямоугольника, не лежащих на границе Γ .

Рассмотрим линейное пространство H функций, заданных на сетке ω_h . Обозначим через w_{ij} значение сеточной функции $w \in H$ в узле сетки $(x_i, y_j) \in \omega_h$. Будем считать, что в пространстве H задано скалярное произведение и евклидова норма

$$(u, v) = \sum_{i=1}^{M-1} \sum_{j=1}^{N-1} h_1 h_2 u_{ij} v_{ij}, \quad \|u\|_E = \sqrt{(u, u)}. \quad (8)$$

В методе конечных разностей дифференциальная задача математической физики заменяется конечно-разностной операторной задачей вида

$$Aw = B, \quad (9)$$

где $A : H \rightarrow H$ – оператор, действующий в пространстве сеточных функций, $B \in H$ – известная правая часть. Задача (9) называется разностной схемой. Ее решение приближает точное решение в узлах сетки $\bar{\omega}_h$ и является численным решением исходной дифференциальной задачи.

При построении разностной схемы следует аппроксимировать (приближенно заменить) все уравнения краевой задачи их разностными аналогами – сеточными уравнениями, связывающими значения искомой сеточной функции в узлах сетки. Полученные таким образом уравнения должны быть функционально независимыми, а их общее количество – совпадать с числом неизвестных, т.е. с количеством узлов сетки.

Дифференциальное уравнение задачи (3) во всех внутренних точках сетки аппроксируется разностным уравнением

$$-\frac{1}{h_1} \left(a_{i+1j} \frac{w_{i+1j} - w_{ij}}{h_1} - a_{ij} \frac{w_{ij} - w_{i-1j}}{h_1} \right) - \frac{1}{h_2} \left(b_{ij+1} \frac{w_{ij+1} - w_{ij}}{h_2} - b_{ij} \frac{w_{ij} - w_{ij-1}}{h_2} \right) = F_{ij}, \quad (10)$$

$i = \overline{1, M-1}$, $j = \overline{1, N-1}$, в котором коэффициенты

$$a_{ij} = \frac{1}{h_2} \int_{y_{j-1/2}}^{y_{j+1/2}} k(x_{i-1/2}, t) dt, \quad b_{ij} = \frac{1}{h_1} \int_{x_{i-1/2}}^{x_{i+1/2}} k(t, y_{j-1/2}) dt \quad (11)$$

при всех $i = \overline{1, M}$, $j = \overline{1, N}$. Здесь полуцелые узлы

$$x_{i\pm 1/2} = x_i \pm 0.5h_1, \quad y_{j\pm 1/2} = y_j \pm 0.5h_2.$$

Правая часть разностного уравнения

$$F_{ij} = \frac{1}{h_1 h_2} \iint_{\Pi_{ij}} F(x, y) dx dy, \quad \Pi_{ij} = \{(x, y) : x_{i-1/2} \leq x \leq x_{i+1/2}, \quad y_{j-1/2} \leq y \leq y_{j+1/2}\} \quad (12)$$

при всех $i = \overline{1, M-1}$, $j = \overline{1, N-1}$.

Введем обозначения правой и левой разностных производных по переменным x, y соответственно:

$$w_{\bar{x},ij} = \frac{w_{ij} - w_{i-1j}}{h_1}, \quad w_{x,ij} = \frac{w_{i+1j} - w_{ij}}{h_1}, \quad w_{\bar{y},ij} = \frac{w_{ij} - w_{ij-1}}{h_2}, \quad w_{y,ij} = \frac{w_{ij+1} - w_{ij}}{h_2}.$$

С учетом принятых обозначений разностное уравнение (10) можно представить в более компактном и удобном виде:

$$-(aw_{\bar{x}})_{x,ij} - (bw_{\bar{y}})_{y,ij} = F_{ij}, \quad i = \overline{1, M-1}, \quad j = \overline{1, N-1}. \quad (13)$$

Краевые условия Дирихле задачи (3),(6) аппроксимируются точно равенством

$$w_{ij} = w(x_i, y_j) = 0, \quad (x_i, y_j) \in \Gamma. \quad (14)$$

Переменные w_{ij} , заданные равенством (14), исключаются из системы уравнений (13). В результате остаются неизвестными значения w_{ij} при $i = \overline{1, M-1}$, $j = \overline{1, N-1}$ и их количество совпадает с числом уравнений. Система является линейной относительно неизвестных величин и может быть представлена в виде (9) с самосопряженным и положительно определенным оператором $Aw = -(aw_{\bar{x}})_x - (bw_{\bar{y}})_y$ и правой частью F , определенной равенством (12). Таким образом, построенная разностная схема (13),(14) линейна и имеет единственное решение при любой правой части.

