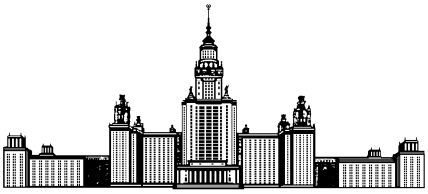

Московский государственный университет имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра математических методов прогнозирования

Задание по курсу «Суперкомпьютерное моделирование и технологии»

Решение краевой задачи для уравнения Пуассона методом конечных разностей.

Выполнил:

студент 617 группы

Г.В. Кормаков

Москва, 2021

Содержание

1 Постановка задачи

В прямоугольнике $\Pi = \{(x, y) : A_1 \leqslant x \leqslant A_2, B_1 \leqslant y \leqslant B_2\}$, граница Γ которого состоит из отрезков

$$\gamma_R = \{(A_2, y), B_1 \leqslant y \leqslant B_2\}, \quad \gamma_L = \{(A_1, y), B_1 \leqslant y \leqslant B_2\}$$
$$\gamma_T = \{(x, B_2), A_1 \leqslant x \leqslant A_2\}, \quad \gamma_B = \{(x, B_1), A_1 \leqslant x \leqslant A_2\}$$

рассматривается дифференциальное уравнение Пуассона с потенциалом

$$-\Delta u + q(x, y)u = F(x, y) \tag{1}$$

в котором оператор Лапласа

$$\Delta u = \frac{\partial}{\partial x}\left(k(x, y)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(k(x, y)\frac{\partial u}{\partial y}\right) \tag{2}$$

Для выделения единственного решения уравнение дополняется граничными условиями. На каждом отрезке границы прямоугольника Π задается условие одним из двух способов - условиями Дирихле (1-ого типа) или условиями второго (Неймана) и третьего типа.

Для выданного варианта 8 задания краевые условия задаются следующими условиями: третьего типа на правой и левой границе и второго на верхней и нижней на сетке (см. раздел 2). Общая формула условий третьего типа выглядит следующим образом:

$$\left(k\frac{\partial u}{\partial n}\right)(x, y) + \alpha u(x, y) = \psi(x, y), \tag{3}$$

где n - единичная внешняя нормаль к границе прямоугольника.

Краевое условие второго типа (условие Неймана) содержится в краевом условии третьего типа (случай $\alpha = 0$ в 3).

Функции $F(x, y), \varphi(x, y), \psi(x, y)$, коэффициент $k(x, y)$, потенциал $q(x, y)$ и параметр $\alpha \geqslant 0$ считаются известными, функцию $u(x, y)$, удовлетворяющую уравнению 1 и граничным условиям, определенным вариантом задания 8, требуется найти.

Важно отметить, что нормаль n не определена в угловых точках прямоугольника. Краевое условие третьего типа будет рассматриваться лишь в тех точках границы, где нормаль существует.

2 Численный метод решения

2.1 Общая схема метода конечных разностей

В качестве метода решения задачи Пуассона 1 с потенциалом предлагается использовать метод конечных разностей.

Для этого область П дискретизуем сеткой $\overline{\omega}_h = \overline{\omega}_1 \times \overline{\omega}_2$, где $\overline{\omega}_1$ – разбиение сетки по оси Ox с шагом $h_1 = \frac{A_2 - A_1}{M}$ и $\overline{\omega}_2$ – разбиение сетки по оси Oy с шагом $h_2 = \frac{B_2 - B_1}{N}$:

$$\overline{\omega}_1 = \left\{x_i = A_1 + ih_1, i = \overline{0, M}\right\}, \overline{\omega}_2 = \left\{y_j = B_1 + jh_2, j = \overline{0, N}\right\}$$

Также примем обозначение ω_h для внутренних узлов сетки $\overline{\omega}_h$

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

$$[u, v] = \sum_{i=0}^{M} h_1 \sum_{j=0}^{N} h_2 \rho_{ij} u_{ij} v_{ij}, \quad \|u\|_E = \sqrt{[u, u]},$$

где ρ_{ij} – весовая функция $\rho_{ij} = \rho^{(1)}(x_i)\,\rho^{(2)}(y_j)$ с

$$\rho^{(1)}(x_i) = \begin{cases} 1, & 1 \leqslant i \leqslant M-1 \\ 1/2, & i = 0, i = M \end{cases} \qquad \rho^{(2)}(y_j) = \begin{cases} 1, & 1 \leqslant j \leqslant N-1 \\ 1/2, & j = 0, j = N \end{cases}$$

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

$$Aw = B \tag{4}$$

где $A : H \to H$ – оператор, действующий в пространстве сеточных функций, $B \in H-$ известная правая часть. Задача 4 называется разностной схемой. Решение этой задачи считается численным решением исходной дифференциальной задачи.

