И. В. Бурдонов, Ф. А. Мурзин

О РАСПАРАЛЛЕЛИВАНИИ МЕТОДА МЕДУЗА[1]

1. ВВЕДЕНИЕ

Для решения ряда задач математической физики используется методика МЕДУЗА [1]. Ее сущность состоит в том, что область исследования представляется в виде точек и окружающих их областей. На первом этапе область заполняется точками, а потом строится сетка таким образом, чтобы в каждой ее ячейке находилась ровно одна точка. Для вычисления величин в узле (центре ячейки) используются данные из вершин (многоугольника, образующего данную ячейку), вычисленные на предыдущем шаге, и наоборот.

В работе рассматривались различные вопросы, связанные с распараллеливанием методики МЕДУЗА на многопроцессорной вычислительной системе, использующей коммутатор. Были найдены способы распределения данных по процессорам и способы связи между ними.

Во второй части работы приводится краткое описание методики МЕДУЗА в применении к двумерным газодинамическим задачам.

Далее исследуется отображение метода на параллельную вычислительную машину. В конце работы приведены оценки времени выполнения алгоритма в параллельном и последовательном случаях и коэффициента ускорения.

2. МЕТОДИКА МЕДУЗА РАСЧЕТА ДВУМЕРНЫХ ГАЗОДИНАМИЧЕСКИХ ЗАДАЧ

2.1. Постановка дифференциальной задачи

Основой постановок задач, решаемых предложенной методикой в данной работе, является лагранжева система:

$$\begin{array}{lll} \frac{du}{dt} = -\sigma\frac{\partial p}{\partial x}, & \frac{dv}{dt} = -\sigma\frac{\partial p}{\partial y}, & \frac{dx}{dt} = u, \\ \frac{dy}{dt} = v, & \frac{dE}{dt} = -p\frac{d\sigma}{dt}, & \frac{d\sigma}{dt} = \sigma div(u,v). \end{array} \qquad (1.1)$$

[1]Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

Здесь $\sigma = \frac{1}{\rho}$ — удельный объем. Система (1.1) замыкается уравнением состояния в форме

$$\mathrm{E} = \mathrm{E}(\mathrm{p}, \rho). \tag{1.2}$$

В начальный момент времени $t = t_0$ состояние среды в некоторой области G переменных (x, y) предполагается известным:

$$u = u_0(\mathrm{x}, \mathrm{y}), v = v_0(\mathrm{x}, \mathrm{y}), \rho = \rho_0(\mathrm{x}, \mathrm{y}), \mathrm{p} = \mathrm{p}_0(\mathrm{x}, \mathrm{y}). \tag{1.3}$$

На границе Γ области G задано какое-нибудь правильное граничное условие, корректно замыкающее задачу, например:

$$\mathrm{p}|_\Gamma = p(t, M), M \in \Gamma. \tag{1.4}$$

Требуется рассчитать решение эволюционной системы (1.1), (1.2) с условиями (1.3), (1.4).

В действительности рассчитываемая система от указанной отличается вязкой добавкой к давлению

$$q = -c\frac{d\sigma}{dt}|\frac{d\sigma}{dt}|$$

всюду, кроме (1.2). В дальнейшем об этом упоминаться не будет, и все выкладки будут проводиться с точностью до этого слагаемого.

2.2. Сетка

Конструирование разностной схемы связано с идеями Паста и Улама представления среды в виде глобул. Точнее, среда имеет представление в виде точек и окружающих их областей. Каждая точка несет следующую информацию:

X, Y — эйлеровы координаты;
u, v — компоненты скорости;
p — давление;
σ — удельный объем;
m — масса точки; (2.1)
q — вязкость;
N — номер уравнения состояния;
A_i — адреса “соседей” данной точки.

Указание “соседств” A_i обозначает, что в расчете используется сетка, представляющая собой граф с вершинами в лагранжевых точках и

ребрами, обозначающими “соседство”. В процессе расчета архитектура графа меняется в соответствии с метрическими отображениями о близости точек. В этом смысле методика может быть названа методикой на нерегулярной сетке.

Для указания необходимых свойств сетки введем отношение “соседства” $A \to$ B (A является “соседом” B). Элементарные свойства этого отношения состоят в следующем:

а) $A \to$ B $\Rightarrow B \to A$, т. е. всегда имеет смысл $A \Leftrightarrow$ B, что означает неориентированность графа, выражаемую предложением: если A является “соседом” B, то и B является “соседом” A.

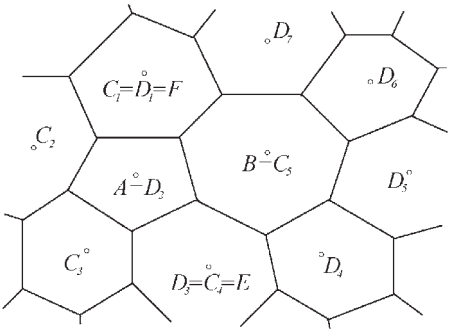

Рис. 1.

б) Если $C_j \leftarrow$ A, $D_k \leftarrow B, AB, j, k = 1..l > 3$, то найдутся две и только две точки E и F такие, что E $\in C_j, E \in D_k, F \in C_j, F \in D_k$, и F не является “соседом” E. Иными словами: если A и B — “соседи”, то в пересечении множества “соседей” A с множеством “соседей” B найдутся две точки E и F, не являющиеся “соседями” между собой (см. рис. 1).

Это свойство важно в дальнейшем для перестройки “соседств”, происходящей по стандартному алгоритму при числе “соседей” большем трех. Перестройка графа при числе “соседств” равном трем возможна, но при этом следует использовать другой алгоритм. Чтобы без необходимости не усложнять программу, лучше сконструировать сетку с числом “соседей” не меньшим четырех.

Каждой точке сеточного графа ставится в соответствие некоторый многоугольник, именуемый далее областью соответствия. Сама точка будет называться узлом, или центром, своей области соответствия, а вершины многоугольника — вершинами области соответствия.

При этом требуется выполнение следующих условий:

1) точки сеточного графа должны принадлежать к внутренним точкам своей области соответствия;

2) вся исследуемая область движения должна быть покрыта множеством областей соответствия;

3) вершины областей соответствия должны быть точками лагранжевой сетки, т. е. должны перемещаться вместе со средой;

4) области соответствия должны быть выпуклыми.

Очевидно, требованию 3) можно удовлетворить, полагая координаты вершин областей соответствия линейными комбинациями координат исходных точек, не зависящими от времени. В реализованном алгоритме эти веса выбраны по 1/3, но, вообще говоря, ими можно распорядиться по-иному для достижения определенных целей, например улучшения аппроксимации границ.

Резюме высказанных положений о сетке может быть сформулировано следующим образом: дискретизация движущегося континуума осуществляется точками, являющимися вершинами плоского неориентированного графа.

2.3. Дискретизация задачи

Расчет по методике МЕДУЗА начинается с расстановки точек A_i в области G. Хорошая расстановка точек помогает получать более аккуратную информацию о движении. Этот момент не алгоритмизирован и диктуется опытом и вкусами вычислителя. Из общих соображений можно высказать лишь следующее: точки, сближающиеся в расчете сильнее других, разумно расставлять реже, а расширяющиеся — гуще. В случае, когда информативность различных участков G должна быть различной, следует прибегнуть к соответствующей расстановке.