Интегралы (11) от кусочно-постоянной функции $k(x, y)$ следует вычислять аналитически. Нетрудно видеть, что если отрезок, соединяющий точки $P_{ij} = (x_{i-1/2}, y_{j-1/2})$ и $P_{ij+1} = (x_{i-1/2}, y_{j+1/2})$, целиком расположен в области D , то $a_{ij} = 1$. Если же указанный отрезок находится в фиктивной области \hat{D} , то $a_{ij} = 1/\varepsilon$. В противном случае $a_{ij} = h_2^{-1}l_{ij} + (1 - h_2^{-1}l_{ij})/\varepsilon$, где l_{ij} – длина той части отрезка $[P_{ij}, P_{ij+1}]$, которая принадлежит области D . Аналогичным образом вычисляются коэффициенты b_{ij} .

Очевидно, правая часть схемы F_{ij} равна нулю при всех $(i, j) : \Pi_{ij} \subset \hat{D}$. Если $\Pi_{ij} \subset D$, то правую часть предлагается приближенно заменить значением $f(x_i, y_j)$. В противном случае, когда прямоугольник Π_{ij} содержит точки оригинальной области D и фиктивной области \hat{D} , величина F_{ij} может быть вычислена приближенно как произведение $(h_1 h_2)^{-1} S_{ij} f(x_i^*, y_j^*)$, где (x_i^*, y_j^*) – любая точка пересечения $\Pi_{ij} \cap D$, $S_{ij} = \text{mes}(\Pi_{ij} \cap D)$ – площадь пересечения множеств, при вычислении которой криволинейную часть границы можно заменить отрезком прямой.

2.3 Метод решения системы линейных алгебраических уравнений

Приближенное решение разностной схемы (10),(14) может быть получено итерационным методом сопряженных градиентов [4],[5]. Для ускорения сходимости метода применяется диагональное предобуславливание. Пусть оператор $D : H \rightarrow H$ действует на сеточные функции $w \in H$ по правилу

$$(Dw)_{ij} = \left[\frac{a_{i+1,j} + a_{i,j}}{h_1^2} + \frac{b_{i,j+1} + b_{i,j}}{h_2^2} \right] w_{ij}, \quad i = \overline{1, M-1}, \quad j = \overline{1, N-1}.$$

Начальное приближение $w^{(0)}$ к решению разностной схемы можно выбрать любым способом, например, равным нулю во всех точках расчетной сетки. Первая итерация совершается по формулам метода скорейшего спуска. Пусть $r^{(0)} = B - Aw^{(0)}$ – невязка начального приближения, функция $z^{(0)} \in H$ удовлетворяет уравнению $Dz^{(0)} = r^{(0)}$. Тогда направление спуска $p^{(1)} = z^{(0)}$, шаг вдоль направления спуска определяется параметром

$$\alpha_1 = \frac{(z^{(0)}, r^{(0)})}{(Ap^{(1)}, p^{(1)})}.$$

Следующее приближение $w^{(1)}$ вычисляется согласно равенству

$$w^{(1)} = w^{(0)} + \alpha_1 p^{(1)}. \quad (15)$$

Дальнейшие вычисления проводятся по следующим формулам. Пусть выполнено k итераций метода и функции $r^{(k-1)}, z^{(k-1)}, p^{(k)}, w^{(k)} \in H$, а также коэффициент α_k являются известными. Тогда невязка последней итерации $r^{(k)} = r^{(k-1)} - \alpha_k A p^{(k)}$, сеточная функция $z^{(k)} \in H$ вычисляется из уравнения $Dz^{(k)} = r^{(k)}$. Следующее направление спуска:

$$p^{(k+1)} = z^{(k)} + \beta_{k+1} p^{(k)},$$

где коэффициент

$$\beta_{k+1} = \frac{(z^{(k)}, r^{(k)})}{(z^{(k-1)}, r^{(k-1)})}.$$

Шаг спуска определяется параметром

$$\alpha_{k+1} = \frac{(z^{(k)}, r^{(k)})}{(Ap^{(k+1)}, p^{(k+1)})}.$$

Следующее приближение к точному решению $w^{(k+1)}$ вычисляется согласно равенству:

$$w^{(k+1)} = w^{(k)} + \alpha_{k+1} p^{(k+1)}. \quad (16)$$

Метод сопряженных градиентов гарантирует, что при некотором k , не превосходящем количества неизвестных $(M-1) \times (N-1)$, приближение $w^{(k)}$ станет равным точному решению разностной схемы. На практике это равенство нарушается из-за ошибок округлений, возникающих в процессе вычислений, и в качестве условия остановки итерационного процесса следует использовать неравенство

$$\|w^{(k+1)} - w^{(k)}\|_E < \delta, \quad (17)$$

где δ – положительное число, определяющее точность итерационного метода. Оценку точности приближенного решения можно проводить в других нормах пространства сеточных функций, например, в максимум норме

$$\|w\|_C = \max_{x \in \omega_h} |w(x)|. \quad (18)$$

Константу δ для данной задачи предлагается выбрать так, чтобы итерационный процесс укладывался в отведенное для него время.