2.2 Конкретный вид разностной схемы для варианта 8

Уравнение 1 приближается для всех внутренних точек ω_h сетки разностной схемой следующего вида:

$$-\Delta_h w_{ij} + q_{ij} w_{ij} = F_{ij}, \quad i = \overline{1, M-1}, j = \overline{1, N-1},$$

в котором $F_{ij} = F(x_i, y_j)$, $q_{ij} = q(x_i, y_j)$ и разностный оператор Лапласа

$$\Delta_h w_{ij} = \frac{1}{h_1}\left(k(x_i + 0.5h_1, y_j)\frac{w_{(i+1)j} - w_{ij}}{h_1} - k(x_i - 0.5h_1, y_j)\frac{w_{ij} - w_{(i-1)j}}{h_1}\right) +$$
$$+\frac{1}{h_2}\left(k(x_i, y_j + 0.5h_2)\frac{w_{i(j+1)} - w_{ij}}{h_2} - k(x_i, y_j - 0.5h_2)\frac{w_{ij} - w_{i(j-1)}}{h_2}\right)$$

В разностном операторе Лапласа введём обозначения для правых и левых разностных производных по координатам:

$$w_{x,ij} \equiv \frac{w_{(i+1)j} - w_{ij}}{h_1}, \quad w_{\overline{x},ij} \equiv w_{x,(i-1)j} = \frac{w_{ij} - w_{(i-1)j}}{h_1}$$
$$w_{y,ij} \equiv \frac{w_{i(j+1)} - w_{ij}}{h_2}, \quad w_{\overline{y},ij} \equiv w_{y,i(j-1)} \equiv \frac{w_{ij} - w_{i(j-1)}}{h_2}$$

и зададим сеточные коэффициенты

$$a_{ij} \equiv k(x_i - 0.5h_1, y_j), \quad b_{ij} \equiv k(x_i, y_j - 0.5h_2).$$

Тогда разностный оператор Лапласа равен

$$\Delta_h w_{ij} = \frac{1}{h_1}\left(k(x_i + 0.5h_1, y_j)\, w_{x,ij} - k(x_i - 0.5h_1, y_j)\, w_{x,(i-1)j}\right) +$$
$$+\frac{1}{h_2}\left(k(x_i, y_j + 0.5h_2)\, w_{y,ij} - k(x_i, y_j - 0.5h_2)\, w_{y,i(j-1)}\right) =$$
$$= \frac{a_{(i+1)j} w_{x,ij} - a_{ij} w_{x,(i-1)j}}{h_1} + \frac{b_{i(j+1)} w_{y,ij} - b_{ij} w_{y,i(j-1)}}{h_2} =$$
$$= \frac{a_{(i+1)j} w_{\overline{x},(i+1)j} - a_{ij} w_{\overline{x},ij}}{h_1} + \frac{b_{i(j+1)} w_{\overline{y},i(j+1)} - b_{ij} w_{\overline{y},ij}}{h_2} \equiv (a w_{\overline{x}})_{x,ij} + (b w_{\overline{y}})_{y,ij} \qquad (5)$$

Итого, получаем $(M-1)\cdot(N-1)$ уравнений во внутренних точках:

$$-\Delta_h w_{ij} + q_{ij} w_{ij} = F_{ij}, \quad i = \overline{1, M-1}, j = \overline{1, N-1} \qquad (6)$$

Рассмотрим конкретные граничные условия, заданные в варианте 8 (см. таблицу 1).

