ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ РАСЧЕТА ОПТИМАЛЬНЫХ СЕТОК*

Н. А. Артемова, А. Ф. Хайруллин, О. Б. Хайруллина
Институт математики и механики УрО РАН
Екатеринбург, Россия
e-mail: kob@imm.uran.ru

The generation of high-quality computational grids is one of the most important stages of numerical solution of mathematical physics problems. These high-quality grids are sufficiently smooth, close to homogenous and orthogonal grids, and they have large number of nodes. The time, which is necessary for constructing a grid of the required quality, can be a significant part of the full calculation time. In order to reduce the time of grid generation the parallel algorithm is suggested for constructing two-dimensional block-structured curvilinear optimal grids with large number of nodes in the domains of simple and complicated topologies with compact storage of grid nodes.

1. Оптимальные сетки

Пусть требуется построить криволинейную сетку из класса координатных сеток в заданной области D с границей Γ. В этом случае в криволинейной системе координат (p, q) участки границы объявляются координатными линиями, область представляется в виде прямоугольника или совокупности прямоугольников (блоков), в которых вводится регулярная равномерная ортогональная сетка. Сетку в области D будем считать оптимальной, если функции отображения сетки $x = x(p, q)$, $y = y(p, q)$ из плоскости (p, q) в плоскость (x, y) при заданной расстановке узлов на границе Γ минимизируют функционал [1, 2]

$$\Phi = \iint_D \left[\left(\ln \sqrt{x_p^2 + y_p^2} \right)_p^2 + \left(\ln \sqrt{x_q^2 + y_q^2} \right)_q^2 + A \frac{\left(x_p^2 + y_p^2\right)\left(x_q^2 + y_q^2\right)}{\left(x_q y_p - x_p y_q\right)^2} \right] dp\, dq,$$

формализующий критерии оптимальности — близость сеток к равномерным по расстояниям между узлами и близость к ортогональным в точках пересечения координатных линий. Управлять качеством сеток можно, меняя вес $A>0$, который может быть постоянным во всей области либо иметь свое значение в каждом блоке.

В работах [3, 4] описан итерационный последовательный метод конструирования блочно-регулярных криволинейных оптимальных сеток в одно- и многосвязных областях. В сетках могут присутствовать элементы базисных сеток типа O, C, H [5], когда отображения их из плоскости (x, y) в плоскость (p, q) и обратно имеют неоднозначность: в плоскости (x, y) —

*Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований, гранты №99–01–00326, №00–15–96042.

вдоль линий двойственности (сетки типа H), в плоскости криволинейных координат — вдоль разрезов (сетки типа O и C). Образ заданной области в плоскости (p, q) может быть и неоднолистным.

2. Последовательный алгоритм расчета сеток

Блочная сетка в области рассчитывалась на каждой итерации поочередно во всех блоках в заданной последовательности. Общая сетка формировалась в результате объединения регулярных сеток в блоках, покрывающих эту область, и обладала гладкостью сеточных линий на границах стыковки блоков, которые так же, как и разрезы, и линии двойственности, объявлялись сеточными линиями в плоскости (x, y).

Структура матрицы координат точек сетки в результате специальной нумерации [3] определялась геометрией образа области в плоскости (p, q), который вписывался в прямоугольник размера $M \times N$. Матрица заполнялась флажковым методом: если точка не принадлежала области, то в соответствующий элемент матрицы засылался "флажок". Для запоминания неоднозначности координат линий двойственности в матрицу вводились два столбца, если линия проходила по вертикали, или две строки, если по горизонтали. Каждой точке разреза в плоскости (x, y) соответствуют не менее двух элементов матрицы, так как разрез в плоскости (p, q) имеет два образа, а концевые его точки — от двух до четырех в случае выхода из точки более одного разреза.

На рис. 1, *б*, *в* приведены два образа одной сетки (рис. 1, *а*) при разном раскрое ее на блоки. В первом случае (рис. 1, *б*) место для хранения координат точек сетки используется неэкономно ($56 \times 121 \times 2 = 12552$ элементов). Во втором случае (рис. 1, *в*, неоднолистное отображение) более плотно заполненную матрицу ($61 \times 61 \times 2 = 7442$ элементов) получили, используя другие раскрой и нумерацию точек в результате ручной работы.