Замечание. Метод сопряженных градиентов является методом вариационного типа, в основе которого находится задача минимизации квадратичного функционала $J(w) = 0.5(Aw, w) - (B, w)$, эквивалентная системе уравнений (9). О качестве работы метода можно судить по поведению функционала $J(w)$, который должен монотонно убывать на итерационной последовательности $w^{(k)}$, $k = 0, 1, 2, \dots$. Поскольку невязка $r^{(k)} = B - Aw^{(k)}$ на каждой итерации, то

$$J(w^{(k)}) = -0.5(B + r^{(k)}, w^{(k)}) = -0.5H(w^{(k)}), \quad H(w^{(k)}) = (B + r^{(k)}, w^{(k)}).$$

Ошибки округления, возникающие во время работы метода сопряженных градиентов, могут нарушить сходимость метода. Поэтому целесообразно отслеживать монотонность скалярного произведения $H(w^{(k)})$, $k = 0, 1, 2, \dots$. Если монотонный рост нарушен, то следует остановить итерационный процесс и перезапустить его, взяв в качестве начального приближения функцию $w^{(k)}$ с той итерации, на которой монотонность соблюдалась. Необходимо также иметь в виду, что количество итераций в методе сопряженных градиентов не должно превосходить числа неизвестных задачи.

3 Описание проделанной работы

3.1 Математические выкладки

3.1.1 Решение основной задачи

Численный метод решения задачи представлен в общем виде, и для выполнения задания необходимо вывести формулы для частного случая, когда

$$D = \{(x, y) \in \mathbb{R}^2 : |x| + |y| < 2, y < 1\}, \quad f(x, y) = 1, \quad (x, y) \in D.$$

Для начала определим прямоугольник Π , внутри которого рассматривается данная задача Дирихле:

$$\Pi = \{(x, y) \in \mathbb{R}^2 : -2 < x < 2, -2 < y < 1\}.$$

Определим равномерную прямоугольную сетку

$$\bar{\omega}_1 = \{x_i = -2 + ih_1, \quad i = \overline{0, M}\}, \quad \bar{\omega}_2 = \{y_j = -2 + jh_2, \quad j = \overline{0, N}\}, \quad h_1 = \frac{4}{M}, \quad h_2 = \frac{3}{N}$$

и возьмем $\varepsilon = \max(h_1, h_2)^2$.

Коэффициенты a_{ij} из разностного уравнения (10) требуют вычисления длины частей отрезка $[P_{ij}, P_{ij+1}]$, где $P_{ij} = (x_{i-1/2}, y_{j-1/2})$, $P_{ij+1} = (x_{i+1/2}, y_{j+1/2})$. Найдем значения на границе при $x = x_{i-1/2}$:

$$y_i^{max} = \min(2 - |x_{i-1/2}|, 1), \quad y_i^{min} = |x_{i-1/2}| - 2.$$

Длина отрезка в области D

$$l_{ij}^a = \min(y_i^{max}, y_{j+1/2}) - \max(y_i^{min}, y_{j-1/2}).$$

Формула верна в случае, когда отрезок имеет с областью D непустое пересечение. Чтобы она была корректна и при пустом пересечении, внесем в нее небольшое изменение:

$$l_{ij}^a = \max(\min(y_i^{max}, y_{j+1/2}) - \max(y_i^{min}, y_{j-1/2}), 0), \quad i = \overline{1, M}, j = \overline{1, N}. \quad (19)$$

Аналогично получаем l_{ij}^b :

$$l_{ij}^b = \max(\min(x_j^{max}, x_{i+1/2}) - \max(x_j^{min}, x_{i-1/2}), 0), \quad i = \overline{1, M}, j = \overline{1, N}, \quad (20)$$

где

$$x_j^{max} = 2 - |y_{j-1/2}|, \quad x_j^{min} = |y_{j-1/2}| - 2.$$

Осталось найти коэффициент F_{ij} , который при данной $f(x, y)$ выражается в

$$F_{ij} = mes(\Pi_{ij} \cap D) \cdot \frac{1}{h_1 h_2}, \quad \Pi_{ij} = \{(x, y) : x_{i-1/2} \leq x \leq x_{i+1/2}, y_{j-1/2} \leq y \leq y_{j+1/2}\}.$$

Для вычисления площади достаточно воспользоваться самым простым методом - методом средних прямоугольников. Подсчет высоты пересечения аналогичен формуле (19).

3.1.2 Алгоритм двумерного разбиения прямоугольника

Дана прямоугольная расчетная область Π , на которой задана равномерная прямоугольная сетка: M узлов по оси x с индексами $i = 0, 1, \dots, M - 1$ и N узлов по оси y с индексами $j = 0, 1, \dots, N - 1$. Требуется разбить сетку на P прямоугольных доменов (подобластей), так чтобы выполнялись следующие условия:

1. Отношение количества узлов по переменным x и y в каждом домене принадлежит диапазону $[\frac{1}{2}, 2]$, или формально:

$$\frac{M_p}{N_p} \in [\frac{1}{2}, 2], \quad \forall p = 0, 1, \dots, P - 1,$$

M_p, N_p - число узлов в p -м домене по осям x и y соответственно.