Для правой (схема 7) и левой (схема 8) границы (γ_R, γ_L соответственно) задаются условия третьего типа. Разностный вариант (с учётом членов для соответствия порядка погрешности аппроксимации с основным уравнением 5) для них выглядит следующим образом:

$$\frac{2}{h_1}(a w_{\overline{x}})_{Mj} + \left(q_{Mj} + \frac{2\alpha_R}{h_1}\right) w_{Mj} - (b w_{\overline{y}})_{y,Mj} = F_{Mj} + \frac{2}{h_1}\psi^{(R)}_{Mj}, \quad j = \overline{1, N-1} \qquad (7)$$
$$-\frac{2}{h_1}(a w_{\overline{x}})_{1j} + \left(q_{0j} + \frac{2\alpha_L}{h_1}\right) w_{0j} - (b w_{\overline{y}})_{y,0j} = F_{0j} + \frac{2}{h_1}\psi^{(L)}_{0j}, \quad j = \overline{1, N-1} \qquad (8)$$

Для верхней (схема 9) и нижней (схема 10) границы (γ_T, γ_B соответственно) задаются условия второго типа (Неймана).

$$\frac{2}{h_2}(b w_{\overline{y}})_{iN} + q_{iN} w_{iN} - (a w_{\overline{x}})_{x,iN} = F_{iN} + \frac{2}{h_2}\psi^{(T)}_{iN}, \quad i = \overline{1, M-1} \qquad (9)$$
$$-\frac{2}{h_2}(b w_{\overline{y}})_{i1} + q_{i0} w_{i0} - (a w_{\overline{x}})_{x,i0} = F_{i0} + \frac{2}{h_2}\psi^{(B)}_{i0}, \quad i = \overline{1, M-1} \qquad (10)$$

Также в угловых точках не определена нормаль, поэтому необходимо их учесть отдельными уравнениями.

Для точки $P(A_1, B_1)$ прямоугольника Π

$$-\frac{2}{h_1}\left(aw_{\overline{x}}\right)_{10} - \frac{2}{h_2}\left(bw_{\overline{y}}\right)_{01} + \left(q_{00} + \frac{2\alpha_L}{h_1}\right) w_{00} = F_{00} + \left(\frac{2}{h_1} + \frac{2}{h_2}\right)\psi_{00} \tag{11}$$

Для точки $P(A_2, B_1)$ прямоугольника Π

$$\frac{2}{h_1}\left(aw_{\overline{x}}\right)_{M0} - \frac{2}{h_2}\left(bw_{\overline{y}}\right)_{M1} + \left(q_{M0} + \frac{2\alpha_R}{h_1}\right) w_{M0} = F_{M0} + \left(\frac{2}{h_1} + \frac{2}{h_2}\right)\psi_{M0} \tag{12}$$

Для точки $P(A_2, B_2)$ прямоугольника Π

$$\frac{2}{h_1}\left(aw_{\overline{x}}\right)_{MN} + \frac{2}{h_2}\left(bw_{\overline{y}}\right)_{MN} + \left(q_{MN} + \frac{2\alpha_R}{h_1}\right) w_{MN} = F_{MN} + \left(\frac{2}{h_1} + \frac{2}{h_2}\right)\psi_{MN} \tag{13}$$

Для точки $P(A_1, B_2)$ прямоугольника Π

$$-\frac{2}{h_1}\left(aw_{\overline{x}}\right)_{1N} + \frac{2}{h_2}\left(bw_{\overline{y}}\right)_{0N} + \left(q_{0N} + \frac{2\alpha_L}{h_1}\right) w_{0N} = F_{0N} + \left(\frac{2}{h_1} + \frac{2}{h_2}\right)\psi_{0N} \tag{14}$$

Уточним, что обозначили за $\psi^{(T)}, \psi^{(B)}$ фунции краевых условий второго типа (для данного варианта), а за $\psi^{(R)}, \psi^{(L)}$ – функции краевых условий третьего типа. В угловых точках вектор нормали не определён, поэтому краевые условия равны значению функции $u(x, y)$.