После расстановки точек производится установление “соседств”. В работе [1] отмечено, что начальные “соседства” могут быть получены, например, на основе разбиения G на области Дирихле, каждая из которых является континуальным односвязным подмножеством G, соотнесенным к A_i по принципу метрической близости. Иными словами, каждая точка $Q \in G$ и $Q \notin A_i$ относится к той точке A_i, к которой она расположена ближе, чем к остальным. Такое разбиение G позволяет однозначно установить “соседей” для каждой точки A_i, поскольку границами областей Дирихле являются линии, равноудаленные от соседних точек. Одним из наиболее сложных вопросов, связанных с ме-

тодом, является геометрическая задача построения данного разбиения (в геометрии — построение диаграммы Вороного множества точек).

В различных исследованиях по вычислительной геометрии приводятся некоторые способы решения этой задачи. В работе [2] приводится способ построения диаграммы Вороного методом "разделяй и властвуй". Графическая интерпретация данного метода состоит из нескольких шагов.

Шаг 1. Множество точек $\{A_i\}$ делится на два приблизительно равных подмножества $\{A_i^{'}\}$ и $\{A_i^{''}\}$. Для этого используется медиана по x-координате.

Шаг 2. Рекурсивно строятся диаграммы Вороного для $\{A_i^{'}\}$ и $\{A_i^{''}\}$.

Шаг 3.1. Строится разделяющая цепь — некоторая ломаная, разделяющая $\{A_i^{'}\}$ и $\{A_i^{''}\}$. Ее звенья — ребра областей соответствия, пересеченные медианой.

Шаг 3.2. Ребра $\{A_i^{'}\}$, которые оказались за границей, в области $\{A_i^{''}\}$ удаляются. Аналогично с $\{A_i^{''}\}$.

По указанному алгоритму строится диаграмма Вороного. В работе [2] приведены все формулы и подробное описание построения разделяющей цепи. В нашей программе не важна полученная картина, необходимо лишь определить "соседей" для каждого узла. Так как ребро является показателем "соседства", то из указанного выше алгоритма путем введения (обоюдного) в отношение "соседства" узлов при определении ребра, которое их разделяет, получаем алгоритм определения "соседей". Если к нему сделать небольшую добавку в виде сортировки, то в конце получим упорядоченный список "соседей". Аналогично с точками пересечения ребер. На многопроцессорной вычислительной системе данный алгоритм в параллельном режиме может реализоваться лишь частично (процессы разделения множеств). Построение разделяющих цепей возможно только при последовательном продвижении по координатам, поэтому приведенные здесь выдержки из алгоритма носят характер примера.

В настоящее время известны алгоритмы построения диаграммы Вороного на основе сортировки точек через введение лексикографического порядка. Алгоритмы сортировки на данный момент изучены довольно хорошо, многие из них можно распараллелить (в данной работе это не рассматривается).

Будем говорить, что три соседних между собой узла "образуют" вершину, если она принадлежит границам областей этих узлов. Иными

словами, вершина находится в треугольнике (а в нашей реализации — в точке пересечения серединных перпендикуляров к его сторонам), образованном этими узлами при триангуляции. Каждую из вершин необходимо в процессе работы алгоритма занумеровать и составить для каждой список номеров трех "образующих" узлов. При этом номеру вершины однозначно соответствует тройка номеров узлов. Будем считать, что задана функция, которая по трем номерам узлов выдает номер вершины, которую они образуют: $m(n_1, n_2, n_3)$. Эта функция понадобится в дальнейшем для связи.

Таким образом, строится начальный граф, удовлетворяющий требованиям пп. 1), 2), 4) для областей соответствия, поскольку области Дирихле обладают свойствами разбиения G на выпуклые полигоны. Использованию областей Дирихле в качестве областей соответствия в задачах гидродинамики препятствует п. 3) о лагранжевом описании вершин областей соответствия. Поэтому после установления "соседств" производится разбиение на области соответствия с последующим вычислением масс.

Если среди областей соответствия встречаются треугольники, то точки либо слегка смещаются для получения четырех "соседей", либо им приписывается четвертый "сосед" из числа "соседей соседей". Дискретизация гидродинамических величин по начальным данным производится далее очевидным образом. Таким образом, начальная информация (2.1) установлена для каждой внутренней точки. Специфику информации о граничных точках мы рассмотрим позже.

2.4. Расчет внутренних точек

Расчет заключается в последовательном переходе с одного временного слоя на последующий в соответствии с разностной аппроксимацией системы (1.1—1.2).

Одна из разновидностей аппроксимации заключается в разностном представлении объемных законов сохранения. Термин "объемные" подчеркивает то, что они относятся к телам вращения вокруг оси x, полученным из многоугольников соответствия. Приведем расчетные формулы для точки с номером n. Определим s_n как число "соседей" для узла с номером n.

Пусть X_n, Y_n обозначают координаты рассчитываемой точки, а X_i, Y_i, $1 \le i \le s_n$ — координаты ее "соседей". Обозначим координаты вершин многоугольников соответствия через x_j, y_j, $1 \le j \le s_n$.

Необходимо установить соответствие между номерами вершин для n-го узла и номерами соседей (список которых уже упорядочен). Предположим, что такое соответствие установлено и, например, узлы $i, i+1$, $1 \le i \le s_n - 1$ "образуют" i-ю вершину, а узлы s_n и 1 — вершину с номером s_n (см. рис. 2).

Тогда координаты вершин рассчитываются по формуле

$$x_j = \frac{X_n + X_j + X_{j+1}}{3}, y_j = \frac{Y_n + Y_j + Y_{j+1}}{3}. \tag{4.1}$$

Далее рассчитываются площади S_j треугольников, являющихся частью области соответствия (см. рис. 2):

$$S_j = \frac{1}{2}[(x_j - X_n)(\mathrm{y}_{j+1} - Y_n) - (x_{j+1} - X_n)(\mathrm{y}_j - Y_n)]. \tag{4.2}$$

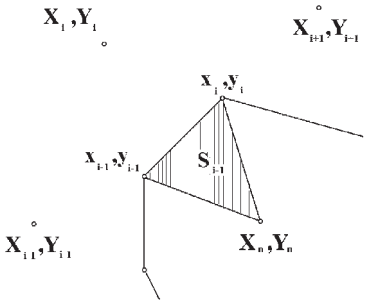

Рис. 2.

Затем рассчитываются объем тора многоугольного сечения, нормированный на единичный азимутальный угол и удельный объем:

$$V^n = \sum_{i=1}^{s_n} S_i \frac{Y_n + y_i + y_{i+1}}{3}; \tag{4.3}$$

$$\sigma^n = \frac{V_n}{m}. \tag{4.4}$$

Здесь и в дальнейших формулах нижний индекс означает соотнесение к нижнему временному слою, верхний — к верхнему.

Давление P^n вычисляется из следующей системы уравнений:

$$\begin{array}{l} E^n - E_n + \frac{P^n+P_n}{2}(\sigma^n - \sigma_n) = 0, \\ E^n = E(P^n, \sigma^n), \\ E_n = E(P_n, \sigma_n). \end{array} \tag{4.5}$$

Давление в вершинах многоугольника соответствия

$$p^i = \frac{P^n + P^i + P^{i+1}}{3}. \tag{4.6}$$

Линейно интерполируя давление с вершин на грани тора, получаем силу, действующую на него. Используя разностную форму уравнений Ньютона, получаем аппроксимацию уравнений Эйлера:

$$\frac{u^n - u_n}{\Delta t} = \frac{1}{6}\sum_{i=1}^{s_n}(y_i - y_{i+1})[(2y_i + y_{i+1})p^i + (y_i + 2y_{i+1})p^{i+1}]; \tag{4.7}$$

$$\frac{v^n - v_n}{\Delta t} = \frac{1}{6}\sum_{i=1}^{s_n}(x_{i+1} - x_{i-1})[(2y_{i-1} + y_{i+1})p^i + (y_i + 2y_{i+1})p^{i+1}] + \\ + \sum_{i=1}^{s_n} S_i \frac{P^n + p^i + p^{i+1}}{3}. \tag{4.8}$$

Отсюда, вычислив скорости u^n, v^n, получаем координаты сдвинутых точек:

$$\begin{array}{rcl} X^n & = & X_n + u^n \Delta t, \\ Y^n & = & Y_n + v^n \Delta t. \end{array} \tag{4.9}$$

Этим расчет временного шага внутренней точки заканчивается.

