Отчёт по заданию: Решение двумерного уравнения Пуассона методом фиктивных областей с MPI

Вариант: №9
Студент: Ма Синьюэ 617

Цель работы

Целью данной работы является численное решение двумерного уравнения Пуассона с использованием метода фиктивных областей и реализация алгоритма на основе технологии MPI.

Постановка задачи

Для успешного выполнения задания по MPI требуется:

1. Разработать и реализовать алгоритм **двумерного разбиения** прямоугольного контейнера $\Pi = [A_1, B_1] \times [A_2, B_2]$ на домены (подобласти) так, чтобы

   - отношение количества узлов по переменным x и y в каждом домене принадлежало диапазону $[1/2, 2]$;
   - количество узлов по переменным x и y любых двух доменов отличалось не более чем на единицу.

2. Используя средства библиотеки MPI и разработанный алгоритм разбиения, реализовать параллельный вариант метода сопряжённых градиентов для решения двумерного уравнения Пуассона. Проверить качество работы алгоритма, выполнив расчёты на сетке

   $$(M, N) = (40, 40)$$

   на одном, двух и четырёх процессах, провести сравнение с последовательным вариантом алгоритма по числу итераций и времени решения.

3. Выполнить серию вычислительных экспериментов на более крупных сетках

   $$(M, N) = (400, 600), \quad (M, N) = (800, 1200)$$

   при различном числе MPI-процессов и заполнить Таблицу 2 с результатами расчётов (число итераций, время решения и ускорение относительно последовательного запуска).

Метод и реализация

Область, сетка и правая часть

Рассматривается эллиптическая область

$$D = \{(x, y) : \ x^2 + 4y^2 < 1\},$$

вложенная в прямоугольный контейнер

$$A_1 = -1, \ B_1 = 1, \qquad A_2 = -0.6, \ B_2 = 0.6.$$

Сетка равномерная:

$$x_i = A_1 + i\,h_x, \quad i = 0, \ldots, M; \qquad y_j = A_2 + j\,h_y, \quad j = 0, \ldots, N,$$

где $h_x = \dfrac{B_1 - A_1}{M}, \ h_y = \dfrac{B_2 - A_2}{N}$. Правая часть строится как индикатор области:

$$B_{ij} = \begin{cases} F_{\mathrm{VAL}} = 1, & (x_i, y_j) \in D, \\ 0, & (x_i, y_j) \notin D. \end{cases}$$

Метод фиктивных областей и выбор ε

Коэффициент теплопроводности задаётся по правилу

$$k(x, y) = \begin{cases} 1, & (x, y) \in D, \\ \dfrac{1}{\varepsilon}, & (x, y) \notin D, \end{cases} \qquad \boxed{\varepsilon = \big(\max\{h_x, h_y\}\big)^2}.$$

(Именно такая формула ε используется в коде: `eps = max(h1,h2)*max(h1,h2);`).

Коэффициенты схемы a_{ij}, b_{ij}

Для ячейки (i, j) вычисляются доли пересечения её граней с областью D. Длины пересечения отрезков с эллипсом:

$$x = \mathrm{const} = x_0: \quad y \in \Big[-\sqrt{\tfrac{1-x_0^2}{4}}, \ \sqrt{\tfrac{1-x_0^2}{4}}\Big], \qquad y = \mathrm{const} = y_0: \quad x \in \Big[-\sqrt{1-4y_0^2}, \ \sqrt{1-4y_0^2}\Big].$$

Пусть $\ell_{ij}^{(v)}$ — часть вертикального ребра длиной h_y, лежащая в D, а $\ell_{ij}^{(h)}$ — часть горизонтального ребра длиной h_x, лежащая в D. Тогда (ровно как в функции `fictitious_regions_setup`):

$$a_{ij} = \begin{cases} 1, & \ell_{ij}^{(v)} = h_y, \\ \dfrac{1}{\varepsilon}, & \ell_{ij}^{(v)} = 0, \\ \dfrac{\ell_{ij}^{(v)}}{h_y} + \dfrac{1 - \frac{\ell_{ij}^{(v)}}{h_y}}{\varepsilon}, & \text{иначе,} \end{cases} \qquad b_{ij} = \begin{cases} 1, & \ell_{ij}^{(h)} = h_x, \\ \dfrac{1}{\varepsilon}, & \ell_{ij}^{(h)} = 0, \\ \dfrac{\ell_{ij}^{(h)}}{h_x} + \dfrac{1 - \frac{\ell_{ij}^{(h)}}{h_x}}{\varepsilon}, & \text{иначе.} \end{cases}$$