2. Количество узлов по переменным x и y любых двух доменов отличается не более, чем на единицу, то есть:

$$|M_{p_1} - M_{p_2}| \leq 1, \quad |N_{p_1} - N_{p_2}| \leq 1, \quad \forall p_1, p_2 \in \{0, 1, \dots, P - 1\}.$$

Идеальная непрерывная оценка

Пусть M_p, N_p вещественные для всех p . Введем $P_x, P_y \in \mathbb{R}_+$ - число доменов по осям x и y соответственно: $P_x \cdot P_y = P$. Для выполнения второго условия поделим оси на равные части (что всегда возможно в непрерывном случае), то есть для домена с индексом p :

$$M_p = \frac{M}{P_x}, \quad N_p = \frac{N}{P_y}.$$

Тогда первое условие разбиения выглядит следующим образом:

$$R = \frac{M_p}{N_p} = \frac{M/P_x}{N/P_y} = \frac{M}{N} \cdot \frac{P}{P_x^2}.$$

Получаем допустимый диапазон для P_x :

$$\frac{1}{2} \leq R \leq 2 \Leftrightarrow \sqrt{\frac{P}{2} \cdot \frac{M}{N}} \leq P_x \leq \sqrt{2P \cdot \frac{M}{N}}.$$

Будем разбивать сетку на квадратные домены, так как такое разбиение эффективнее с точки зрения параллельных вычислений:

$$R = 1 \Leftrightarrow P_x^* = \sqrt{P \cdot \frac{M}{N}}.$$

Алгоритм решения в целых числах

Учитывая идеальную оценку, получаем следующий алгоритм:

1. Находим P_x - делитель числа P , ближайший к $P_x^* = \sqrt{P \cdot \frac{M}{N}}$ и принадлежащий допустимому диапазону; полагаем $P_y = \frac{P}{P_x}$.
2. Разбиваем расчетную область на почти равномерную прямоугольную сетку с количеством доменов P_x и P_y по осям x и y соответственно. Формулы для оси x (для y аналогично):

$$M_p = q_x + [p < r_x], \quad q_x = \left\lfloor \frac{M}{P_x} \right\rfloor, \quad r_x = M \bmod P_x, \quad p = 0, 1, \dots, P_x - 1.$$

Стоит отметить, что алгоритм не работает для произвольных M, N, P , так как подходящего разбиения может и не существовать. Однако он гарантированно работает для чисел из задания.

3.2 Программы

3.2.1 Последовательная программа

Программа была написана в несколько этапов:

1. Получение математических выкладок из пункта 3.1.1.
2. Написание и отладка подсчета коэффициентов разностного уравнения (10).
Ручная проверка на сетке 8×6 , имеющей удобный для этого шаг в 0.5.
3. Написание итерационного метода сопряженных градиентов, проверка его сходимости и подбор оптимальных параметров на сгущающихся сетках
$$(M, N) = (10, 10), (20, 20), (40, 40).$$
4. Реструктуризация: логика работы программы была поделена между несколькими классами, в результате чего код стал более комфортным для чтения и отладки.

3.2.2 OpenMP-программа

Для написания OpenMP-программы по последовательной, были выполнены шаги:

1. Выявление распараллелиемых частей кода.
2. Добавление нужных директив во все распараллелиемые функции.
3. Сравнение времени работы на сетке $(40, 40)$ с 1, 4 и 16 OpenMP-нитями для проверки корректности параллельной версии.

3.2.3 MPI-программа

Для написания MPI-программы были проделаны следующие шаги:

1. Разработка алгоритма двумерного разбиения расчетной области на прямоугольные домены (пункт 3.1.2).
2. Написание и отладка кода: добавление новых классов и функций, связанных с разбиением на домены, и модификация существующих.
3. Проверка качества работы алгоритма на сетке $(M, N) = (40, 40)$ на одном, двух и четырех процессах, сравнение с последовательным вариантом алгоритма.

3.2.4 MPI+OpenMP-программа

Для написания MPI+OpenMP-программы по MPI было проделано 2 шага:

1. Добавление аналогичных OpenMP-директив во все распараллелиемые циклы.
2. Сравнение времени работы на сетке $(40, 40)$ на одном и двух процессах с 4 нитями для проверки корректности программы.

3.2.5 MPI+CUDA-программа

Для написания MPI+CUDA-программы по MPI, были выполнены следующие шаги:

1. Переписывание вычислительных функций в виде CUDA-ядр, обеспечивающих параллельное выполнение на GPU.
2. Добавление функций для корректного распределения памяти и работы с устройством.