2.5. Особенности расчета граничных точек

Достаточно качественные способы расчета граничных точек для рассматриваемой методики не разработаны до сих пор. Проще рассчитываются задачи с заданной нормальной скоростью. Задание граничного давления может приводить к сильным немонотонностям в геометрии границ, поэтому в рассчитываемых ранее задачах приходилось использовать регуляризационные механизмы. В некоторых задачах этих явлений можно избежать, вставив дополнительные ряды фиктивных точек, что не всегда возможно.

Далее приводится одна из реализаций условий с давлением.

При начальном разбиения области G на области Дирихле может оказаться, что некоторые из них будут рассечены границей Г. Точки, которым соответствуют такие ячейки Дирихле, назовем граничными. Сам алгоритм определения граничных точек вписывается в алгоритм определения "соседей", поэтому к началу расчета полагаем, что разбиение на внутренние и граничные точки осуществлено.

Со стороны границы, противоположной рассматриваемой области G, расставляются дополнительные фиктивные точки Φ_n так, чтобы обычный алгоритм построения областей соответствия для всех точек, включая фиктивные, давал бы удовлетворительную аппроксимацию границы. В действительности фиктивные точки расставляются априорно и снабжаются признаком особого счета. Сами граничные точки при этом считаются в обычном режиме внутренних точек.

Каждая из фиктивных точек Φ_n несет следующую информацию: X_n^{Φ}, Y_n^{Φ} — координаты, Σq_i — вес, $\Sigma q_i u_i$, $\Sigma q_i v_i$ — взвешенные скорости, P_n — давление и номера двух фиктивных "соседей".

Если фиктивным точкам приписать скорости, близкие к скоростям граничных точек, то сжатие ячейки будет определяться, грубо говоря, ее половиной. В двумерном случае это приведет к чрезмерному заполнению границы внутренними точками. Только за счет этого давление в них может подняться до заданного в фиктивных точках. Избежать этих явлений можно, увеличив подвижность фиктивных точек. Реализация этого соображения осуществляется следующим образом.

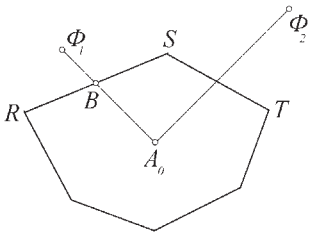

Рис. 3.

Масса граничной точки A_0 распределяется неким способом по границе ее области соответствия, но так, что $m_{RS} < m_0, m_{ST} < m_0$ (рис. 3).

Масса стенки RS приписывается точке B. Вычисляется сила, действующая на B по направлению $\Phi_1 A_0$, как разница давлений в точках

Φ_1 и A_0, помноженная на произведение площади конуса от вращения отрезка RS на косинус угла между этим отрезком и нормалью к отрезку $\Phi_1 A_0$ (проекцию площади RS на нормаль к $\Phi_1 A_0$). Полученные сила и масса позволяют вычислить ускорение точки B и приписать его точке Φ_1. Отсюда вычисляется составляющая скорости точки Φ_1 по направлению $A_0\Phi_1$. Касательная составляющая скорости берется та же, что и в точке A_0.

Если фиктивная точка обслуживает несколько граничных, то ее различные скорости взвешиваются с коэффициентами q_i типа угла зрения линии SR из Φ_1.

Чтобы предотвратить выскакивание точки A_0 из ее области соответствия, формула распределения масс сконструирована так, что $lim_{B\to A_0} m_{RS} = m_0$.

Осевые точки в отличие от граничных считаются неплохо. Основное отличие счета таких точек состоит в наличии естественной симметрии (рис. 4).

Координата x точки C полагается равной полусумме координат осевых соседей A и B. Так же вычисляется давление в данной точке. Это изменение вызвано потерей условия стыковки трех линий в вершине области соответствия из-за симметрии вращения. Второе отличие заключается в том, что возникающие в процессе счета отрицательные координаты y точки A заменяются положительными с одновременным изменением знака скорости v, т.е. происходит как бы обмен местами симметричных точек A и A'. В остальном расчет осевых точек идентичен расчету внутренних.

Таким образом, все граничные точки считаются в режиме внутренних точек, для этого им приписываются фиктивные соседи, которые расставляются априорно.

2.6. Об аппроксимации и устойчивости метода

Полностью доказательство аппроксимации системы (1.1) формулами (4.1)—(4.8) приведено в [1]. Приведем результат. Пусть L обозначает оператор левой части первого уравнения системы (1.1), а L^* — оператор левой части схемы (4.7), деленной на m. Тогда для счетного соотношения шагов порядка τ справедлива формула $(L - L^*)\{u, p, \sigma\} = O(\tau)$, т.е. в общем случае имеет место аппроксимация первого порядка. Аналогичный результат имеет место и для (4.8).

Основным вопросом, поставленным при изучении устойчивости дан-

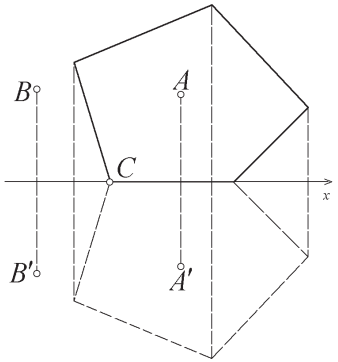

Рис. 4.

ной методики, был вопрос о том, как влияет на устойчивость приближние точки к границе области соответствия. Если бы это состояние привело к падению временного шага до 0, то расчеты по данной методике были бы значительно затруднены. В результате проведенных расчетов установлено, что близкое расположение счетной точки к границе области соответствия и выскакивание наружу слабо сказывается на устойчивости и формально не влияет на порядок аппроксимации уравнений. Таким образом, методика МЕДУЗА не обесценивается в описанных ситуациях, тем не менее их следует избегать из-за неконтролируемых потерь точности. Поэтому приближение точки к границе области соответствия приводит либо к изменению “соседства”, либо к пересадке точек вглубь области соответствия с последующими пересчетами некоторых величин. Иными словами, в конце временного шага сетка нуждается в коррекции. Соответствующие выкладки будут приведены в п. 3.3.

3. ОТОБРАЖЕНИЕ АЛГОРИТМА НА ПАРАЛЛЕЛЬНУЮ ВЫЧИСЛИТЕЛЬНУЮ СИСТЕМУ

3.1. Описание алгоритма параллельного вычисления и требования к вычислительной системе

Сформулируем требования к архитектуре ЭВМ, которая может быть пригодна для реализации метода МЕДУЗА.

Введем ряд обозначений.

Пусть $\{A_i\}_{1\le i\le I}$ — множество расчетных узлов (центров ячеек) и $\{B_j\}_{1\le j\le J}$ — множество вершин областей соответствия. Размерность этих множеств (величины I и J) связаны следующим образом. В работе [2] доказана теорема о соответствии диаграммы Вороного и триангуляции множества узлов и по ее непосредственному следствию. Диаграмма Вороного множества I точек имеет не более $2I-5$ вершин. Таким образом, имеем выражение для числа вершин J : $J=\xi I$, где $\xi=2-\frac{5}{I}\to 2$ с ростом I. В дальнейшем по умолчанию считаем, что $\xi=2$.