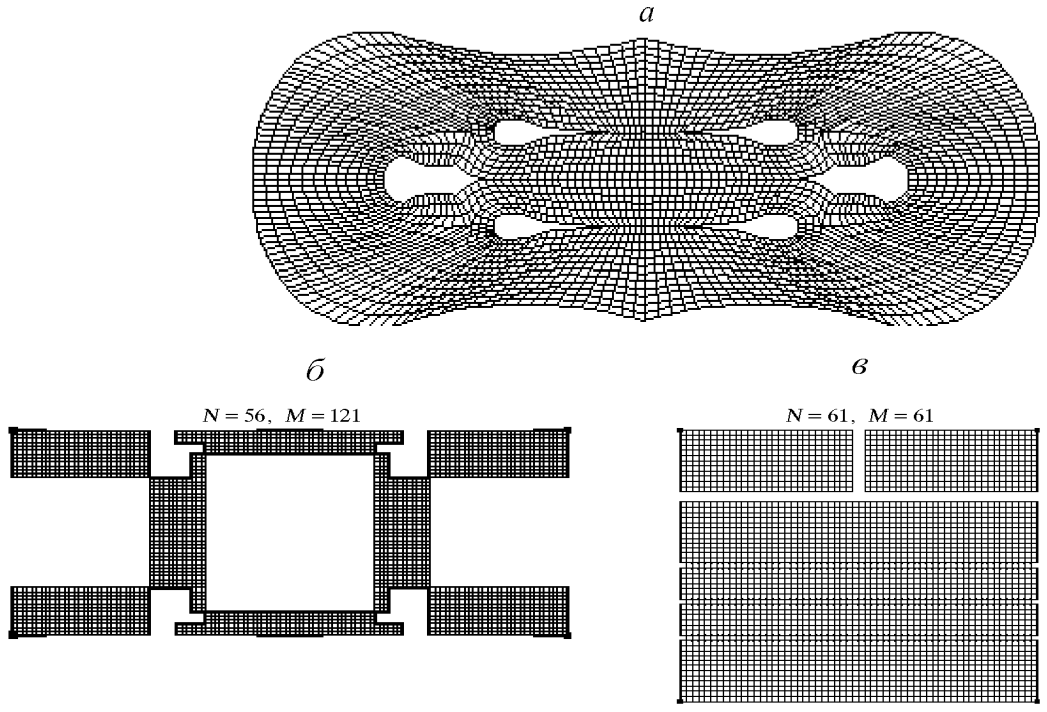

Рис. 1. Оптимальная сетка и два ее образа в плоскости криволинейных координат.

Начальное приближение сетки строится методами, описанными в [3, 4], либо берется из какого-то этапа расчета решаемой задачи. Иногда оказывается, что полученная сетка не может достаточно быстро оптимизироваться (в некоторых подобластях слишком густая сетка либо очень вытянутые ячейки в одном из направлений и т. д.). В этом случае проводится промежуточная оптимизация с помощью дискретного функционала, формализующего критерий близости сеток к равномерным по площадям соседних ячеек:

$$\Psi = \sum_{i,j}(S_{1ij} - S_{2ij})^2\left(\frac{1}{S_{1ij}^2} + \frac{1}{S_{2ij}^2}\right) + (S_{2ij} - S_{3ij})^2\left(\frac{1}{S_{2ij}^2} + \frac{1}{S_{3ij}^2}\right) +$$

$$+(S_{3ij} - S_{4ij})^2\left(\frac{1}{S_{3ij}^2} + \frac{1}{S_{4ij}^2}\right) + (S_{4ij} - S_{1ij})^2\left(\frac{1}{S_{4ij}^2} + \frac{1}{S_{1ij}^2}\right).$$

Площади ячеек шаблона (рис. 2, *а*) для всех точек (i, j) сетки рассчитываются по формуле

$$S_{kij} = \frac{1}{2}\left[(x_4 - x_1)(y_2 - y_1) - (x_2 - x_1)(y_4 - y_1) + (x_4 - x_3)(y_2 - y_3) - (x_2 - x_3)(y_4 - y_3)\right],$$

где x_n, y_m — координаты вершин ячейки, $n, m = 1, 2, 3, 4$ (рис. 2, *б*).

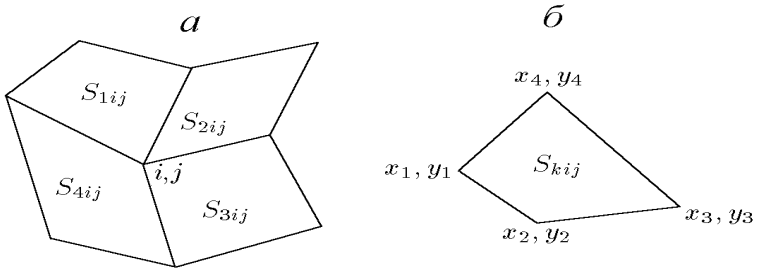

Рис. 2. Шаблон для расчета координат точки (i, j) (*а*) и ячейка сетки (*б*).

