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

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

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

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 условием Дирихле:

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

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:

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

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 $\overline{D}$, $\overline{\Pi}$ 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

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

с кусочно-постоянным коэффициентом

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

и правой частью

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

Требуется найти непрерывную в $\overline{\Pi}$ 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

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

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,

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

В частности, $|v(x, y)| < C\varepsilon$ во всех точках кривой γ. Этот результат позволяет получить искомую 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 $\overline{\omega}_h = \overline{\omega}_1 \times \overline{\omega}_2$, где

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

Здесь $h_1 = (B_1 - A_1)/M, \quad h_2 = (B_2 - A_2)/N$. Через ω_h обозначим множество внутренних узлов сетки $\overline{\omega}_h$, 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

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

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

$$Aw = B, \tag{9}$$

где $A : H \to H$ – 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 $\overline{\omega}_h$ 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 аппроксимируется разностным уравнением

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

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

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

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

$$x_{i\pm1/2} = x_i \pm 0.5h_1, \quad y_{j\pm1/2} = y_j \pm 0.5h_2.$$

Правая часть разностного уравнения

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

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:

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

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

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

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 $Aw = -\left(aw_{\bar{x}}\right)_x - \left(bw_{\bar{y}}\right)_y$ 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 $\Pi_{ij} \cap D$, $S_{ij} = mes(\Pi_{ij} \cap D)$ – 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 $D : H \to H$ действует на сеточные функции $w \in H$ по правилу

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

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

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

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:

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

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

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

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

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

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 $w^{(k)}, \quad k = 0, 1, 2, \ldots$ 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 $H(w^{(k)}), k = 0, 1, 2, \ldots$ 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

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

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

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

Определим равномерную прямоугольную сетку

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

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:

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

Аналогично получаем l_{ij}^b:

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

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 $f(x,y)$ вырождается в

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

Для вычисления площади достаточно воспользоваться самым простым методом - методом средних прямоугольников. Подсчет высоты пересечения аналогичен формуле (19).
Как показала практика, для получения высокой точности ($\|r\|_C \le$ 1e-14 на сетках размером до 800×1200 достаточно взять δ из (17), равной 1e-20 и при подсчете площади пересечения делить ось x всего на 10 отрезков.

3.1.2 Алгоритм двумерного разбиения прямоугольника

Дана прямоугольная расчетная область П, на которой задана равномерная прямоугольная сетка: M узлов по оси x с индексами $i = 0, 1, ..., M-1$ и N узлов по оси y с индексами $j = 0, 1, ..., N-1$. 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:

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

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

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

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

Идеальная непрерывная оценка

Пусть M_p, N_p вещественные для всех p. Введем $P_x, P_y \in \mathbb{R}_+$ - число доменов по оси 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 оси x и y соответственно. Формулы для оси x (для y аналогично):

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

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 8x6, 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 распараллеливаемых частей кода.
2. Добавление нужных директив во все распараллеливаемые 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 прямогоульные 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+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	2463	144.507	1x	33.038	40.815	15.551	54.912	0.011	0.120
2	400×600	2463	76.196	1.9x	17.278	21.549	8.231	29.023	0.006	0.062
4	400×600	2463	37.286	3.88x	8.442	10.528	4.031	14.206	0.003	0.031
8	400×600	2463	19.315	7.48x	4.371	5.452	2.081	7.347	0.002	0.016
16	400×600	2463	10.211	14.15x	2.314	2.866	1.093	3.884	0.001	0.008
1	800×1200	4764	1054.734	1x	330.000	400.000	160.000	540.000	0.100	1.000
4	800×1200	4764	284.295	3.71x	64.397	80.557	30.625	108.448	0.012	0.122
8	800×1200	4764	149.763	7.04x	33.977	42.319	16.179	57.092	0.006	0.065
16	800×1200	4764	76.334	13.81x	17.384	21.539	8.303	28.948	0.003	0.032
32	800×1200	4764	55.302	19.06x	12.380	15.728	6.126	20.902	0.002	0.027