Вариант задания	Граничные условия				Решение u(x, y)	Коэфф. $k(x, y)$	Потенциал $q(x, y)$
	γ_R	γ_L	γ_T	γ_B			
8	3 тип	3 тип	2 тип	2 тип	$u_2(x, y)$	$k_3(x, y)$	$q_2(x, y)$

Таблица 1: Условия задания

Запишем **финальную систему**, убедившись, что она определена.

$$-\left(aw_{\overline{x}}\right)_{x,ij} - \left(bw_{\overline{y}}\right)_{y,ij} + q_{ij}w_{ij} = F_{ij}, i = \overline{1, M-1}, j = \overline{1, N-1}$$

$$\frac{2}{h_1}\left(aw_{\overline{x}}\right)_{Mj} + \left(q_{Mj} + \frac{2\alpha_R}{h_1}\right)w_{Mj} - \left(bw_{\overline{y}}\right)_{y,Mj} = F_{Mj} + \frac{2}{h_1}\psi^{(R)}_{Mj},\ j = \overline{1, N-1}$$

$$-\frac{2}{h_1}\left(aw_{\overline{x}}\right)_{1j} + \left(q_{0j} + \frac{2\alpha_L}{h_1}\right)w_{0j} - \left(bw_{\overline{y}}\right)_{y,0j} = F_{0j} + \frac{2}{h_1}\psi^{(L)}_{0j},\ j = \overline{1, N-1}$$

$$\frac{2}{h_2}\left(bw_{\overline{y}}\right)_{iN} + q_{iN}w_{iN} - \left(aw_{\overline{x}}\right)_{x,iN} = F_{iN} + \frac{2}{h_2}\psi^{(T)}_{iN},\ i = \overline{1, M-1}$$

$$-\frac{2}{h_2}\left(bw_{\overline{y}}\right)_{i1} + q_{i0}w_{i0} - \left(aw_{\overline{x}}\right)_{x,i0} = F_{i0} + \frac{2}{h_2}\psi^{(B)}_{i0},\ i = \overline{1, M-1}$$

$$-\frac{2}{h_1}\left(aw_{\overline{x}}\right)_{10} - \frac{2}{h_2}\left(bw_{\overline{y}}\right)_{01} + \left(q_{00} + \frac{2\alpha_L}{h_1}\right)w_{00} = F_{00} + \left(\frac{2}{h_1} + \frac{2}{h_2}\right)\psi_{00}$$

$$\frac{2}{h_1}\left(aw_{\overline{x}}\right)_{M0} - \frac{2}{h_2}\left(bw_{\overline{y}}\right)_{M1} + \left(q_{M0} + \frac{2\alpha_R}{h_1}\right)w_{M0} = F_{M0} + \left(\frac{2}{h_1} + \frac{2}{h_2}\right)\psi_{M0}$$

$$\frac{2}{h_1}\left(aw_{\overline{x}}\right)_{MN} + \frac{2}{h_2}\left(bw_{\overline{y}}\right)_{MN} + \left(q_{MN} + \frac{2\alpha_R}{h_1}\right)w_{MN} = F_{MN} + \left(\frac{2}{h_1} + \frac{2}{h_2}\right)\psi_{MN}$$

$$-\frac{2}{h_1}\left(aw_{\overline{x}}\right)_{1N} + \frac{2}{h_2}\left(bw_{\overline{y}}\right)_{0N} + \left(q_{0N} + \frac{2\alpha_L}{h_1}\right)w_{0N} = F_{0N} + \left(\frac{2}{h_1} + \frac{2}{h_2}\right)\psi_{0N}$$

Получили $(N-1)(M-1) + 2(M-1) + 2(N-1) + 4 = MN - M - N + 1 + 2M + 2N - 4 + 4 = MN + N + M + 1$ уравнений. Число неизвестных w_{ij} равно $(M+1)(N+1) = MN + N + M + 1$. И система гарантирует единственность решения для данной разностной схемы.

Таким образом, матрица оператора A определяется коэффициентами перед неизвестными в левой части, а матрицы B – в правой.

2.3 Метод решения СЛАУ

Приближенное решение системы уравнений для сформулированных выше краевых задач может быть получено итерационным методом наименьших невязок. Этот метод позволяет получить последовательность сеточных функций $w^{(k)} \in H, k = 1, 2, \ldots$, сходящуюся по норме пространства H к решению разностной схемы, т.е.

$$\left\|w - w^{(k)}\right\|_E \xrightarrow[k \to +\infty]{} 0$$

Метод является одношаговым. Итерация обновления $w^{(k+1)}$ записывается в виде

$$w^{(k+1)}_{ij} = w^{(k)}_{ij} - \tau_{k+1} r^{(k)}_{ij},$$

где невязка $r^{(k)} = Aw^{(k)} - B$, итерационный параметр $\tau_{k+1} = \frac{\left[Ar^{(k)}, r^{(k)}\right]}{\left\|Ar^{(k)}\right\|^2_E}$

В качестве условия остановки используем неравенство $\left\|w^{(k+1)} - w^{(k)}\right\|_E < \varepsilon$, где ε - положительное число, определяющее точность итерационного метода. Константу ε для данной задачи предлагается взять равной 10^{-6}.