Функционал предложен А. Ф. Сидоровым.

Построение сеток, обладающих гладкостью координатных линий. Каждый блок рассматривается как заданная односвязная область с заданной расстановкой узлов на границе [3, 4]. Если в блоках строить сетки независимо от координат точек сеток соседних блоков, то на их общих границах будет нарушена гладкость сеточных линий. Такие сетки являются плохими при решении задач, когда искомые величины имеют большие градиенты в окрестности линий стыковок блоков [6]. Поэтому для расчета блочных сеток используется метод перекрытия блоков. Каждый блок, имеющий на своей границе участок, общий с участком границы другого блока (внутренняя граница), расширяется за эту границу на одну координатную полосу. В качестве границы расширенного блока берутся соответственно вертикаль или горизонталь из соседнего блока. После этого на каждой итерации сетка рассчитывается поочередно во всех блоках. За счет перекрытия блоков на внутренних границах области точки сетки будут рассчитываться в соответствии с заданными критериями оптимальности. Аналогично следует поступать и при расчетах сеток в многосвязных областях со сложной топологией. Здесь необходимо обеспечить движение узлов на разрезах и линиях двойственности. В этих случаях анализ геометрических возможностей стыковок блоков существенно сложней.

Последовательный алгоритм расчета сеток в многосвязных областях сложных топологий с автоматической организацией движения узлов сетки на внутренних границах, разрезах и линиях двойственности реализован в комплексе программ МОПС-2а для персонального компьютера. Алгоритм имеет ряд недостатков, главные из которых — неэкономное использование ресурсов памяти для хранения координат точек сетки, ограничение максимального размера рассчитываемой сетки оперативной памятью компьютера.

В настоящей работе предлагается параллельный алгоритм расчета оптимальных блочно-регулярных сеток с плотной упаковкой матрицы координат узлов сетки, не зависящей от раскроя области на блоки. Этот алгоритм, частично описанный в [7, 8], основан на алгоритме расчета сетки в блоке из последовательного алгоритма и в значительной мере использует распределенную память многопроцессорной вычислительной системы (МВС) [9].

3. Перекрытие блоков в параллельном алгоритме

Каждый блок k в плоскости криволинейных координат (p, q) можно описать координатами i_{1k}, j_{1k}, i_{2k}, j_{2k} его вершин и признаками $c_{1k}, c_{2k}, c_{3k}, c_{4k}$ сторон. Связи между блоками организуются по координатам их вершин. Признаки c_{lk} характеризуют тип границы блока и могут принимать значения 0 (заданная граница области), 1 (внутренняя граница), 31 (внутренняя граница на линии двойственности) и $c_{lk} > 40$ (разрезы) [3, 4].

Для расчета блочных сеток используем метод перекрытия. Расширим каждый блок на одну координатную полосу со всех сторон независимо от типа границы. Тогда любой блок будет состоять из двух взаимосвязанных частей: внутренней (“тело”) и внешней (“бордюр”) (рис. 3).

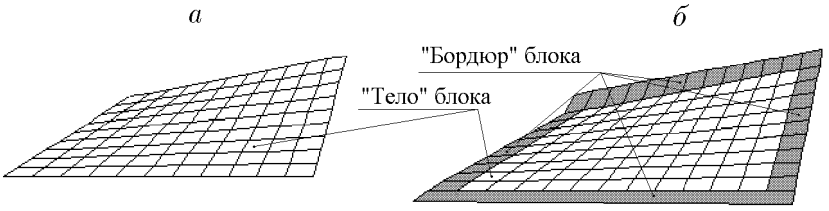

Рис. 3. Структура блока: блок без перекрытий (*а*); расширенный блок (*б*).

Если $c_{ik} = 1$ или $c_{ik} = 31$ на какой-либо стороне блока, то “бордюр” блока накладывается на соседние блоки (случай внутренней границы) и координаты точек внешней линии “бордюра” приравниваются значениям координат точек этой линии в смежных блоках.