Предположим, что в распоряжении имеются K процессоров. Считаем, что процессоры занумерованы и обозначим их $\{\mathcal{P}_k\}_{1\le k\le K}$.

Для простоты считаем, что $\frac{I}{K}$ — целое число, и в последующих вычислениях будем считать распределение узлов по процессорам равномерным. Такое распределение узлов задается пользователем в явном виде. Полагаем, например, что $\mathcal{P}_k$ отвечает за узлы, имеющие номера $nK+k, 0\le n\le \frac{I}{K}-1$. В этом случае процессор $\mathcal{P}_1$ отвечает за узлы с номерами $1, K+1, ..., (\frac{I}{K}-1)K+1$, процессор $\mathcal{P}_2$ — за узлы с номерами $2, K+2, ..., (\frac{I}{K}-1)K+2$ и т.д. Когда мы говорим, что процессор отвечает за какой-то узел или какой-то узел содержится в процессоре, то подразумеваем, что в процессоре хранятся все характеристики данного узла. Таким образом имеем представление узла в виде записи, т.е. если A_i хранится в $\mathcal{P}_k$, то в $\mathcal{P}_k$ хранится запись $A_i.X, A_i.Y, A_i.P$ и т.д. В реальной ситуации эти величины могут иметь другой тип данных, но в данном алгоритме удобно представить их в виде записи.

Определим функцию $\nu : (i,K)\to(1,..,K)$ для нахождения номера процессора, хранящего узел A_i следующим образом: $\nu(i,K)=i-[\frac{i-1}{K}]\,K$, где квадратными скобками обозначена целая часть числа.

В результате имеем соответствие между начальным номером узла и его адресом (номером процессора). По одной нумерации однозначно находится другая.

Далее рассмотрим начало алгоритма вплоть до нумерации вершин.

Сначала происходит расстановка узлов (способом, указанным пользователем). Далее выполняется алгоритм построения диаграммы Вороного, “соседств” и вершин, в конце выполнения которого мы имеем следующую информацию: процессор $\mathcal{P}_k$ содержит записи $A_{nK+k}.X$, $A_{nK+k}.Y$, $A_{nK+k}.m_i$, $0\le n\le \frac{I}{K}-1$. Здесь $m_i, 1\le i\le s_{nK+k}$ — упорядоченный список номеров “соседей” для узла A_{nK+k}. А также имеем занумерованные вершины и списки “образующих” их узлов.

Далее рассмотрим распределение вершин по процессорам. Возникает вопрос о целесообразности данного распределения: не меньше ли будет затрачено всеми процессорами времени на простой пересчет координат для всех своих узлов, ведь при таком способе счета будет намного меньше обменов между процессорами? В принципе понятно, что при таком подходе число вычислений возрастает в три раза (что очень невыгодно с точки зрения экономии времени), так как для каждой вершины считать будут три ее ближайших смежных между собой узла. Детально данный вопрос рассмотрен в п. 3.4.

Заметим, что число вершин в общем случае не делится на K нацело, поэтому равномерного распределения не получится, но его характер будет таким же, как и при распределении узлов. Максимальное количество вершин в каждом процессоре равно $2\frac{I}{K}$. Таким образом, в процессоре $\mathcal{P}_k$ находятся вершины $mK + k, 0 \le m \le 2\frac{I}{K} - 1$. т.е. записи $B_{mK+k}.n_i, 0 \le m \le 2\frac{I}{K} - 1$. Здесь $n_i, 1 \le i \le 3$ — список номеров "образующих" узлов для вершины B_{mK+k}. Для нахождения номера процессора, хранящего вершину B_j, используется определенная функция ν.

Таким образом, распределение всех компонентов сетки по процессорам завершено.

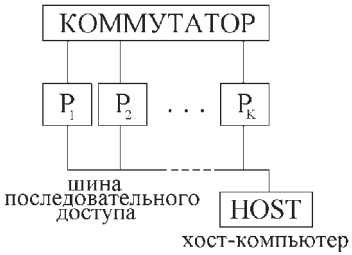

Рис. 5.

Обмен данными между процессорами производится через коммутатор, структуру которого не уточняем, но считаем, что если задана функция $\phi : \{1, .., K\} \to \{1, .., K\}$, то коммутатор способен связать процессоры $\mathcal{P}_k$ с процессорами $\mathcal{P}_{\phi(k)}$ одновременно в параллельном режиме. После коммутации происходит передача данных. Структура алгоритма должна быть такой, чтобы не возникали конфликты при записи. Несущественно, происходят ли обмены между процессорами непосредственно или через общую память.

Такая архитектура ЭВМ (см. рис. 5) ранее описывалась и использовалась в некоторых работах, например в [3].

Построим алгоритм расчета на данной вычислительной системе, реализуем его в несколько этапов.

На нулевом этапе рассматриваются распределение начальных данных и другие необходимые вопросы.

На этапах 1—4 производятся вычисления соответствующих формул из предыдущей части (4.1—4.4). Данные этапы, вместе с нулевым, исполняются только в начале программы. Таким образом, рассылка начальных данных, построение "соседств", вычисление координат вершин, площадей областей соответствия, объемов и удельных объемов производятся только один раз. Далее некоторые из заданных величин будут исправляться при коррекции сетки, но в целом их расчет закончен.

Далее считаются $iter$ итераций, по ходу которых последовательно исполняются этапы 5—9 по вычислению формул 4.5—4.9 и этапы 10—11, на которых производится коррекция сетки способами, рассмотренными в п. 3.3.

3.2. Этапы алгоритма параллельного вычисления

На **нулевом** этапе происходит задание начальных данных (1.3), (1.4).

1. На данной стадии расчета, кроме распределения узлов по процессорам, уже имеются координаты всех узлов и списки "соседей" для каждого узла.

На этом этапе предстоит вычислить координаты вершин областей соответствия $A_{nK+k}.x_j, A_{nK+k}.y_j$ для всех узлов по формуле (4.1).

Для этого необходимо всеми процессорами последовательно обработать списки своих вершин и вычислить $B_{mK+k}.x, B_{mK+k}.y$, затем переслать координаты в соответствующие процессоры.

Рассмотрим пошаговое вычисление координат $A_{nK+k}.x_j, A_{nK+k}.y_j$.

Шаг 1. Перед тем как посчитать координаты вершины $B_{mK+k}.x$, $B_{mK+k}.y$, необходимо за три этапа считать соответствующие координаты узлов $A_{m_1}.X, Y$ из $\mathcal{P}_{\nu(m_1)}$, $A_{m_2}.X, Y$ из $\mathcal{P}_{\nu(m_2)}$ и $A_{m_3}.X, Y$ из $\mathcal{P}_{\nu(m_3)}$. Процессор $\mathcal{P}_{\nu(mK+k)}$ может через коммутатор подключиться к процессору $\mathcal{P}_{\nu(m_1)}$ и считать $A_{m_1}.X, Y$ и т.д. Все процессоры делают это одновременно и синхронно.

Шаг 2. Происходит собственно вычисление координат вершины

B_{mK+k}. Все процессоры проводят вычисления синхронно, так как вычисляют одну и ту же формулу.

Шаги 1 и 2 выполняются, пока все процессоры не обработают все свои врешины. При равномерном распределении вершин все процессоры закончат работу одновременно. Таким образом, для подсчета координат всех вершин необходимо J/k последовательных повторений шагов 1 и 2.