4 Результаты

OpenMP нити	Размер сетки	Число итераций	Полное время работы	Ускорение	calcScaledProd цикл	calcScaledAdd цикл	calcZ цикл	ApplyA цикл	D цикл	a, b, F цикл
1	400 × 600	1855	10.842	1.00	2.047	1.257	0.944	6.541	0.002	0.033
2	400 × 600	1845	5.398	2.01	1.028	0.613	0.454	3.274	0.001	0.016
4	400 × 600	1845	2.763	3.92	0.531	0.316	0.234	1.665	0.000	0.009
8	400 × 600	1845	1.431	7.57	0.282	0.179	0.123	0.834	0.000	0.004
16	400 × 600	1845	0.879	12.33	0.198	0.149	0.081	0.441	0.000	0.002
1	800 × 1200	3706	87.152	1.00	16.807	10.005	7.589	52.496	0.005	0.160
4	800 × 1200	3706	22.510	3.87	4.261	2.641	1.894	13.653	0.002	0.036
8	800 × 1200	3706	11.378	7.66	2.186	1.316	0.982	6.855	0.001	0.019
16	800 × 1200	3706	6.058	14.39	1.194	0.767	0.519	3.556	0.001	0.009
32	800 × 1200	3704	5.912	14.74	0.785	0.789	0.496	3.810	0.001	0.006

Таблица 1: Детальный анализ работы OpenMP-программы на ПВС IBM Polus в секундах. Для 1 нити взяты результаты исходной последовательной программы.

MPI процессы	Размер сетки	Число итераций	Полное время работы	Ускорение	MPI коммуникация	calcScalarProd цикл	calcScaledAdd цикл	calcZ цикл	ApplyA цикл	D цикл	a, b, F цикл
1	400 × 600	1855	10.842	1.00	0.000	2.047	1.257	0.944	6.541	0.002	0.033
2	400 × 600	1846	4.459	2.43	0.061	1.143	0.484	0.599	2.129	0.001	0.019
4	400 × 600	1845	2.241	4.84	0.077	0.560	0.231	0.295	1.049	0.001	0.009
8	400 × 600	1845	1.289	8.41	0.117	0.303	0.112	0.165	0.568	0.000	0.007
16	400 × 600	1845	0.790	13.72	0.219	0.146	0.055	0.079	0.275	0.000	0.002
1	800 × 1200	3706	87.152	1.00	0.000	16.807	10.005	7.589	52.496	0.005	0.160
4	800 × 1200	3706	19.366	4.50	1.059	4.734	2.105	2.494	8.877	0.002	0.038
8	800 × 1200	3708	9.952	8.76	0.962	2.334	0.995	1.225	4.362	0.001	0.019
16	800 × 1200	3706	5.468	15.94	0.839	1.202	0.493	0.632	2.251	0.001	0.011
20	800 × 1200	3706	4.561	19.11	0.767	0.978	0.403	0.521	1.845	0.001	0.009
32	800 × 1200	3705	2.799	31.13	0.504	0.597	0.216	0.316	1.123	0.000	0.006

Таблица 2: Детальный анализ работы MPI-программы на ПВС IBM Polus в секундах. Для 1 процесса взяты результаты исходной последовательной программы. Время посчитано как среднее времен всех процессов.

MPI процессы	OpenMP нити	Размер сетки	Число итераций	Полное время работы	Ускорение	MPI коммуникация	calcScalarProd цикл	calcScaledAdd цикл	calcZ цикл	ApplyA цикл	D цикл	a, b, F цикл
1	1	400 × 600	1855	10.842	1.00	0.000	2.047	1.257	0.944	6.541	0.002	0.033
2	1	400 × 600	1845	2.679	4.05	0.100	0.250	0.352	0.392	1.484	0.001	0.016
2	2	400 × 600	1845	2.089	5.19	0.087	0.229	0.275	0.327	1.126	0.000	0.016
2	4	400 × 600	1845	1.221	8.88	0.131	0.116	0.177	0.171	0.582	0.000	0.012
2	8	400 × 600	1845	0.557	19.46	0.028	0.062	0.069	0.081	0.275	0.000	0.009
1	1	800 × 1200	3706	87.152	1.00	0.000	16.807	10.005	7.589	52.496	0.005	0.160
4	1	800 × 1200	3704	13.390	6.51	1.482	1.137	1.687	2.109	6.539	0.050	0.134
4	2	800 × 1200	3704	8.175	10.66	0.384	0.889	1.091	1.280	4.397	0.001	0.039
4	4	800 × 1200	3704	4.402	19.80	0.296	0.472	0.590	0.663	2.260	0.000	0.016
4	8	800 × 1200	3704	2.607	33.43	0.341	0.251	0.326	0.381	1.183	0.000	0.012
20	6	800 × 1200	3704	2.621	33.25	0.970	0.158	0.316	0.282	0.802	0.000	0.040

Таблица 3: Детальный анализ работы гибридной MPI+OpenMP программы на ПВС IBM Polus в секундах. Для 1 MPI процесса и 1 OpenMP нити взяты результаты исходной последовательной программы. Время посчитано как среднее времен всех MPI процессов.