Для расчета сетки в блоке k с разрезом $c_{ik} = m$ $(m > 40)$ ищется блок l, у которого одна из сторон является разрезом с тем же номером $c_{jl} = m$. Координаты точек внутренней линии “бордюра” блока k, соответствующей этому разрезу, а также внешней его линии, приравниваются значениям координат точек граничной и приграничной линий блока l. Если “бордюр” прилегает к границе заданной области, то координаты точек его внешней линии маркируются “флажком” — числом 10^{18}. Гладкие сетки в блоках, у которых “бордюр” полностью не отмаркирован “флажком”, нельзя в произвольном порядке одновременно рассчитывать на разных процессорах.

Рассмотрим, например, блок A_1 (рис. 4), у которого стороны bc и cd имеют признаки $c_{2k} = c_{3k} = 1$. Сетка в соседних блоках B_1, C_1 при использовании нескольких процессоров должна рассчитываться в разное время, чтобы после ее вычисления в одном из блоков

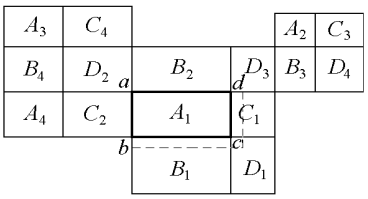

Рис. 4. Принципиальная схема распараллеливания.

можно было обменяться информацией о границе с блоком A_1. Координаты точки, лежащей на пересечении bc и cd, должны быть вычислены также в соответствии с заданными критериями оптимальности. Для этого расчет сетки в блоке D_1 следует отделить по времени от расчета сетки в блоках A_1, B_1, C_1. Таким же образом должны быть рассмотрены другие стороны этого и остальных блоков.

После такого анализа можно выделить блоки A_i ($i = 1, ..., I_a$), B_i ($i = 1, ..., I_b$), C_i ($i = 1, ..., I_c$), D_i ($i = 1, ..., I_d$), которые объединяются в группы. В одну группу не попадают блоки, имеющие общие границы либо общие угловые точки. Сортировка блоков по группам позволяет рассчитывать сетки одновременно на разных процессорах в блоках из одной группы и поочередно в блоках разных групп. Из схемы, приведенной на рис. 4, являющейся образом в плоскости (p, q) некоторой области в плоскости (x, y), видно, что максимальное количество групп — четыре.

Перед сортировкой блоков по группам программным образом проверяется у каждого блока число боковых соседних блоков. Может оказаться, что с какой-то стороны блока расположены сразу несколько соседних блоков. В этом случае для универсальности алгоритма организуется дробление основного блока на два или более мелких блоков.

4. Упаковка координат точек сетки

Для хранения координат точек сетки предлагается блочная структура. Координаты x_{jk}, y_{ik} ($k = 1, 2, ...$) точек сеток расширенных блоков формируются в одномерные массивы с перебором точек по вертикалям снизу вверх и по горизонталям справа налево, которые записываются друг за другом. В результате имеем массив координат точек сетки с плотной упаковкой, не зависящей от раскроя области на блоки. Тем самым при расчете сетки исчезает очень трудный этап ручной работы, когда при раскрое области на блоки необходимо заботиться о структуре матрицы координат сетки. Кроме этого, хранение сетки в блоках вместе с “бордюром” позволяет уменьшить число обменов информацией о перекрытиях, а также время поиска этой информации.

Блочная структура хранения матрицы позволила рассчитывать блоки с несколькими разрезами, в то время как при расчете сетки с помощью последовательного алгоритма предполагалось, что рассчитываемый блок не может иметь более одного разреза [4]. Если бы это было не так, а в общем случае блок может иметь до четырех разрезов, то эта проблема решалась размножением блока на несколько (по числу разрезов) блоков, каждому из которых приписывался определенный разрез. На рис. 5 приведен случай, когда разрезами являются две соседние стороны блока.

При расчете сетки с помощью параллельного алгоритма возникла проблема обработки угловых внешних точек “бордюра”, являющихся пересечением его сторон, примыкающих к разрезам (точка A на рис. 5). В случае, когда они не принадлежат расчетной области,

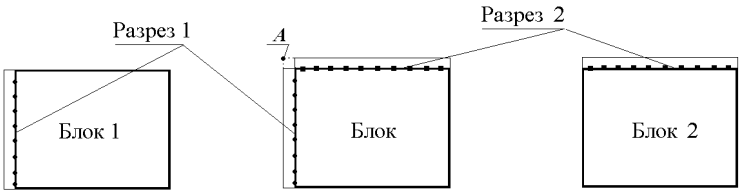

Рис. 5. Размножение блока с двумя разрезами на два с одним разрезом у каждого.