Действие оператора A

Разностный оператор для внутренних узлов ($i = 1..M-1,\ j = 1..N-1$) реализован как

$$(Aw)_{ij} = -\frac{1}{h_x}\left(a_{i+1,j}\frac{w_{i+1,j} - w_{i,j}}{h_x} - a_{i,j}\frac{w_{i,j} - w_{i-1,j}}{h_x}\right) - \frac{1}{h_y}\left(b_{i,j+1}\frac{w_{i,j+1} - w_{i,j}}{h_y} - b_{i,j}\frac{w_{i,j} - w_{i,j-1}}{h_y}\right).$$

(См. функцию `apply_A`).

Диагональный предобуславливатель D^{-1}

Используется диагональное предобуславливание (см. `apply_Dinv`):

$$D_{ij} = \frac{a_{i+1,j} + a_{i,j}}{h_x^2} + \frac{b_{i,j+1} + b_{i,j}}{h_y^2}, \qquad z_{ij} = (D^{-1}r)_{ij} = \frac{r_{ij}}{D_{ij}}.$$

Метод сопряжённых градиентов (PCG)

Итерационный процесс в `solve`:

$$r^{(0)} = B, \quad z^{(0)} = D^{-1}r^{(0)}, \quad p^{(1)} = z^{(0)}, \quad \langle u, v\rangle = \sum_{i,j} u_{ij}v_{ij}\,h_x h_y;$$

$$\text{на шаге } k: \quad \alpha_k = \frac{\langle z^{(k-1)}, r^{(k-1)}\rangle}{\langle Ap^{(k)}, p^{(k)}\rangle}, \qquad w^{(k)} = w^{(k-1)} + \alpha_k p^{(k)},$$

$$r^{(k)} = r^{(k-1)} - \alpha_k Ap^{(k)}, \qquad z^{(k)} = D^{-1}r^{(k)}, \qquad \beta_{k+1} = \frac{\langle z^{(k)}, r^{(k)}\rangle}{\langle z^{(k-1)}, r^{(k-1)}\rangle},$$

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

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

где $\|\cdot\|_E$ — евклидова норма, а δ задаётся пользователем (в тестах $\delta = 10^{-6}$).

Параллельная реализация MPI

Для распараллеливания разностной схемы используется библиотека MPI. Разбиение прямоугольника по условию задания (пункт 4) осуществляется по переменной x на несколько подобластей, каждая из которых обрабатывается отдельным MPI-процессом.

Двумерное разбиение области

Пусть всего P процессов и $(M-1) \times (N-1)$ внутренних узлов по координатам x и y. В программе сначала выбирается двумерная решётка процессов $P_x \times P_y$ (функция `choose_process_grid(P, Px, Py)`), где $P_x P_y = P$ и решётка по возможности близка к квадратной.

Затем функция `decompose_2d(M, N, Px, Py, rank, i_start, i_end, j_start, j_end)` делит множество внутренних индексов

$$i = 1, \ldots, M-1, \qquad j = 1, \ldots, N-1$$

между процессами так, что каждому процессу с координатами (p_x, p_y) в решётке соответствует прямоугольный блок

$$i_{\min}^{(p)} \le i \le i_{\max}^{(p)}, \qquad j_{\min}^{(p)} \le j \le j_{\max}^{(p)},$$

где

$$n_x^{(p)} = i_{\max}^{(p)} - i_{\min}^{(p)} + 1, \qquad n_y^{(p)} = j_{\max}^{(p)} - j_{\min}^{(p)} + 1$$

— число внутренних узлов по x и y на данном процессе. Размеры блоков по каждому направлению отличаются не более чем на единицу:

$$|n_x^{(p)} - n_x^{(q)}| \le 1, \qquad |n_y^{(p)} - n_y^{(q)}| \le 1$$

для любых процессов p, q. Тем самым выполняется требование задания о двумерном разбиении прямоугольника: каждый MPI-процесс отвечает за собственный подпрямоугольник сетки.