Программа	Число итераций	Полное время работы	Ускорение	MPI коммуникация	CPU/GPU обмен	GPU управление памятью	calcScaledProd цикл	calcScaledAdd цикл	calcZ цикл	ApplyA цикл	D цикл	a, b, F цикл
Последовательная	3706	87.152	1.00	0.000	0.000	0.000	16.807	10.005	7.589	52.496	0.005	0.160
OpenMP 4 нити	3706	22.510	3.87	0.000	0.000	0.000	4.261	2.641	1.894	13.653	0.002	0.036
OpenMP 8 нитей	3706	11.378	7.66	0.000	0.000	0.000	2.186	1.316	0.982	6.855	0.001	0.019
OpenMP 16 нитей	3706	6.058	14.39	0.000	0.000	0.000	1.194	0.767	0.519	3.556	0.001	0.009
MPI 4 процесса	3706	19.366	4.50	1.059	0.000	0.000	4.734	2.105	2.494	8.877	0.002	0.038
MPI 8 процессов	3706	9.952	8.76	0.962	0.000	0.000	2.334	0.995	1.225	4.362	0.001	0.019
MPI 16 процессов	3706	5.468	15.94	0.839	0.000	0.000	1.202	0.493	0.632	2.251	0.001	0.011
MPI 20 процессов	3706	4.561	19.11	0.767	0.000	0.000	0.978	0.403	0.521	1.845	0.001	0.009
MPI 32 процесса	3705	2.799	31.13	0.504	0.000	0.000	0.597	0.216	0.316	1.123	0.000	0.006
MPI+OpenMP 4x1	3704	13.390	6.51	1.482	0.000	0.000	1.137	1.687	2.109	6.539	0.050	0.134
MPI+OpenMP 4x2	3704	8.175	10.66	0.384	0.000	0.000	0.889	1.091	1.280	4.397	0.001	0.039
MPI+OpenMP 4x4	3704	4.402	19.80	0.296	0.000	0.000	0.472	0.590	0.663	2.260	0.000	0.016
MPI+OpenMP 4x8	3704	2.607	33.43	0.341	0.000	0.000	0.251	0.326	0.381	1.183	0.000	0.012
MPI+CUDA 1 GPU	3704	8.638	10.09	0.031	0.348	1.731	3.595	1.363	0.336	0.873	0.000	0.000
MPI+CUDA 2 GPU	3704	5.998	14.53	0.117	0.299	1.540	2.444	0.669	0.199	0.471	0.000	0.000

Таблица 4: Детальное сравнение программ на ПВС IBM Polus в секундах.
Размер сетки 800×1200 .

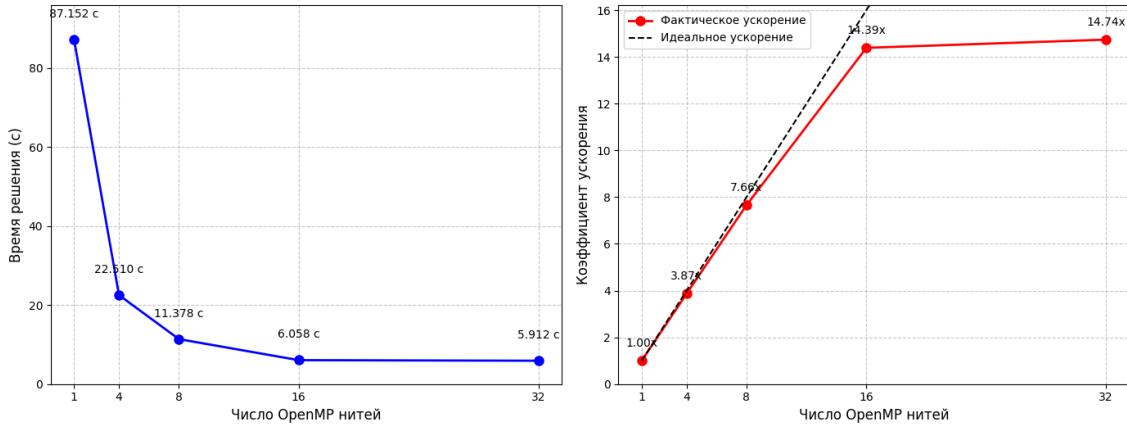


Рис. 1: Анализ времени работы OpenMP-программы.
Размер сетки 800×1200 , количество итераций 4764.

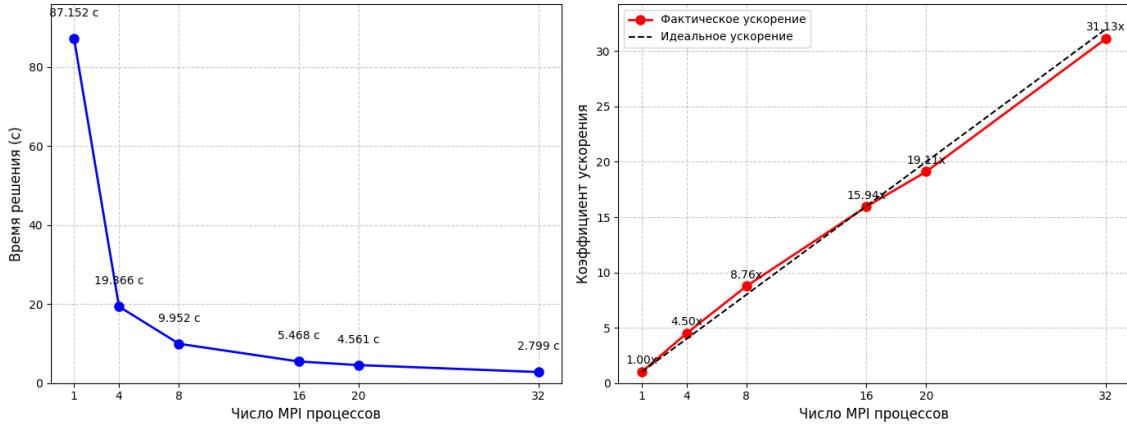


Рис. 2: Анализ времени работы MPI-программы.
Размер сетки 800×1200 , количество итераций 4775.