точка метится особым образом. Если при расчете встречается точка с такой меткой, то координаты центральной точки шаблона не пересчитываются [4].

Зная номер блока k, номер элемента в общем массиве, являющегося началом одномерного массива для данного блока, а также значения i_{1k}, j_{1k} для него, можно восстановить образ построенной сетки.

С другой стороны, для решения задач на больших сетках образ сетки можно не восстанавливать, а решать задачи поблочно на нескольких процессорах, используя распределенную память МВС и информацию о соседних блоках. Так как каждый блок имеет не более восьми соседних блоков (по одному у каждой стороны и у каждого угла рассчитываемого блока), то взаимосвязи решаемых на разных процессорах задач легко организуются.

5. Загрузка процессоров

Предлагаемый параллельный алгоритм итерационный. При расчете оптимальной криволинейной сетки в заданной области предполагается, что известно какое-то начальное распределение узлов сетки в ней и набор параметров, характеризующих блоки этой области (координаты i_{1k}, j_{1k}, i_{2k}, j_{2k} вершин и признаки $c_{1k}, c_{2k}, c_{3k}, c_{4k}$) сторон. Автоматически определяется тип границы каждого блока, заполняются внешние линии “бордюра”, все блоки сортируются на группы. На экран монитора выдается образ исходной области в плоскости криволинейных координат, раскроенной на блоки (рис. 6), где каждый вид штриховки характеризует свою группу.

Время расчета оптимальной сетки на многопроцессорной машине состоит в основном из времени расчета сетки в блоках на процессорах и времени обмена результатами вычислений между этими процессорами. Главными критериями минимизации времени расчета являются равномерная загрузка используемых процессоров и минимум объема передаваемой информации.

Будем считать, что время расчета координат каждого узла блока примерно одинаковое. Тогда время расчета сетки в блоке k определится количеством L_k его узлов. Назовем его весом блока. Введем параметр B_k, который будет равен L_k, если у блока k нет разрезов. Блоки с одноименными разрезами могут находиться как в разных группах, так и в одной. Если такие блоки попадают в одну группу, то с целью сокращения обменов информацией между процессорами они распределяются на один процессор. Пусть блоки k и l имеют один и тот же разрез. Тогда параметр B_k приравнивается сумме весов этих блоков $(L_k + L_l)$, а параметр B_l обнуляется. В общем случае в рассматриваемой группе у блока k может быть несколько “соседей” по разрезам (например, у блока k есть общие разрезы с блоками l_1 и l_2, а у блока l_1 кроме блока k имеется еще общий разрез с блоком l_3, и все они оказались

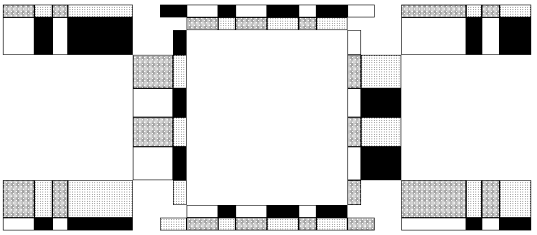

Рис. 6. Сортировка блоков на группы.

в одной группе). Тогда

$$B_k = L_k + \sum_j L_{l_j}, \qquad B_{l_j} = 0 \ \text{ для } \ j = 1, 2, \ ...$$

Блоки данной группы нумеруются в порядке убывания параметра B_k, и информация о них формируется в массиве M_1. Количество блоков с $B_k \neq 0$ определит максимальное число необходимых процессоров P_1 для расчета этой группы блоков. В зависимости от соотношения заданного P и требуемого P_1 числа процессоров возникают две ситуации.

Первая ситуация, когда $P \geq P_1$. Блоки с $B_k \neq 0$ распределяются по процессорам, а блоки с $B_l = 0$ рассылаются на те процессоры, где должны рассчитываться блоки с одноименными разрезами. Здесь наблюдается неравномерность работы процессоров, часть которых простаивает или работает незначительное время.

При второй ситуации, когда $P < P_1$, удается добиться более равномерной загрузки. Для каждой группы формируется матрица M_2 с числом столбцов, соответствующих заданному числу процессоров, и числом строк, равным количеству блоков в группе. Элементами столбца m_i матрицы M_2 являются номера блоков, рассчитываемых на i-м процессоре.