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	Коммуникация	calcScalarProd цикл	calcScaledAdd цикл	calcZ цикл	ApplyA цикл	D цикл	a, b, F цикл
1	400×600	2463	144.507	1x	0.000	33.038	40.815	15.551	54.912	0.011	0.120
2	400×600	2464	78.255	1.84x	0.298	17.982	22.314	8.102	29.439	0.006	0.059
4	400×600	2464	40.885	3.53x	1.226	9.149	11.341	4.130	14.943	0.004	0.039
8	400×600	2464	21.464	6.73x	1.595	4.576	5.679	2.062	7.482	0.002	0.020
16	400×600	2464	11.521	12.53x	1.294	2.349	2.918	1.059	3.844	0.001	0.011
1	800×1200	4764	1054.734	1x	0.000	330.000	400.000	160.000	540.000	0.100	1.000
4	800×1200	4775	328.600	3.21x	12.981	72.752	90.409	32.758	119.398	0.015	0.155
8	800×1200	4775	184.806	5.71x	21.280	37.655	46.891	16.975	61.781	0.009	0.108
16	800×1200	4775	82.995	12.71x	4.033	18.207	22.527	8.268	29.801	0.006	0.054
32	800×1200	4775	53.449	19.73x	12.451	9.431	11.708	4.259	15.475	0.003	0.027

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

Программа	Число итераций	Полное время работы	Ускорение	Коммуникация	CPU/GPU обмен	calcScaledProd цикл	calcScaledAdd цикл	calcZ цикл	ApplyA цикл	D цикл	a, b, F цикл
Последовательная	4764	1054.734	1x	0.000	0.000	330.0	400.0	160.0	540.0	0.100	1.000
OpenMP 4 нити	4764	284.295	3.71x	0.000	0.000	64.397	80.557	30.625	108.448	0.012	0.122
OpenMP 8 нитей	4764	149.763	7.04x	0.000	0.000	33.977	42.319	16.179	57.092	0.006	0.065
OpenMP 16 нитей	4764	76.334	13.81x	0.000	0.000	17.384	21.539	8.303	28.948	0.003	0.032
MPI 4 процесса	4775	328.600	3.21x	12.981	0.000	72.752	90.409	32.758	119.398	0.015	0.155
MPI 8 процессов	4775	184.806	5.71x	21.280	0.000	37.655	46.891	16.975	61.781	0.009	0.108
MPI 16 процессов	4775	82.995	12.71x	4.033	0.000	18.207	22.527	8.268	29.801	0.006	0.054
MPI+CUDA 1 GPU	4764	136.230	7.74x	0.456	98.234	4.127	5.003	2.001	6.753	0.001	0.003
MPI+CUDA 2 GPU	4764	71.391	14.77x	1.892	51.456	2.158	2.614	1.045	3.531	0.001	0.002

Таблица 3: Детальное сравнение программ на ПВС IBM Polus в секундах. Размер сетки 800×1200.

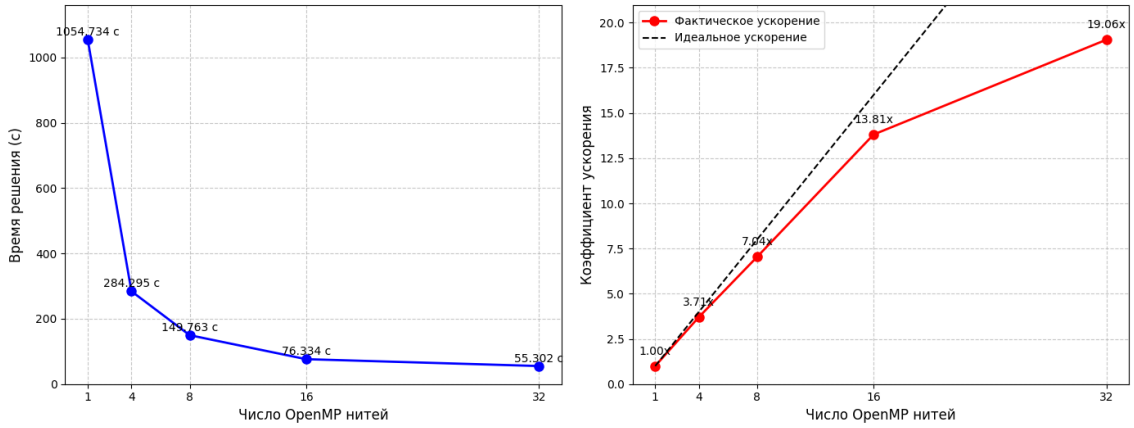

Рис. 1: Анализ времени работы OpenMP-программы.
Размер сетки 800 × 1200, количество итераций 4764.