2.4 Вид функций

В таблице 1 приведены конкретные функции для данного варианта. Они задаются следующим образом

$$u_2(x,y) = \sqrt{4+xy}, \Pi = [0,4] \times [0,3] \tag{15}$$

$$k_3(x,y) = 4 + x + y \tag{16}$$

$$q_2(x,y) = \begin{cases} x+y, & x+y \geqslant 0 \\ 0, & x+y < 0 \end{cases} \tag{17}$$

3 Нахождение $F(x,y), \psi(x,y)$

Поскольку для численной реализации нам предлагается сравнить приближеннное решение с истинным, то необходимо найти аналитический вид функции $F(x,y)$ и краевых условий[1] для известного решения $u(x,y) = u_2(x,y) = \sqrt{4+xy}$, $\Pi = [0,4] \times [0,3]$, $k(x,y) = k_3(x,y) = 4 + x + y$ и потенциалом $q(x,y) = q_2(x,y) = \max(0, x+y)$. Для определённости, возьмём вектор нормали $\boldsymbol{n}$, направленным извне.

$$\frac{\partial u}{\partial x} = \frac{y}{2\sqrt{4+xy}}; \quad \frac{\partial u}{\partial y} = \frac{x}{2\sqrt{4+xy}};$$

$$\frac{\partial}{\partial x}\left(k(x,y)\frac{y}{2\sqrt{4+xy}}\right) = \frac{y}{2\sqrt{4+xy}} - k(x,y)\frac{y^2}{4(4+xy)^{3/2}}$$

Аналогично для $\dfrac{\partial}{\partial y}$.

[1] В случае варианта 8 необходимо найти вид только функции $\psi(x,y)$

Тогда

$$
\begin{aligned}
&F(x,y) = -\Delta u + q(x,y)u = \\
&= -\frac{\partial}{\partial x}\left(k(x,y)\frac{y}{2\sqrt{4+xy}}\right) - \frac{\partial}{\partial y}\left(k(x,y)\frac{x}{2\sqrt{4+xy}}\right) + q(x,y)u = \\
&= -\frac{y}{2u(x,y)} + k(x,y)\frac{y^2}{2u^3(x,y)} - \frac{x}{2u(x,y)} + k(x,y)\frac{x^2}{4u^3(x,y)} + q(x,y)u = \\
&= \frac{-2u^2y + ky^2 - 2u^2x + kx^2 + 4qu^4}{4u^3} = \\
&= \frac{-2(4+xy)(x+y) + (4+x+y)y^2 - 2(4+xy)x + (4+x+y)x^2 + 4qu^4}{4(4+xy)^{3/2}} = \\
&= \frac{x^3 - x^2(2y-4-y) - x(8+2y^2-y^2) + y(-8+4y+y^2) + 4q(4+xy)^2}{4(4+xy)^{3/2}} = \\
&= \frac{x^3 - x^2(y-4) - x(y^2+8) + y(y^2+4y-8) + 4\max(0,x+y)(4+xy)^2}{4(4+xy)^{3/2}}
\end{aligned} \tag{18}
$$

Приведём пример вычислений функции $\psi(x,y)$ для в «общем виде» (знак $\pm$ не значит обязательное наличие члена, в зависимости от направления, одна из компонент нормали может равняться 0).

$$
\begin{aligned}
&\psi(x,y) = \left(k\frac{\partial u}{\partial \boldsymbol{n}}\right)(x,y) + \alpha u(x,y) = -k\left(\pm\frac{y^2}{4u^3} \pm \frac{x^2}{4u^3}\right) + \alpha u = \\
&= \frac{-(4+x+y)(\pm x^2 \pm y^2) + 4\alpha(4+xy)^2}{4(4+xy)^{3/2}}
\end{aligned}
$$