Определяется среднеарифметическое значение S параметров B_k всех блоков данной группы. Если для каких-либо блоков $B_k \geq S$, то их номера сразу переставляются из массива M_1 в массив M_2 и сумма величин B_k для каждого i-го процессора накапливается в S_{1i}. Оставшиеся блоки распределяются с учетом убывания веса B_k. При выполнении условия $S_{1i} + B_k < S$ номер блока k с максимальным значением B_k переставляется из массива M_1 в M_2, соответственно изменяется S_{1i}.

Если после такой процедуры распределения блоков на P процессоров в массиве M_1 остались номера блоков, для которых $B_k \neq 0$, то описанная выше процедура повторяется, а значение S увеличивается на 1 % по сравнению со значением S на предыдущей итерации. С каждой итерацией увеличивается неравномерность загрузки процессоров. После распределения всех блоков с $B_k \neq 0$ по процессорам номера блоков с $B_k = 0$ добавляются в те столбцы массива M_2, где находятся номера блоков с одноименными разрезами.

С целью сокращения обменов информацией блоки, попавшие на один и тот же процессор и имеющие общую сторону с признаком $c_{ik} = 1$, объединяются. Полный анализ блоков и распределение их по процессорам занимают несколько секунд процессорного времени.

После расчета начального приближения и сортировки блоков по процессорам на каждый процессор посылаются координаты точек сетки тех блоков, которые приписаны данному процессору, и служебная информация об этих блоках (размеры, очередность расчета, информация о пересылках перекрытий).

Для совмещения времени расчета сетки в блоках и обмена общей информацией работа каждого процессора построена по схеме, состоящей из четырех этапов:

— открыть каналы для приема информации с других процессоров;

— произвести необходимые вычисления на данном этапе;

— передать информацию, необходимую другим процессорам, и дождаться окончания приема информации о перекрытиях;

— приступить к обработке следующей группы блоков или расчету следующей итерации, если вся принимаемая информация на процессор уже поступила.

Информация о перекрытиях, которую необходимо направить другим блокам на этом же процессоре, передается сразу после расчета сетки в блоке-отправителе.

После прохождения четырех этапов для всех групп итерация полностью рассчитана. Последний процессор, как наименее загруженный, собирает со всех остальных значения функционала в блоках, суммирует, принимает решение перейти к расчету следующей итерации или закончить расчет сетки и сообщает о своем решении остальным процессорам. После окончания расчета процессоры записывают свою часть сетки в файл, в котором хранятся в упакованном виде координаты точек сетки всех блоков.

6. Пример расчета сетки

В методических целях при расчете сеток в разных областях замерялось время счета, задержки старта счета (работа МВС-100 по маршрутизации), чтения входных данных и записи результатов счета, рассылки перекрытий и ожидания их приема. Подсчитывались коэффициенты ускорения счета $K_{\text{у}}$ (отношение времени расчета на одном процессоре к времени счета на N процессорах) в зависимости от числа процессоров, коэффициенты эффективности загрузки процессоров ($K_{\text{э}} = K_{\text{у}}/P$).

Так, в области, представленной на рис. 1, *а*, рассчитывалась сетка, образ которой (рис. 6) вписывался в прямоугольник размера 1183×562 (664846 узлов, часть из которых фиктивные). Информация о координатах точек сетки плотно упакована по блокам. Для расчета потребовалось 14 итераций.

На рис. 7 приведены графики зависимости времени расчета сетки с учетом (T, полное) и без учета (T_1, чистое) времени считывания начального приближения, записи результата и задержки старта счета от числа процессоров P.

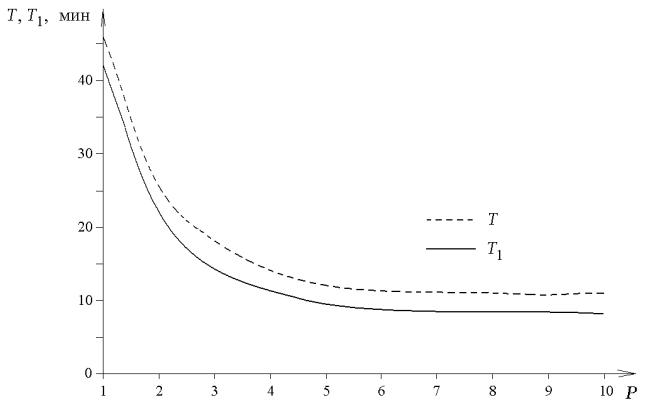

Рис. 7. Время расчета сетки на P процессорах.