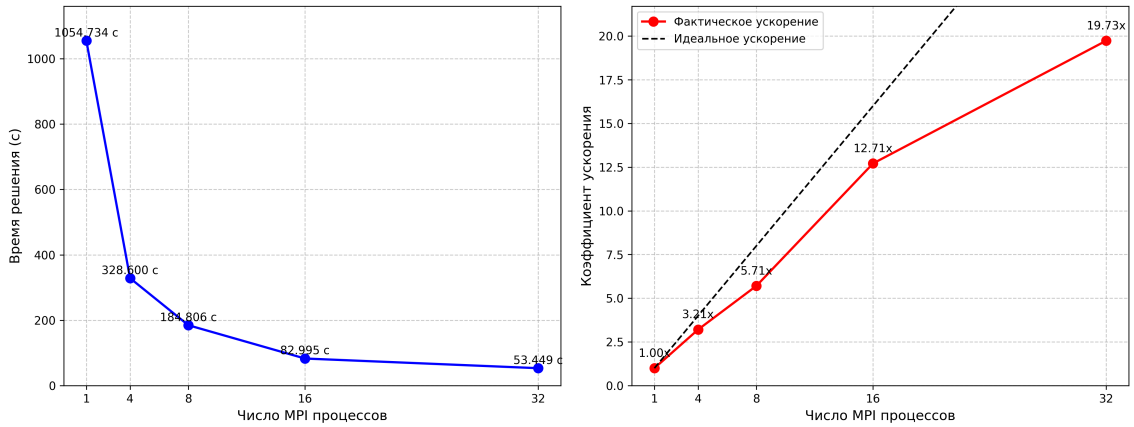

Рис. 2: Анализ времени работы MPI-программы.
Размер сетки 800 × 1200, количество итераций 4775.