Для γ_R, γ_L $\alpha = 1$, для γ_T, γ_B $\alpha = 0$. Также учтём знаки нормали к поверхности в соответствующих направлениях. Таким образом, в обозначениях уравнений 11 - 13

$$
\psi^{(R)}(x,y) = \frac{-y^2(4+x+y) + 4(4+xy)^2}{4(4+xy)^{3/2}};\ \psi^{(L)}(x,y) = \frac{y^2(4+x+y) + 4(4+xy)^2}{4(4+xy)^{3/2}} \tag{19}
$$

$$
\psi^{(T)}(x,y) = \frac{-x^2(4+x+y)}{4(4+xy)^{3/2}};\ \psi^{(B)}(x,y) = \frac{x^2(4+x+y)}{4(4+xy)^{3/2}} \tag{20}
$$

4 Описание программной реализации

4.1 Формализация требований на домены

Перед непосредственной реализацией важно понять, как осуществлять разбиение на блоки-домены Π_{ij}, соблюдая следующие условия:

1. отношение количества узлов по переменым x и y в каждом домене принадлежало диапазону $[1/2, 2]$

2. количество узлов по переменым x и y любых двух доменов отличалось не более, чем на единицу.

Условия в данной формулировке приведены в постановке задания. Уточним более корректно смысл каждого пункта[2]. Обозначим количество узлов сетки в одном домене $n_x(i)$ и $n_y(j)$ соответственно по x и y. Количество узлов, вообще говоря, зависит от числа процессоров p, выделенных на задачу, в приведённых обозначениях считаем, что размер зависит от числа процессов, выделенных линейно на каждую ось (далее будет объяснён алгоритм разбиения).

Первое условие утверждает, что для любого домена его форма должна быть похожа на прямоугольную или квадратную. Формальнее,

$$\frac{n_x(i)}{n_y(j)} \in [0.5, 2] \quad \forall \Pi_{ij} \tag{21}$$

Данная конфигурация позволяет минимизировать потери во времени на пересылки.

Второе условие даёт равномерность разбиения на домены. Т.е., вообще говоря, размеры доменов по оси x не должны отличаться более, чем на 1, и по y аналогично. Таким образом, $n_x(i), n_y(j)$, фактически должны быть константными, но из-за того, что размеры сетки могут быть не кратны числу процессоров, на граничных блоках мы получаем иные размеры. Именно на то, чтобы отличия в размерах были в этом случае минимальны и направлено ограничение 2.

Формальнее,

$$|n_x(i_1) - n_x(i_2)| \leqslant 1,\ |n_y(j_1) - n_y(j_2)| \leqslant 1,\ \forall (i_1, j_1), (i_2, j_2) \tag{22}$$

4.2 Алгоритм разбиения на блоки

Приведём алгоритм, удовлетворяющий условиям 21,22. Обозначим за p_x число процессоров, работающих в ячейках по оси x, и p_y – по y.

Для гарантированного выполнения условия 22 возьмём последовательность размеров блоков по каждой оси, отличающихся только на 1 точку. Т.е. будем чередовать по оси x домены с размером a_x и a_x+1 (по y соответственно обозначим за a_y и a_y+1).

[2]Корректность гарантирована объяснениями преподавателей на лекции, посвящённой постановке задачи

Таким образом, получаем следующие разложения

$$\sum_{i=1}^{p_x} n_x(i) = k_1 a_x + k_2(a_x + 1) = M,\ k_1 + k_2 = p_x \Rightarrow a_x p_x + k_2 = M \tag{23}$$

$$\sum_{j=1}^{p_y} n_y(j) = s_1 a_y + s_2(a_y + 1) = N,\ s_1 + s_2 = p_y \Rightarrow a_y p_y + s_2 = N \tag{24}$$

Условие 21 в новых обозначениях формулируется как $0.5 \leqslant \frac{a_x}{a_y} \leqslant 2$, т.к. в приведённой формулировке алгоритма чередоваться бóльшие блоки будут одновременно по осям.

Из формул 23, 24 видно, что необходимо поделить с остатком M и N на количество процессоров по осям. При этом не стоит забывать, что должно соблюдаться неравенство $p_x p_y \leqslant p$.