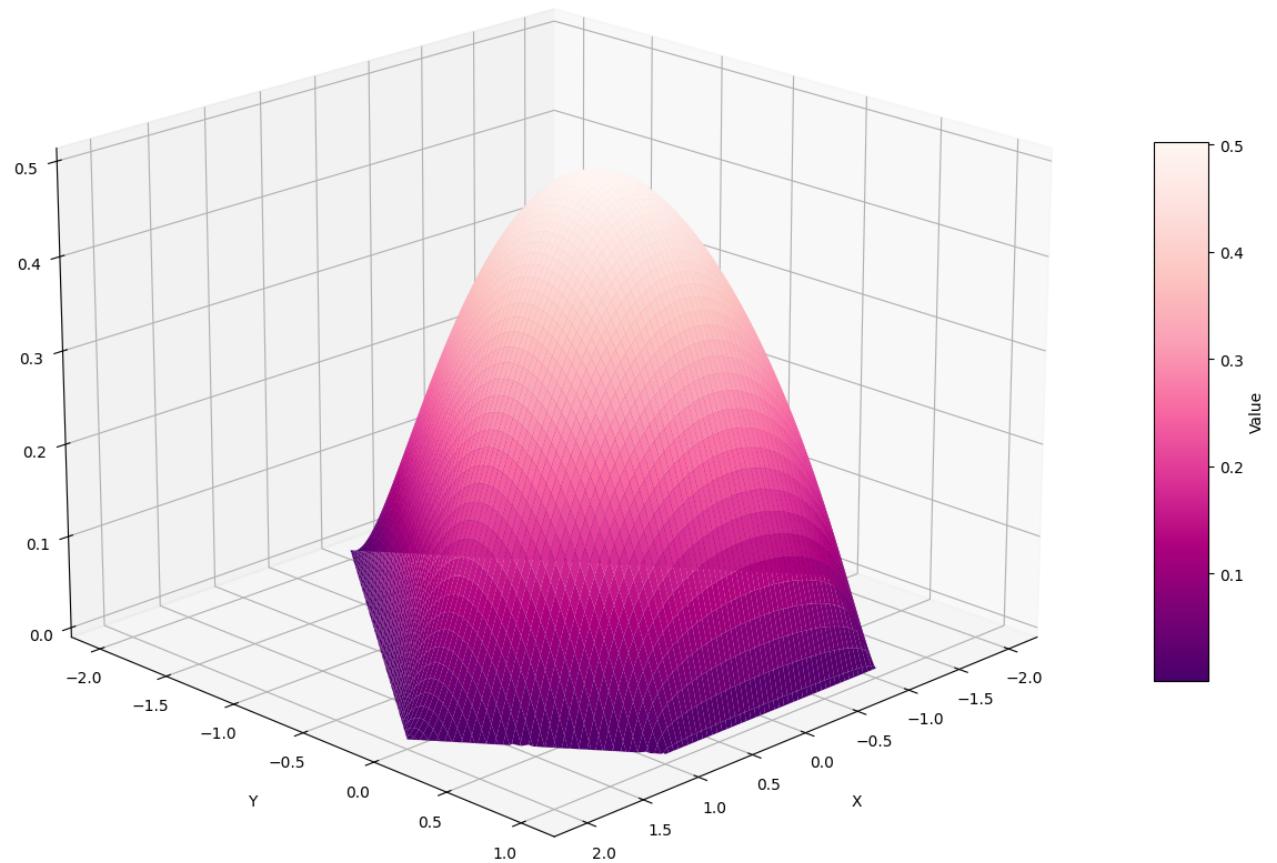


Рис. 3: Приближенное решение на сетке 800×1200 .

5 О результатах

5.1 Команды компиляции программ

- Последовательная:

```
g++ original.cpp -std=c++11 -o original
```

- OpenMP:

```
g++ omp.cpp -O3 -fopenmp -std=c++11 -o omp
```

- MPI:

```
module load SpectrumMPI/10.1.0
mpicxx mpi.cpp -O3 -std=c++11 -o mpi
```

- MPI+OpenMP:

```
module load SpectrumMPI/10.1.0
mpicxx mpi_omp.cpp -fopenmp -O3 -std=c++11 -o mpi_omp
```

- MPI+CUDA:

```
module load OpenMPI/4.0.0
nvcc -arch=sm_35 -O3 -ccbin=mpicc --std=c++11 main.cu \
cuda_kernels.cu -o mpi_cuda -lm -lstdc++
```

5.2 Оценка корректности

Оценка корректности работы программы происходит в функции `cout_discrepancy`, которая считает максимальный модуль элементов конечной невязки. Во всех запусках это значение не больше 1e-10.