Шаг 3. Процессор $\mathcal{P}_{\nu(nK+k)}$ начинает последовательное считывание величин $A_{nK+k}.x_j$, $A_{nK+k}.y_j$ посредством подключения через коммутатор к $\mathcal{P}_{\nu(m(n_j,n_{j+1},nK+k))}$, где n_j, n_{j+1} — номера соответствующих "соседей", $m(n_1, n_2, n_3)$ — функция связи.

Таким образом, после того как посчитаны координаты всех вершин, следует считывание данных. Все процессоры последовательно обрабатывают списки своих узлов. Для каждого считываются соответствующие координаты. Все процессоры делают описанные действия параллельно, на считывание координат одной вершины тратится один этап. Так как в этом случае имеем зависимость от числа всех вершин для всех узлов в процессоре, то общее время исполнения 1-го этапа будет зависеть от их наибольшего числа.

2. Вычисление площадей областей соответствия. На самом деле, как видно из формул, вычисляются только все составляющие площади ячейки (площади треугольников). При вычислении каждый процессор обрабатывает все свои узлы, используя при этом уже имеющиеся координаты вершин, по формулам (4.2). Таким образом считаются $A_{nK+k}.S_j$, где $1 \le j \le s_{nK+k}$. На данном этапе все процессоры работают параллельно. Время работы будет зависеть от времени вычисления процессором с максимальным числом всех вершин.

3. Нахождение объемов $A_{nK+k}.V^*$ производится в параллельном режиме, при этом используются уже посчитанные координаты вершин и площади частей ячеек. Каждый процессор обрабатывает все свои узлы, считая при этом одну формулу (4.3) для каждого из них. И хотя мы имеем равномерное распределение узлов, время исполнения данного этапа также будет зависеть от процессора с наибольшим числом вершин всех узлов, так как в формуле (4.3) фигурирует суммирование по всем вершинам.

4. Удельные объемы $A_{nK+k}.\sigma^*$ считаются параллельно и за сравнительно небольшое, одинаковое для всех узлов время. Формула (4.4).

На последующих этапах производятся итерационные вычисления. Сначала находятся значения гидродинамических величин, потом производится проверка новых координат на близость к границе и коррекция сетки в необходимых местах.

5. Вычисляем давление $A_{nK+k}.P^*$ в центре ячейки из неявной схемы. При этом заметим, что процесс нахождения давления не зависит от числа вершин в областях соответствия. Система (4.5). Сделаем допущение, что время вычисления независимо от его вида (будь то простая формула или некоторый прцесс, включающий в себя итерации и подстановки) и в последовательном, и в параллельном режиме для всех узлов одно и то же и является постоянным. Все процессоры на данном этапе работают параллельно.

6. На данном этапе необходимо посчитать давление $A_{nK+k}.p_j$ в каждой вершине области соответствия. Сами вычисления (расчетная формула (4.6)) и их порядок идентичны вычислениям на 1-м этапе (координаты). Сначала так же за три шага (для каждой вершины) считываются значения давлений в "образующих" узлах. Потом вычисляется давление в вершине и осуществляется переход к следующей вершине. После того как обработаны все вершины, начинается считывание данных (обрабатываются все узлы). Время исполнения на данном этапе будет аналогично времени исполнения 1-го этапа.

7. Находим скорость $A_{nK+k}.u^*$ по формуле (4.7). При этом все процессоры вычисления производят параллельно, в формуле имеется зависимость от числа вершин области соответствия для каждого узла. Время работы на данном этапе также будет зависеть от этого числа.

8. По формуле (4.8) находим скорость $A_{nK+k}.v^*$. Эта формула похожа на предыдущую, в ней также имеется подсчет суммы по всем вершинам.

9. Подсчет координат $A_{nK+k}.X^*, Y^*$ состоит из вычислений по одинаковым формулам (4.9) и производится на одном этапе. Все процессоры в параллельном режиме обрабатывают все свои узлы.

На этом расчет временного шага заканчивается. После вычисления новых координат необходимо произвести проверку каждого узла на близость к границе и откорректировать сетку.

3.3. Коррекция сетки

Далее необходимо разобрать ситуацию с коррекцией сетки. С определением узлов, центры масс в которых сместились близко к границе (проверкой на близость), и их пометкой все понятно — необходимо просто найти расстояния до каждой стороны области соответствия (для каждого узла) и сравнить их с некоторым параметром ϵ, который зависит от длины временного шага, скоростей точек и размеров сетки. Случаи, когда ϵ будет больше, чем найденное расстояние, необходимо пометить.

10. Процессор $\mathcal{P}_k$ параллельно обрабатывет все свои узлы, для каждого вычисляет расстояния до всех сторон границы области соответствия $A_{nK+k}.\rho_j$ и сравнивает их с параметром ϵ. При этом используются следующие (либо подобные) формулы (индексы опущены):

$\rho_j = |A_jX + B_jY + C_j|$, где X, Y — координаты центра ячейки;

$A_j = \frac{1}{x_{j+1}-x_j}$, $B_j = -\frac{1}{y_{j+1}-y_j}$;

$C_j = -\frac{x_j}{x_{j+1}-x_j} + \frac{y_j}{y_{j+1}-y_j}$.

После сравнения необходимо пометить узел. Для этого введем некоторый целочисленный признак $A_{nK+k}.\rho$: либо $\rho = 0$, тогда узел находится в обычном состоянии, либо $\rho = j$, где j — номер стороны границы. После переопределения ρ на очередном шаге будем иметь информацию о том, к какой именно стороне приблизилась точка (в некоторых способах корректировки эта информация необходима). Время исполнения на данных этапах будет зависеть от наибольшего количества вершин всех узлов в процессоре.

Самый простой с точки зрения формул для координат способ — пересадка точки вглубь области соответствия с последующими интерполяциями. Методы и формулы интерполяций могут быть самыми различными, и в данной работе их анализ не приводится. Время, которое будет затрачено на интерполяции, для всех узлов считаем одинаковым. Процессоры исполняют данный этап в параллельном режиме, последовательно обрабатывая списки узлов, помеченных ранее. Общее время работы зависит от количества помеченных узлов $I_{\text{п}} = bpI$, где величина bp определяет долю помеченных узлов. Чтобы не усложнять расчеты, будем считать, что шаг по времени выбран так, что bp сравнительно невелико по сравнению с 1. При таком подходе зависимость времени исполнения от распределения помеченных узлов по процессорам не сильная, приближенно будем считать, что они распределены равномерно.

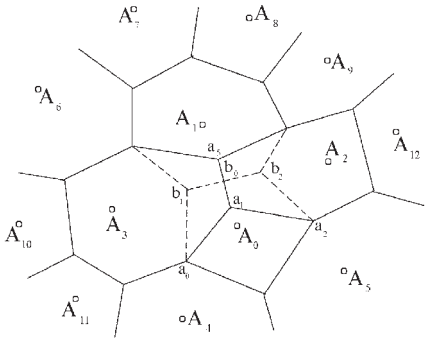

Рис. 6.

Далее приведем выдержки из способа корректировки "изменение соседства", предложенного авторами применения методики МЕДУЗА для решения газодинамических задач [1].

Изменение "соседства" предусмотрено структурой графа, которая может быть переменной. Техника реализации изменения структуры не очень сложна, поскольку новые "соседи" могут появиться лишь из числа "соседей соседей". Можно пользоваться лишь этим приемом, и тогда пересадка не потребуется. При перестройке "соседств" скачкообразно меняются площади областей соответствия, в связи с чем резко меняется плотность и другие гидродинамические параметры, что приводит к усилению пиков немонотонности. Для их уменьшения используются интерполяционные приемы, укладывающиеся в рамки законов сохранения. В целом в этом случае оценка влияния интерполяций на конечный результат расчета труднодоступна, но известно, что чем меньше произведено интерполяций на одну точку за расчет, тем надежнее результаты.