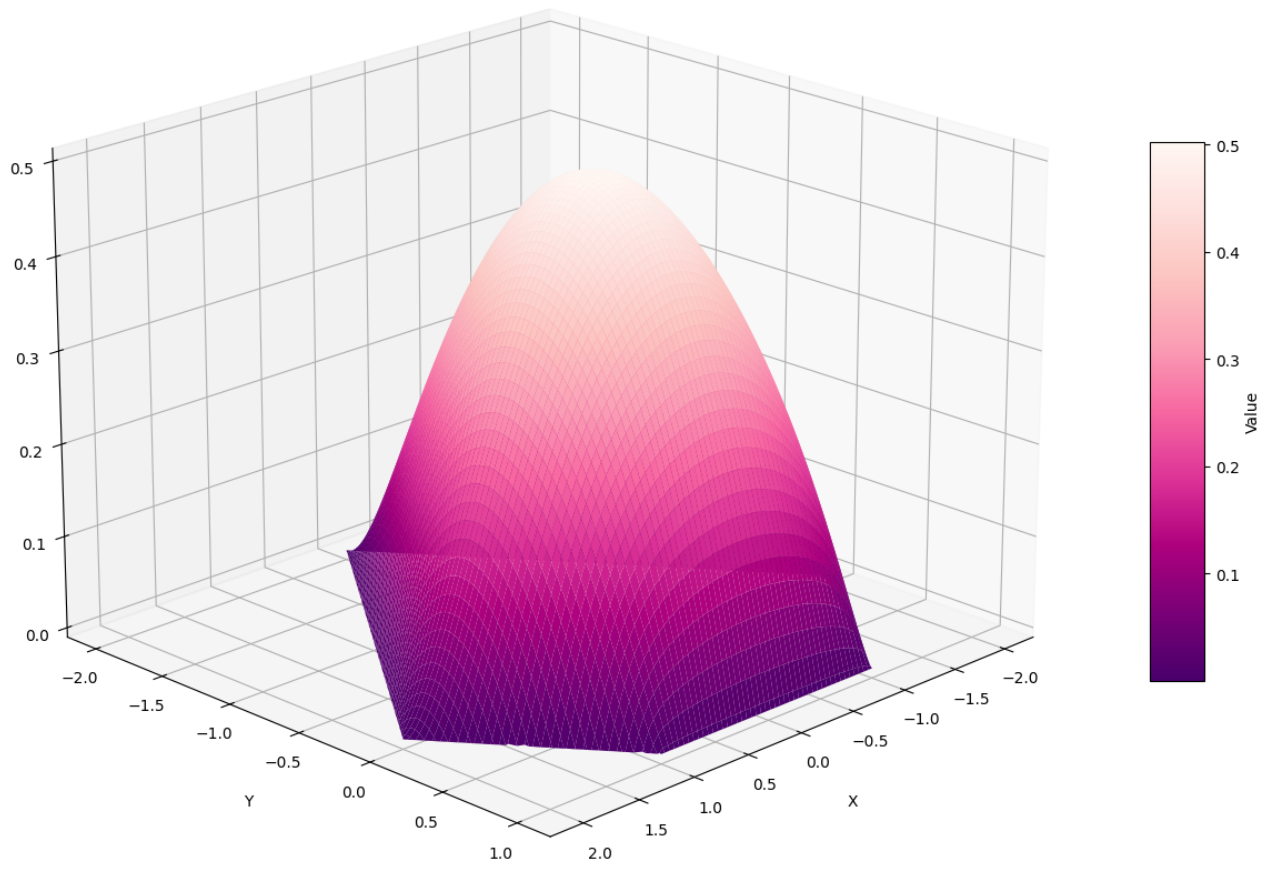

Рис. 3: Приближенное решение на сетке 800×1200.

5 О результатах

5.1 Команды компиляции программ

- Последовательная:

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

- OpenMP:

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

- MPI:

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

- MPI+CUDA:

```
module load OpenMPI/4.0.0
nvcc -arch=sm_35 -O0 -g -G -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-14.

5.3 Анализ результатов

- Для программы с использованием OpenMP наблюдается почти линейное ускорение при увеличении числа нитей от 1 до 8. Например, для сетки 400×600 ускорение с 1 до 8 нитей составляет около $7.48\times$. При увеличении числа нитей до 16 ускорение растет менее линейно ($14.15\times$) из-за накладных расходов на синхронизацию и управление потоками.

- Для программы с использованием MPI ускорение также увеличивается с числом процессов, но менее линейно, особенно на больших сетках (800×1200 ускорение $19.73\times$ при 32 процессах). Основная причина — коммуникационные затраты между процессами.

- Сравнение OpenMP и MPI показывает, что на малом числе потоков OpenMP эффективнее, так как отсутствует межпроцессное взаимодействие. MPI становится более полезным при больших объемах данных и числах процессов, хотя коммуникации снижают идеальное ускорение.

- Использование GPU (MPI+CUDA) обеспечивает значительное ускорение: $7.74\times$ для 1 GPU и $14.77\times$ для 2 GPU по сравнению с последовательной программой. Основной выигрыш достигается в вычислительных циклах (`calcScaledProd`, `calcScaledAdd`, `calcZ`, `ApplyA`).

- Основные причины отклонений от идеального ускорения:

  - OpenMP: снижение эффективности при большом числе нитей из-за синхронизации и конкуренции за кэш и память.
  - MPI: коммуникационные издержки между процессами, особенно для больших сеток.
  - MPI+CUDA: накладные расходы на передачу данных и синхронизацию.

- Вычислительные циклы доминируют по времени выполнения и приносят основной выигрыш при параллельной обработке. Малые циклы (`D` и `a, b, F`) практически не влияют на общее ускорение.