Предлагается следующая схема: берём $p_x = 2^k$ процессоров на ось x и $p_y = \lfloor \frac{p}{2^k} \rfloor$ процессоров на ось y.[3] Для выполнения условия 21 логичным кажется выбирать эти числа, исходя из пропорции 25.

$$\frac{2^k}{\frac{p}{2^k}} = \frac{M}{N} \Rightarrow \frac{2^{2k}}{p} = \frac{M}{N} \Rightarrow 2^{2k} = \frac{pM}{N} \Rightarrow k = \left\lfloor \frac{\log_2\left(\frac{pM}{N}\right)}{2} \right\rfloor \tag{25}$$

В случае $p = 2^n$ выражение 25 переходит в $k = \left\lfloor \frac{n + \log_2\left(\frac{M}{N}\right)}{2} \right\rfloor$.

Таким образом, итоговая схема получения блоков, удовлетворяющих условиям 21 (из-за выбора k согласно 25) и 22 (из-за выбора определённой последовательности), выглядит следующим образом:

1. Задаём $p_x = 2^{\left\lfloor \frac{\log_2\left(\frac{pM}{N}\right)}{2} \right\rfloor}$, $p_y = \left\lfloor \frac{p}{p_x} \right\rfloor$.

2. Получаем $a_x = \lfloor \frac{M}{p_x} \rfloor$, $a_y = \lfloor \frac{N}{p_y} \rfloor$ и фиксируем $k_2 = M \mod p_x$, $s_2 = N \mod p_y$

3. Далее генерируем $k_1 = p_x - k_2$ доменов по оси x с размером a_x и $s_1 = p_y - s_2$ доменов по оси y с размером a_y.

4. Затем начинается генерация k_2 доменов по оси x с размером $a_x + 1$ и s_2 доменов по оси y с размером $a_y + 1$

[3] Данная формула носит общий характер, в рамках текущего задания $p = 2^n$ и можно сразу сказать, что $p_y = 2^{n-k}$

4.3 Реализация на MPI

5 Результаты на системах Blue Gene/P и Polus

Для данной задачи выполнены подсчёты ускорения программы на системах Blue Gene/P и Polus.

Под ускорением программы, запущенной на p MPI-процессах, понимается величина:

$$S_p = \frac{T_2}{T_p}$$

где T_2– время работы на минимальном числе MPI-процессов, T_p– время работы программы на p MPI-процессах.

Число процессоров p	Число точек сетки $M \times N$	Время решения T	Ускорение S
128	500×1000		
256	500×1000		
512	500×1000		
128	1000×1000		
256	1000×1000		
512	1000×1000		

Таблица 2: Таблица с результатами расчетов MPI версии на ПВС IBM Blue Gene/P

Число процессоров p	Число точек сетки $M \times N$	Время решения T	Ускорение S
128	500×1000		
256	500×1000		
512	500×1000		
128	1000×1000		
256	1000×1000		
512	1000×1000		

Таблица 3: Таблица с результатами расчетов гибридной MPI+OpenMP версии на ПВС IBM Blue Gene/P

На Blue Gene/P

-

На Polus

-

Число процессоров p	Число точек сетки $M \times N$	Время решения T	Ускорение S
4	500×500		
8	500×500		
16	500×500		
32	500×500		
4	500×1000		
8	500×1000		
16	500×1000		
32	500×1000		

Таблица 4: Таблица с результатами расчетов MPI версии на ПВС IBM Polus

6 Заключение

7 Литература

Источники

[1] Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. ЧИСЛЕННЫЕ МЕТОДЫ. - М.: Наука, 1987.

8 Приложение 1. Код программы

```
#include <stdio.h>
#include <stdlib.h>
#include "mpi.h"
#include <time.h>
#include <math.h>

double A1 = 0;
double A2 = 4;
double B1 = 0;
double B2 = 3;


double u_2(double x, double y){
    return sqrt(4 + x * y);
}

double k_3(double x, double y){
    return 4 + x + y;
}

double q_2(double x, double y){
    double sum = x + y;
    if (sum < 0) {
        return 0;
    } else {
        return sum;
    }
}

double F(double x, double y){
    return ((pow(x, 3) - x*x*(y - 4) - x*(y*y + 8) + y*(y*y + 4*y - 8) + 4*
        q_2(x, y)*pow((4 + x*y), 2)) /
    4 * pow((4 + x*y), 1.5));
}


double psi_R(double x, double y){
    return (-y*y*(4 + x + y) + 4*pow(4 + x*y, 2)) / (4*pow(4 + x*y, 1.5));
}

double psi_L(double x, double y){
    return (y*y*(4 + x + y) + 4*pow(4 + x*y, 2)) / (4*pow(4 + x*y, 1.5));
}

double psi_T(double x, double y){
    return (-x*x*(4 + x + y)) / (4*pow(4 + x*y, 1.5));
}

double psi_B(double x, double y){
    return -psi_T(x, y);
}


double dot_product(double *U[], double *V[],
                   int M, int N,
                   double h1, double h2
                   ){
    if (M != N)
        printf("M is not equal to N. The answer will not be correct!");