Техника изменения "соседств" заключается в следующем (см. рис. 6).

Пусть для точек A_0, A_1, A_2, A_3 указаны следующие соседства:

для $A_0 : A_2, A_3, A_4, A_5,$

для $A_1 : A_9, A_8, A_7, A_6, A_3, A_2,$

для $A_2 : A_0, A_5, A_{12}, A_9, A_1, A_3,$

для $A_3 : A_{11}, A_4, A_0, A_2, A_1, A_6, A_{10}.$

Пусть теперь точка A_0 находится близко к границе a_0a_1 и в то же время близко к вершине a_1.

По вершине a_1 определяются те "соседи" точки A_0, которые "образуют" данную вершину. В примере это точки A_2 и A_3. По свойству областей соответствия для точек A_2 и A_3 найдется "сосед" A_1 такой, что он не будет "соседом" A_0. Эта точка и вводится в "соседство" к A_0, а точки A_2 и A_3 из взаимного "соседства" выводятся. Исправленная таблица "соседств" теперь будет выглядеть так:

для $A_0 : A_2, A_1, A_3, A_4, A_5$,
для $A_1 : A_9, A_8, A_7, A_6, A_3, A_0, A_2$,
для $A_2 : A_0, A_5, A_{12}, A_9, A_1$,
для $A_3 : A_{11}, A_4, A_0, A_1, A_6, A_{10}$.

Перестройка свелась к тому, что в "соседство" A_0 введена A_1, в "соседство" A_1 введена A_0, из "соседства" A_2 выведена A_3, из "соседства" A_3 выведена A_2. При этом все свойства областей соответствия сохранились. Этот пример показывает, что граф "соседств" в процессе перестройки числа ребер не меняет.

Интерполяции, сопутствующие перестройке, состоят в следующем. Масса $a_0a_1b_0b_1$, рассчитанная по точке A_3, из массы этой точки удаляется, а к массе точки A_0 добавляется. То же самое относится и к количествам движения и энергиям. Аналогичные приемы применяются ко всем четырем перестроенным точкам.

Данную процедуру разумно проводить, когда исходная точка приближается к границе, но не к вершине, что означает попарное сближение точек. В таких случаях разумнее проводить раздвижку сблизившейся пары без изменения "соседств". Интерполяция остается вследствие изменения областей соответствия.

Указанные процедуры при расчете газов с различными свойствами приводят к необходимости рассчитывать ячейки смешанных составов и, в отличие от (2.1), хранить указания о разных составах смешанных ячеек.

Одним из слабых мест данной методики, как отмечено в [1], является недоказанность корректности механизма перестроек "соседств". Действительно, в указанном выше примере точка A_3 может приблизиться к границе a_1a_5 (и к вершине a_5) одновременно с A_0. И тогда после проведения указанной выше перестройки A_3 окажется за пределами своей области соответствия. Потребуется пересаживать точку. Из подобных затруднений авторы вышли путем снабжения узлов особым признаком, который принимал несколько целочисленных значений (характеризующих состояние узла). Изменение его значения для каждого узла зависе-

ло от значений этого признака у всех его "соседей" (а в ряде случаев и от состояния всех "соседей соседей"). Указанный способ не эффективен при вычислении на параллельной вычислительной системе, так как все узлы в этом случае должны обрабатываться последовательно несколько раз.

Напомним, что основа механизма перестройки "соседств" — свойство б) из п. 2.2. В работе [1] утверждается, что это свойство верно, если число "соседей" каждой точки не меньше четырех, доказательство данного предложения не приводится. Тем не менее, если использовать метод построения "соседств" на основе метрической близости, нужно добавить еще одно условие: никакие четыре точки, являющиеся "соседями", не лежат на одной окружности. Проверка данного положения трудна в алгоритмическом смысле и требует затрат большого количества ресурсов. В целом такое расположение точек не влияет на точность счета, если оно не является общим для всей области, например в случае задания целочисленных координат точек и т.д.

Таким образом, получаем, что при распараллеливании метода МЕДУЗА при коррекции сетки метод перестройки "соседств" (использующийся с самых ранних реализаций) неприемлем. Гораздо выгоднее использовать способ пересадки.

3.4. Оценка времени выполнения

Отметим, что в расчетах принимается оптимальное время для последовательного случая и приведенное (с округлением вверх) для параллельного. Так, например, в случае, когда требуется выполнить какие-либо (одинаковые) действия со всеми "соседями" (вершинами) для каждого узла, необходимо $\sum_{i=1}^{I} s_i$ таких действий. При этом затраченное время в последовательном режиме будет равно произведению этого числа на время исполнения данного действия. В параллельном случае в различных процессорах находятся узлы с различными числами s_i, поэтому некоторые (имеющие узлы с меньшим количеством вершин) процессоры завершат работу раньше и общее время исполнения будет зависеть от времени, затраченного процессорами с наибольшим количеством вершин всех узлов. Такая ситуация возникает при обычном распределении узлов по процессорам или, например, в ситуации, когда $I = K$. Таким образом, общее время равно произвдению времени исполнения одного действия на $max_k(\sum_{n=0}^{\frac{I}{K}-1} s_{nK+k})$, где s_j — количество вершин

j-го узла. Это число оценивается сверху, оно не больше чем $\frac{I}{K}s_{max}$, где $s_{max} = max_{i=1}^{I} s_i$ — максимальное число вершин для всех узлов.

Формулы и обозначения:

$iter$ — число итераций;

t_j — время исполнения на j-м этапе в параллельном режиме;

T_j — время исполнения на j-м этапе в последовательном режиме;

$t = \sum_{j=1}^{9} t_j, T = \sum_{j=1}^{9} T_j$ — общее время исполнения (соответственно);

$k_j = \frac{T_j}{t_j}$ — ускорение на j-м этапе;

$k = \frac{T}{t}$ — общее ускорение;

τ_k — время обмена (считывание) данными, при котором каждый процессор считывает k величин за одно соединение (в данной реализации $k = 1, 2$);

Σ — краткая запись $\sum_{i=1}^{I}$;

c_k — некоторые константы времени выполнения процессором соответствующих формул, зависящие только от времени выполнения процессором простейших математических действий, т.е. от свойств конкретного процессора.

На **нулевом** этапе производится рассылка начальных данных, построение "соседств", t_0 и T_0 — время, которое необходимо затратить на данный этап в параллельном и последовательном режимах соответственно.