Резкое снижение затрат времени происходит при использовании двух, трех процессоров. Начиная с шести процессоров время счета сетки при данном раскрое области на

блоки мало меняется. Это объясняется неравномерностью загрузки процессоров расчетными подобластями. На рис. 8 представлены диаграммы загрузки процессоров одной из групп (из-за симметричности области все четыре группы одинаковые). При $P = 4$ (рис. 8, *а*) процессоры практически равномерно загружены, при $P = 9$ (рис. 8, *б*) полное время счета определяет время счета на первом процессоре, где расположен блок с наибольшим весом B_k. Равномерность загрузки повышается, если раздробить этот блок на две или более части.

Основные накладные расходы на задержку старта, чтение данных и запись результатов счета в процентах от полного времени T расчета сетки приведены в табл. 1.

Из рис. 7 видно, что для разного числа процессоров накладные расходы примерно одинаковы, их доли от общих временных затрат с увеличением P возрастают, но несущественно. Так, на десяти процессорах по сравнению с двумя процессорами доля времени накладных расходов возросла примерно в два раза.

На рис. 9 представлены зависимости коэффициентов ускорения $K_{\text{у}}$ и эффективности счета $K_{\text{э}}$ от числа процессоров. На нижних графиках $K_{\text{у}}$, $K_{\text{э}}$ подсчитаны с учетом накладных расходов, на верхних $K_{1\text{у}}$, $K_{1\text{э}}$ — без их учета, т. е. коэффициенты соответствуют чистому времени T_1 счета. Наибольший коэффициент ускорения при расчете сетки $K_{1\text{у}} = 5.2$ получен на десяти процессорах, хотя реально $K_y = 4.2$. При этом коэффициент

Рис. 8. Диаграммы загрузки четырех (*а*) и девяти (*б*) процесоров.

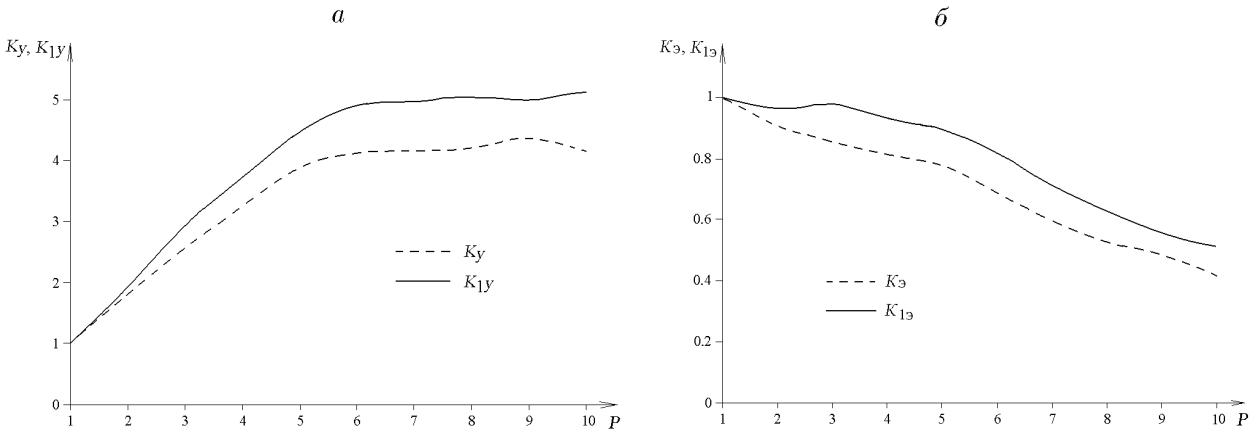

Рис. 9. Зависимость коэффициентов ускорения (*а*) и эффективности счета (*б*) от числа процессоров P.

эффективности загрузки $K_{1\text{э}} = 0.51\,(K = 0.42)$, т. е. процессоры при расчете данной сетки используются практически в половину возможной нагрузки (см. рис. 8, *б*).

Исследовался расход времени и на каждой итерации (табл. 2, проценты от среднего времени счета одной итерации).