5.3 Анализ результатов

- Анализ результатов OpenMP-программы:

- Эффективность параллельной работы зависит от размера сетки: на маленькой сетке 400×600 рост числа потоков выше 16 не дает линейного прироста ускорения. Для большой сетки 800×1200 добавление потоков до 32 все еще дает небольшой выигрыш (ускорение возрастает с 14.39x до 14.74x), так как вычислительная нагрузка достаточна для эффективного распределения между потоками.
- Анализ отдельных циклов показывает, что наиболее ресурсоемким является `ApplyA`, на который приходится значительная доля времени работы (около 60% на последовательной версии). Остальные циклы хорошо распараллеливаются, что объясняет заметное снижение их времени при увеличении числа потоков.
- В целом, результаты соответствуют прогнозируемым закономерностям: ускорение растет с увеличением числа потоков, но не линейно из-за накладных расходов и ограничений по параллельным частям кода.

- Анализ результатов MPI-программы:

- Эффективность параллельной работы заметно возрастает с увеличением числа процессов. При переходе от 1 до 4 процессов достигается ускорение 4.50x, что близко к линейному масштабированию.
- Самые ресурсоемкие операции (`ApplyA`, `calcScalarProd`) по-прежнему занимают значительную долю времени. Однако с ростом числа процессов их время делится почти пропорционально, что и обеспечивает высокое ускорение.
- Время MPI-коммуникаций остается относительно стабильным (0.5-1.0 сек) и составляет от 5% при 4 процессах до 18% при 32 процессах от общего времени выполнения.
- В целом, MPI-программа демонстрирует почти линейное ускорение при малом и среднем числе процессов. На больших числах процессов (32) достигается впечатляющее ускорение 31.13x, что свидетельствует о высокой масштабируемости для крупных сеток.

- Анализ программы MPI+OpenMP:

- Комбинация MPI и OpenMP продемонстрировала превосходство над чисто MPI-реализацией. Гибридный подход позволил достичь лучшего ускорения при меньшем числе MPI процессов, что особенно заметно на крупных сетках.
- Увеличение числа OpenMP-нитей при фиксированном количестве MPI процессов (2 или 4) демонстрирует хорошую масштабируемость, близкую к линейной на малых числах нитей. Конфигурация 2×8 достигает ускорения 19.46x, что сопоставимо с 16 MPI процессами (15.94x).
- Добавление OpenMP-директив активирует дополнительные оптимизации компилятора, включая векторизацию циклов, что обеспечивает ускорение даже при использовании одной нити по сравнению с чисто MPI-версией.
- Небольшая вариативность в числе итераций между запусками (от 3704 до 3708) является нормальным следствием накопления погрешностей округления в параллельных вычислениях и не влияет на точность конечного решения.
- Результат конфигурации 4×8 показал незначительное преимущество над 20×6 (2.607 сек против 2.621 сек), однако конфигурация 20×6 имеет почти втрое большее время MPI-коммуникаций (0.970 сек против 0.341 сек), составившее 37% общего времени выполнения. Это демонстрирует, что при увеличении числа MPI процессов достигается предел эффективного масштабирования для задачи данного размера: дальнейшее дробление домена приводит к критическому росту накладных расходов на межпроцессные обмены, который практически полностью нивелирует выигрыш от дополнительного параллелизма.
- Наилучший результат среди гибридных конфигураций показала 4×8 с ускорением 33.43x, превзойдя как чистый MPI (31.13x при 32 процессах), так и чистый OpenMP (14.74x при 32 нитях).

- Анализ программы MPI+CUDA:

- Использование GPU вместе с MPI значительно сокращает общее время работы по сравнению с чистыми MPI и OpenMP вариантами. Конфигурация с 1 GPU достигает ускорения 10.09x, а с 2 GPU – 14.53x.
- Основная часть времени приходится на вычислительные ядра GPU (`calcScalarProd`, `calcScaledAdd`, `calcZ`, `ApplyA`), в то время как затраты на коммуникацию (MPI + CPU/GPU обмен) составляют 4-7% от общего времени, что свидетельствует о высокой эффективности параллельной реализации.

- Обе конфигурации GPU сошлись за одинаковое число итераций (3704), что свидетельствует о численной стабильности алгоритма при использовании различного числа устройств. Конечная точность решения остается в пределах заданной погрешности.
- Управление памятью GPU составляет значительную долю времени (около 20% от общего времени), что является характерной особенностью GPU-вычислений и связано с необходимостью выделения и освобождения памяти между итерациями.
- В целом MPI+CUDA демонстрирует хорошие показатели для крупных сеток, обеспечивая сопоставимое с 16 OpenMP нитями ускорение (14.39x) при использовании всего 2 GPU. Дальнейшая оптимизация управления памятью GPU могла бы привести к еще более высокой производительности.