1. Координаты вершин:
$t_1 = (2c_1 + 3\tau_2)J\frac{1}{K} + I\tau_2 s_{max}\frac{1}{K}$;
$T_1 = 2c_1 J$;
$k_1 = \frac{2c_1 JK}{(2c_1+3\tau_2)J+I\tau_2 s_{max}} =$
$= \frac{2c_1\xi}{(2c_1+3\tau_2)\xi+\tau_2 s_{max}}K.$

2. Площади частей области соответствия:
$t_2 = c_2 I s_{max}\frac{1}{K}$;
$T_2 = c_2\Sigma s_i$;
$k_2 = \frac{s_{mid}}{s_{max}}K.$

3. Объем:
$t_3 = c_{31} I s_{max}\frac{1}{K} + c_{32} I\frac{1}{K}$;
$T_3 = c_{31}\Sigma s_i + c_{32} I$;
$k_3 = \frac{c_{31}s_{mid}+c_{32}}{c_{31}s_{max}+c_{32}}K.$

4. Удельный объем:
$t_4 = c_4 I\frac{1}{K}$;

5. Давление в узлах неявной схемы:
$t_5 = c_5 I\frac{1}{K}$;

$T_4 = c_4 I;$
$k_4 = K.$

$T_5 = c_5 I;$
$k_5 = K.$

6. Давление в вершинах:
$t_6 = (c_6 + 3\tau_1) J \frac{1}{K} + I \tau_1 s_{max} \frac{1}{K};$
$T_6 = c_6 J;$
$k_6 = \frac{c_6 JK}{(c_6+3\tau_1)J + I\tau_1 s_{max}} =$
$= \frac{c_6 \xi}{(c_6+3\tau_1)\xi + \tau_1 s_{max}} K.$

7. Скорость u:
$t_7 = c_{71} I s_{max} \frac{1}{K} + c_{72} I \frac{1}{K};$
$T_7 = c_{71} \Sigma s_i + c_{72} I;$
$k_7 = \frac{c_{71} s_{mid} + c_{72}}{c_{71} s_{max} + c_{72}} K.$

8. Скорость v:
$t_8 = c_{81} I s_{max} \frac{1}{K} + c_{82} I \frac{1}{K};$
$T_8 = c_{81} \Sigma s_i + c_{82} I;$
$k_8 = \frac{c_{81} s_{mid} + c_{82}}{c_{81} s_{max} + c_{82}} K.$

9. Координаты:
$t_9 = 2 c_9 I \frac{1}{K};$
$T_9 = 2 c_9 I;$
$k_9 = K.$

10. Проверка на близость:
$t_{10} = c_{10} I s_{max} \frac{1}{K};$
$T_{10} = c_{10} \Sigma s_i;$
$k_{10} = \frac{s_{mid}}{s_{max}} K.$

11. Коррекция сетки (пересадка):
$t_{11} = 2 c_{11} bp\ I \frac{1}{K};$
$T_{11} = 2 c_{11} bp\ I;$
$k_{11} = K.$

Теперь рассмотрим вопрос о целесообразности распределения вершин по процессорам. Для его решения необходимо просуммировать время исполнения на этапах, где оно будет различаться для случев с распределением и без него. Далее нужно сравнить результаты (искать минимум).

Учитывая ранее указанные замечания, пренебрегаем временем, затраченным на начальное распределение вершин по узлам.

Время исполнения будет различным только на этапах, где требуется произвести вычисления в вершинах областей соответствия, — это 1-й и 6-й этапы алгоритма. Заметим, что на этих этапах расчетные формулы одни и те же с точностью до идентификаторов, поэтому константы времени вычисления c_i на этих этапах равны, таким образом, $c_1 = c_6 = c$.

Введем обозначения: время исполнения на соответствующих этапах в случае с распределением обозначим со штрихом, а без распределения — с двумя. Сравним величины $time_1 = t_1^{'} + t_6^{'}$ и $time_2 = t_1^{''} + t_6^{''}$:

$$\begin{aligned} time1 &= (2c_1 + 3\tau_2) J \tfrac{1}{K} + I\tau_2 s_{max} \tfrac{1}{K} + ((c_6 + 3\tau_1) J \tfrac{1}{K} + I\tau_1 s_{max} \tfrac{1}{K})\ iter = \\ &= (2c + 3\tau_2)\xi \tfrac{I}{K} + \tau_2 s_{max} \tfrac{I}{K} + (c + 3\tau_1)\xi\ iter \tfrac{I}{K} + \tau_1 s_{max}\ iter \tfrac{I}{K} = \\ &= [(\tau_2 + \tau_1 iter)(3\xi + s_{max}) + (2 + iter)\xi c] \tfrac{I}{K}; \end{aligned}$$

$$\begin{aligned} time2 &= (2c_1 + 2\tau_2) I s_{max} \tfrac{1}{K} + (c_6 + 2\tau_1) I s_{max} \tfrac{1}{K}\ iter = \\ &= (2c + 2\tau_2) s_{max} \tfrac{I}{K} + (c + 2\tau_1) s_{max}\ iter \tfrac{I}{K} = \\ &= [2(\tau_2 + \tau_1 iter) s_{max} + (2 + iter) s_{max} c] \tfrac{I}{K}. \end{aligned}$$

Рассмотрим знак разности $time_1 - time_2$:

$$\begin{aligned} sgn(time1 - time2) &= \\ &= sgn((\tau_2 + \tau_1 iter)(3\xi + s_{max}) + (2 + iter)\xi c - \\ &\quad -2(\tau_2 + \tau_1 iter)s_{max} + (2 + iter)s_{max}c) = \\ &= sgn((\tau_2 + \tau_1 iter)(3\xi - s_{max}) - (2 + iter)(s_{max} - \xi)c). \end{aligned}$$

Так как время τ_2 не может быть больше $2\tau_1$, то знакоопределяемое выражение не превосходит $(2 + iter)(3\xi - s_{max})\tau_1 - (2 + iter)(s_{max} - \xi)c$.

Тогда $sgn(time1 - time2) \le sgn((3\xi - s_{max})\tau_1 - (s_{max} - \xi)c)$.

При рассмотрении этого выражения видно: когда $3\xi \le s_{max}$, что наиболее вероятно, получаем отрицательное выражение (при этом $time_1 < time_2$), поэтому рассмотрим случай, когда $s_{max} < 3\xi$, т. е., принимая во внимание предыдущие оценки для ξ, допускаем, что $s_{max} < 6$ или $s_{max} \in \{4, 5\}$. В этом случае при делении на положительную величину — множитель при τ_1 получаем:

$$sgn(time1 - time2) = sgn(\tau_1 - \Omega\ c), \quad \Omega = (s_{max} - \xi)/(3\xi - s_{max}).$$

Покажем, что $\tau_1 < \Omega c$, т.е. докажем, что время исполнения меньше в варианте с распределением вершин. Необходимо оценить коэффициент Ω, который точно посчитать невозможно, так как он зависит от числа соседей для каждого узла. Заметим, что $\Omega(\xi)$ убывает с ростом ξ и достигает минимума при максимальном значении $\xi = 2$: $\Omega(2) = (s_{max} - 2)/(6 - s_{max})$. При подстановке различных значений s_{max} получаем, что $\Omega > (4 - 2)/(6 - 4) = 1$ в случае с одними четырехугольниками.

В реальных многопроцессорных системах время взаимодействия между процессорами соотносится со временем выполнения элементарных операций следующим образом [4,5]. Обозначим через τ_+ и τ_* время выполнения процессором сложения и умножения двух чисел соответственно. Тогда для известных систем имеют место соотношения: $2\tau_+ \le \tau_* \le 4\tau_+$ и $0,1\tau_+ \le \tau_1 \le \tau_+$. В нашем примере τ_1 сравнивается с константой c, которая не меньше чем $2\tau_+ + \tau_*$. Таким образом, с точки зрения затраченного времени выгоднее распределять вершины по процессорам.

Теперь рассмотрим коэффициент ускорения.

Сначала найдем t и T. Известно, что $c = c_1 = c_6$, введем обозначения:

$a = c_{71} + c_{81} + c_{10}$,

$b = c_5 + c_{72} + c_{82} + 2c_9 + 2c_{11}\ bp$,

$d = c_2 + c_{31}$,

$e = c_{32} + c_4$.