Таблица 1

Причина	Количество процессоров					
	1	2	4	6	8	10
Задержка старта	0.26	0.29	0.70	1.21	1.64	2.47
Чтение данных	4.34	6.91	10.1	12.0	13.4	11.7
Запись сетки	3.97	7.08	11.00	14.2	11.9	14.6

Таблица 2

Итерация	Количество процессоров					
	1	2	4	6	8	10
Открытие буферов	0.02	0.07	0.24	0.31	0.36	0.38
Рассылка перекрытий	0.04	0.07	0.56	0.85	0.84	2.84
Ожидание конца приема	0.02	2.73	2.86	7.33	20.2	22.7
Переход к новой группе	0.02	0.05	0.15	0.15	0.18	0.14

Временные затраты на рассылку и прием перекрытий сильно зависят от равномерности загрузки процессоров, затраты на открытие буферов и переход к новой группе несущественны.

Для сравнения приведем среднее время расчета одной итерации этой сетки: на ПК (Pentium, 100 МГц) с использованием последовательного алгоритма — 19 мин 11 с; с использованием параллельного алгоритма на одном процессоре МВС — 6 мин 7 с, на четырех процессорах — 49 с, на восьми — 36 с.

7. Заключение

Модифицированный алгоритм МОПС-2м позволяет, используя распределенную память МВС, конструировать в многосвязных двумерных областях сложных топологий оптимальные блочно-регулярные сетки большого размера (при настоящей конфигурации МВС [10], имеющейся в Институте математики и механики УрО РАН, порядка 150 млн узлов) за приемлемое время. Сетки обладают гладкостью сеточных линий на границах стыковки блоков. Независимо от раскроя области на блоки массив координат точек сетки плотно упакован, что существенно сокращает необходимую для его хранения память (так, например, для раскроя, представленного на рис. 6, вместо 21.94 Мб в приведенном расчете сетки при плотной упаковке достаточно 9.62 Мб памяти, что в 2.24 раза меньше).

Использование “бордюра” блока делает организацию перекрытий более универсальной, позволяет отказаться от использования дополнительных перекрытий, сокращает время счета.

Список литературы

[1] Serezhnikova T. I., Sidorov A. F., Ushakova O. V. On one method of construction of optimal curvilinear grids and its applications // Soviet J. Numer. Anal. Math. Modelling. 1989. Vol. 4, No. 2. P. 137–155.

[2] Khairullina O. B., Sidorov A. F., Ushakova O. V. Variational Methods of Construction of Optimal Grids // Handbook of Grid Generation / J. F. Thompson, B. K. Soni, N. P. Weatherill (Eds). Boka Raton etc.: CRC Press, 1998. Vol. 36. P. 1–25.

[3] Khairullina O. B. Method of constructing block regular optimal grids in two-dimensional multiply connected domains of complicated geometries // Russ. J. Numer. Anal. and Math. Modelling. 1996. Vol. 11, No. 4. P. 343–358.

[4] Хайруллина О. Б. Метод построения блочно-регулярных оптимальных сеток в двумерных многосвязных областях сложных конфигураций (МОПС-2а): науч. докл. Екатеринбург: УрО РАН. 1998. 56 с.

[5] Thompson J. F., Warsi Z. U. A., Mastin C. W. Numerical Grid Generation: Foundations and Applications. N. Y. etc: North – Holland, 1985.

[6] Khairullina O. B. Modelling subsonic vortex gas flows in channels of complex geometries // Russ. J. Numer. Anal. and Math. Modelling. 1998. Vol. 13, No. 3. P. 191–218.

[7] Артемова Н. А., Хайруллин А. Ф., Хайруллина О. Б. Построение оптимальных сеток в многосвязных областях сложных топологий на многопроцессорных машинах // Алгоритмы и программные средства параллельных вычислений: науч. тр. Екатеринбург: УрО РАН, 1998. №2. С. 22–38.

[8] Хайруллина О. Б., Хайруллин А. Ф., Артемова Н. А. Расчет оптимальных сеток большой размерности в многосвязных областях с использованием распределенной памяти МВС-100 // Алгоритмы и программные средства параллельных вычислений: науч. тр. Екатеринбург: УрО РАН, 1999. №3. С. 239–251.

[9] Zabrodin A. V., Levin V. K., Korneev V. V. The Massively Parallel Computer System MVC-100 // Parallel Computing Technologies: Third Intern. Conf., PaCT-95. St. Petersburg, 1995. P. 341–355.

[10] Гольдштейн М. Л., Закурдаев Н. В. Опыт создания ВЦ на базе ЭВМ МВС-100, МВС-1000 // Алгоритмы и программные средства параллельных вычислений: науч. тр. Екатеринбург: УрО РАН, 1999. №3. С. 48–56.

Поступила в редакцию 14 августа 2000 г.