```

```
60      double answer = 0.;
61      for (int i=0; i <= M + 1; i++){
62          for (int j=0; j <= N + 1; j++){
63              double rho, rho1, rho2;
64
65              if ((i == 0) || (i == M)){
66                  rho1 = 0.5;
67              } else {
68                  rho1 = 1;
69              }
70              if ((j == 0) || (j == N)){
71                  rho2 = 0.5;
72              } else {
73                  rho2 = 1;
74              }
75              rho = rho1 * rho2;
76              answer += rho * U[i][j] * V[i][j];
77          }
78      }
79      answer *= (h1 * h2);
80      return answer;
81  }
82
83  double norm(double *U[],
84             int M, int N,
85             double h1, double h2){
86      return sqrt(dot_product(U, U, M, N, h1, h2));
87  }
88
89  void B(double *B[],
90         int M, int N,
91         double h1, double h2,
92         double x_start, double y_start){
93      int i, j;
94  # pragma omp parallel for private(i, j)
95      for(i = 0; i <= M + 1; i++)
96          for (j = 0; j <= N + 1; j++)
97          {
98              B[i][j] = F(x_start + i * h1, y_start + j * h2);
99          }
100 }
101
102 void Aw(double *A[], double *w[],
103             int M, int N,
104             double h1, double h2,
105             double x_start, double y_start)
106 {
107     double wx, wy;
108     int i, j;
109     for(i = 0; i <= M + 1; i++){
110         for (j = 0; j <= N + 1; j++)
111         {
112             if(i == 0 || i == M + 1 || j == 0 || j == N + 1)
113             {
114                 A[i * (N + 2) + j] = w[i * (N + 2) + j];
115             }
116             else
117             {
118                 wx = (k_3(x_start + (i + 0.5) * h1, y_start + j * h2) * (w[i +
                        1][j] - w[i][j]) / h1
119                    - k_3(x_start + (i - 0.5) * h1, y_start + j * h2) * (w[
                           i][j] - w[i - 1][j]) / h1);
```

```
120                 wy = (k_3(x_start + i * h1, y_start + (j + 0.5) * h2) * (w[i][
                        j + 1] - w[i][j]) / h2
121                         - k_3(x_start + i * h1, y_start + (j - 0.5) * h2) * (w[
                            i][j] - w[i][j - 1]) / h2);
122                 A[i][j] = -wx/h1 - wy/h2 + q_2(x_start + i * h1, y_start + j *
                         h2) * w[i][j];
123             }
124         }
125     }
126 }
127
128 void vector_diff(double* res, double *w1, double *w0, int M, int N)
129 {
130     int i, j;
131     //# pragma omp parallel for private(i, j)
132     for(i = 0; i <= M + 1; i++)
133     {
134         for (j = 0; j <= N + 1; j++)
135         {
136             if(i == 0 || i == M + 1 || j == 0 || j == N + 1 || i == 1 || j ==
                    1 || i == M || j == N)
137                 res[i * (N + 2) + j] = 0;
138             else
139                 res[i * (N + 2) + j] = w1[i * (N + 2) + j] - w0[i * (N + 2) +
                        j];
140         }
141     }
142 }
143
144
145 double calculate_r(double *W[],
146                    int M, int N,
147                    double h1, double h2){
148     double r[M][N];
149     for (int i=0; i < M; i++){
150
151     }
152 }
153
154 int main(int argc, char *argv[]) {
155     if (argc != 3) {
156         printf("Program receive %d numbers. Should be 2: M, N\n", argc);
157         return -1;
158     }
159
160     int M = atoi(argv[argc - 2]);
161     int N = atoi(argv[argc - 1]);
162     if ((M <= 0) || (N <= 0)) {
163         printf("M and N should be integer and > 0!!!\n");
164         return -1;
165     }
166
167
168 }
```

Листинг 1: neyman_pde.c