Тогда

$$\begin{aligned} t &= t_0 + (2c_1 + 3\tau_2)\xi\tfrac{I}{K} + \tau_2 s_{max}\tfrac{I}{K} + c_2 s_{max}\tfrac{I}{K} + c_{31}s_{max}\tfrac{I}{K} + c_{32}\tfrac{I}{K} + \\ &\quad + c_4\tfrac{I}{K} + [c_5\tfrac{I}{K} + (c_6 + 3\tau_1)\xi\tfrac{I}{K} + \tau_1 s_{max}\tfrac{I}{K} + c_{71}s_{max}\tfrac{I}{K} + c_{72}\tfrac{I}{K} + \\ &\quad + c_{81}s_{max}\tfrac{I}{K} + c_{82}\tfrac{I}{K} + 2c_9\tfrac{I}{K} + c_{10}s_{max}\tfrac{I}{K} + 2c_{11}\ bp\tfrac{I}{K}]\ iter = \\ &= t_0 + \tfrac{I}{K}[(\tau_2 + c_2 + c_{31})s_{max} + (\tau_1 + c_{71} + c_{81} + c_{10})s_{max}\ iter(c_5 + (c_6 + \\ &\quad + 3\tau_1)\xi + c_{72} + c_{82} + 2c_9 + 2c_{11}\ bp)\ iter + (2c_1 + 3\tau_2)\xi + c_{32} + c_4] = \\ &= t_0 + \tfrac{I}{K}[(\tau_2 + d)s_{max} + (\tau_1 + a)s_{max}\ iter + ((c + 3\tau_1)\xi + b)\ iter + \\ &\quad + (2c + 3\tau_2)\xi + e], \end{aligned}$$

$$\begin{aligned} T &= T_0 + 2c_1\xi\ I + c_2 s_{mid}\ I + c_{31}s_{mid}\ I + c_{32}\ I + c_4\ I + [c_5\ I + c_6\xi\ I + \\ &\quad + c_{71}s_{mid}\ I + c_{72}\ I + c_{81}s_{mid}\ I + c_{82}\ I + 2c_9\ I + c_{10}s_{mid}\ I + \\ &\quad + 2c_{11}\ bp\ I]\ iter = \\ &= T_0 + I[ds_{mid} + as_{mid}\ iter + (c\xi + b)\ iter + 2c\xi + e]. \end{aligned}$$

Далее введем ряд предложений. На нулевом этапе, при рассылке начальных данных, затраченное время одинаково и в параллельном, и в последовательном случае и сравнительно мало по сравнению со временем исполнения алгоритма на последующих этапах. Поэтому при вычислении ускорений пренебрегаем длительностью нулевого этапа, т. е. величинами t_0 и T_0.

Найдем выражения для коэффициента ускорения, для чего введем дополнительные обозначения :

$\alpha(iter) = d + iter\ a$,

$\beta(iter, \tau_1, \tau_2) = \tau_1 iter + \tau_2$,

$\gamma(iter) = e + iter\ b + iter\ c\xi + 2c\xi$.

Тогда при подстановке получим

$$\begin{aligned} k &= \frac{T}{t} = \\ &= K\ \frac{ds_{mid} + as_{mid}\ iter + (c\xi + b)\ iter + 2c\xi + e}{(\tau_2 + d)s_{max} + (\tau_1 + a)s_{max}\ iter + ((c + 3\tau_1)\xi + b)\ iter + (2c + 3\tau_2)\xi + e} = \\ &= K\ [\frac{\alpha s_{mid} + \gamma}{\alpha s_{max} + \beta s_{max} + 3\beta\xi + \gamma}] \end{aligned} \qquad (1)$$

Таким образом, получено выражение для коэффициента ускорения при распараллеливании методики МЕДУЗА на многопроцессорной машине указанной архитектуры. Интересен тот факт, что число точек I в него не входит.

Далее приведем приблизительную оценку для коэффициента ускорения.

Сразу заметим, что функция β относительно мала по сравнению с α и γ, так как в ней в качестве временных коэффициентов при $iter$ присутствует только τ_1, который мал по сравнению с a, b и c. Свободные члены в α и γ также превосходят τ_2 в несколько раз.

Величину s_{mid} можно оценить в среднем как $(s_{max} + s_{min})/2$. Такая оценка будет неплохой для большинства случаев при достаточном количестве точек. С другой стороны, $s_{min} = 4$ практически в любой расстановке точек. Подставим $s_{mid} = (s_{max} + 4)/2$ в (1):

$$k = K\ [(\frac{1}{2}\alpha(s_{max} + 4) + \gamma)/(\alpha s_{max} + \beta s_{max} + 3\beta\xi + \gamma)] =$$
$$= K\ [1 - \frac{1}{2}(\alpha s_{max} - 4\alpha + (2\beta s_{max} + 6\beta\xi))/(\alpha s_{max} + \gamma + (\beta s_{max} + 3\beta\xi))].$$

В этом выражении члены с относительно малой β сгруппированы отдельно. Для достижения более четкой оценки ими можно пренебречь. В таком случае

$$k \approx K\ [1 - \frac{1}{2}\frac{s_{max} - 4}{s_{max} + \frac{\gamma}{\alpha}}] = K\ [1 - \varepsilon(s_{max}, \frac{\gamma}{\alpha})].$$

4. ЗАКЛЮЧЕНИЕ

Несмотря на многочисленные трудности, которые возникают при создании программ расчета по методике МЕДУЗА, к настоящему времени уже созданы функционирующие программы. В частности в [1] проанализированы результаты работы и приведены сравнения с расчетами по другим методикам. Успешным является применение методики МЕДУЗА к решению различных задач ядерной физики, примером служат расчеты уравнений теплопроводности И.Д. Софронова [6].

Отметим, что предварительные расчеты времени работы параллельного алгоритма дали вполне удовлетворительные результаты. Полученные выражения для времени и коэффициента ускорения характеризуют сложность данной методики. Вместе с тем очевидны многие ее недостатки, такие, например, как нечеткая доказанность в отдельных случаях и отсутствие строгого и эффективного алгоритма. Все это влияет на создание полноценной функционирующей программы. В рамках работы в случае получения приемлемых результатов планируется создание программы, реализующей алгоритм. В настоящее время создана некоторая

ее часть, описаны структуры (на языке C++), которые должны использоваться в программе, рассмотрена реализация построения диаграммы Вороного, основанного на сортировке. Для получения каких-либо значимых результатов требуются большие усилия не только в программировании алгоритма, но и в доработке различных его частей.

В дальнейших теоретических исследованиях можно выделить два направления. Первое связано с разработкой эффективных алгоритмов для расчета вообще, включая поиск более четких путей (с доказательствами), второе — с распараллеливанием этих алгоритмов на различных архитектурах ЭВМ, например матричной и гиперкубе.

СПИСОК ЛИТЕРАТУРЫ

1. **Бабенко К.И.** Теоретические основы и конструирование численных алгоритмов задач математической физики. — М.: Наука, 1979. — 295 с.
2. **Препарата Ф., Шеймос М.** Вычислительная геометрия. — М.: Мир, 1989. — 478 с.
3. **Лобив И.В., Мурзин Ф.А.** О распараллеливании PIC-метода. //Проблемы систем информатики и программирования. — Новосибирск: ИСИ СО РАН, 1998. — С. 146—155.
4. **Системы** параллельной обработки: Сб. статей/ Под ред. Д. Ивенса. — М.: Мир, 1985. — 415 с.
5. **Высокоскоростные** вычисления: Сб. статей/ Под ред. Я. Ковалика. — М.: Радио и связь, 1988. — 385 с.
6. **Софронов И.Д.** О численном решении уравнений теплопроводности на неортогональных сетках// Тр. Моск. Междунар. математического конгресса, 1966 г., Секция 14. — М.: Изд. МГУ, 1966.