Для удобства используется локальная нумерация

$$l_i = i - i_{\min}^{(p)} + 1, \quad l_j = j - j_{\min}^{(p)} + 1,$$

где $l_i = 1, \ldots, n_x^{(p)}$, $l_j = 1, \ldots, n_y^{(p)}$. Дополнительно по каждому направлению вводятся *призрачные* (halo) слои с индексами $l_i = 0$, $l_i = n_x^{(p)}+1$ и $l_j = 0$, $l_j = n_y^{(p)}+1$, в которых хранятся значения граничных узлов, полученные от соседних процессов при обмене данными.

Локальные операции и обмен граничной информацией

Коэффициенты a_{ij}, b_{ij} и правая часть B_{ij} вычисляются локально в процедуре `fictitious_regions_se` только для принадлежащих процессу индексов $i = i_{\min}^{(p)}, \ldots, i_{\max}^{(p)}$ (плюс один слой по x для корректного вычисления потоков через границы). Формулы полностью совпадают с последовательным случаем.

Действие оператора A над вектором w реализовано в функции `apply_A_local`: во внутренних узлах ($l_i = 1..n_p,\ j = 1..N-1$) используется та же разностная формула, что и в последовательной версии, но все вычисления ведутся только по локальным индексам. Пе-

ред каждым вызовом `apply_A_local` выполняется обмен граничными слоями между соседними процессами (`exchange_halos()`), где используются пары вызовов `MPI_Sendrecv`. На крайних процессах граничные слои, соответствующие границе контейнера, задаются равными нулю, что соответствует граничным условиям Дирихле.

Диагональный предобуславливатель D^{-1} также строится локально (`apply_Dinv_local`): для каждого внутреннего узла процесс вычисляет

$$D_{ij} = \frac{a_{i+1,j} + a_{i,j}}{h_x^2} + \frac{b_{i,j+1} + b_{i,j}}{h_y^2}, \qquad z_{ij} = \frac{r_{ij}}{D_{ij}},$$

используя только локальные данные и призрачные слои.

Параллельный метод сопряжённых градиентов

Векторные операции метода сопряжённых градиентов выполняются локально, а глобальные скалярные произведения вычисляются в два шага:

1. каждый процесс считает свою локальную сумму $\sum_{l_i,j} u_{ij} v_{ij}\, h_x h_y$ (функция `dot_local`);
2. все локальные суммы суммируются при помощи `MPI_Allreduce` с операцией `MPI_SUM`, формируя глобальное скалярное произведение.

Аналогично собирается глобальная норма $\|w^{(k+1)} - w^{(k)}\|_E$, которая используется в критерии остановки:

$$\|w^{(k+1)} - w^{(k)}\|_E < \delta, \qquad \delta = 10^{-6}.$$

После каждого шага все процессы получают одинаковое значение нормы, поэтому решение синхронно прекращает итерации.

Вся логика метода PCG вынесена в функцию `gradient_solver_mpi`, в которой используются описанные выше локальные операции и глобальные коллективные вызовы MPI.

Результаты расчётов с MPI

Размер сетки	**Число процессов**	**Число итераций**	**Время с MPI**	**Время с Openmp**
40×40	1	60	0.00328	0.005
40×40	2	60	0.00186	0.004
40×40	4	60	0.00189	0.004

Размер сетки	**Число процессов**	**Число итераций**	**Время решения**	**Ускорение**
400×600	2	546	2.646	1.86
400×600	4	546	1.432	3.44
400×600	8	546	0.905	5.45
400×600	16	546	0.438	11.25
800×1200	4	989	9.689	4.02
800×1200	8	989	5.022	7.75
800×1200	16	989	2.906	13.40
800×1200	32	989	2.382	16.35