УДК 519.63 doi 10.26089/NumMet.v16r105

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ДЛЯ РЕШЕНИЯ 2D-УРАВНЕНИЯ ПУАССОНА В КОНТЕКСТЕ НЕСТАЦИОНАРНЫХ ЗАДАЧ

Н. В. Снытников[1]

Предложен новый параллельный метод решения задачи Дирихле для уравнения Пуассона в контексте нестационарных задач математической физики. Метод основан на декомпозиции прямоугольной декартовой области решения в одном направлении, решении уравнения Пуассона в каждой подобласти прямым методом и сопряжении подобластей с помощью быстрого вычисления потенциала выделенного слоя "экранирующих" зарядов. Тестовые эксперименты, проведенные на суперкомпьютерах Межведомственного суперкомпьютерного центра и Сибирского суперкомпьютерного центра, показали хорошую масштабируемость алгоритма.

Ключевые слова: уравнение Пуассона, задачи Дирихле, декомпозиция области, гравитационный потенциал, звездная динамика, параллельное программирование, масштабируемость алгоритмов.

1. Введение. Для некоторых задач астрофизики и физики плазмы возникает необходимость решать систему дифференциальных уравнений, описывающую нестационарную динамику вещества в гравитационном или электромагнитном поле. Такая система может включать в себя уравнение Пуассона для гравитационного или электростатического потенциала. Одним из типичных примеров является система уравнений звездной динамики с учетом самосогласованного гравитационного поля, описывающая движение звезд в галактиках или пыли в протопланетном диске [1, 2].

Эта система состоит из уравнения Власова (известного также как бесстолкновительное уравнение Больцмана) для функции распределения вещества $f = f(t, \boldsymbol{r}, \boldsymbol{u})$, зависящей от времени t, пространственной координаты $\boldsymbol{r}$ и вектора скорости $\boldsymbol{u}$, с заданным начальным значением $f^0(\boldsymbol{r}, \boldsymbol{u})$:

$$\frac{\partial f}{\partial t} + \boldsymbol{u}\,\frac{\partial f}{\partial \boldsymbol{r}} - \nabla\Phi(t, \boldsymbol{r})\,\frac{\partial f}{\partial \boldsymbol{u}} = 0, \quad f(0, \boldsymbol{r}, \boldsymbol{u}) = f^0(\boldsymbol{r}, \boldsymbol{u}),$$

и уравнения Пуассона для гравитационного потенциала $\Phi = \Phi(t, \boldsymbol{r})$ с краевыми условиями для изолированных систем:

$$\Delta\Phi(t, \boldsymbol{r}) = 4\pi G\rho(t, \boldsymbol{r}), \quad \Phi(t, \boldsymbol{r})\big|_{|\boldsymbol{r}|\to\infty} = 0,$$

где G — гравитационная постоянная.

Замыкает систему уравнение для плотности $\rho = \rho(t, \boldsymbol{r})$, которая связана с функцией распределения f следующим уравнением: $\rho(t, \boldsymbol{r}) = \int\limits_{\boldsymbol{u}} f(t, \boldsymbol{r}, \boldsymbol{u})\, d\boldsymbol{u}$.

Численное решение рассматриваемой системы, как правило, выполняется с помощью метода расщепления по физическим процессам. В этом случае для решения уравнения Власова используется метод частиц в ячейках [3, 4], а для уравнения Пуассона — конечно-разностный метод. При этом для постановки корректной задачи Дирихле на границе конечной расчетной области могут быть использованы несколько альтернативных подходов — аппроксимация потенциала на границе [5, 6], метод Джеймса [7], метод свертки [4] и др.

Перечислим некоторые возникающие на практике трудности, характерные для численного моделирования нестационарных задач астрофизики.

1. Нестационарные процессы в галактиках или протопланетных дисках требуют учета самосогласованного поля и, соответственно, вычисления гравитационного потенциала на каждом временно́м шаге. Общее количество необходимых временны́х шагов для типичного численного эксперимента может составлять сотни тысяч.

2. Необходимо использовать достаточно подробную сетку для того, чтобы воспроизводить процессы, происходящие одновременно на больших и малых масштабах [1, 8]. Например, пространственное разрешение области единичного размера до величин $10^{-3} \div 10^{-5}$ влечет необходимость использования нетри-

[1] Институт вычислительной математики и математической геофизики СО РАН, просп. Лаврентьева, 6, 630090, Новосибирск; науч. сотр., e-mail: nik@ssd.sscc.ru

виальных алгоритмов для сгущающихся сеток (в англоязычной терминологии Adaptive Mesh Refinement, AMR) либо очень большого числа узлов сетки.

3. С точки зрения решения реальных практических задач требуется проводить десятки и сотни серийных численных экспериментов с вариациями начальных параметров математической модели. При этом типичный численный эксперимент должен выполняться не более чем за 24 часа (иначе усложняется работа по интерпретации этих результатов в ходе совместной работы с физиками – постановщиками задач).

Одним из способов преодоления всех этих трудностей является разработка эффективного параллельного алгоритма, предназначенного для расчетов на суперкомпьютерах и хорошо масштабируемого от сотен до десятков тысяч процессорных ядер. Несмотря на то что уравнение Пуассона является одним из базовых и хорошо изученных уравнений математической физики, интерес к созданию эффективных однопроцессорных и многопроцессорных методов решения не ослабевает до сих пор. Детальные обзоры по методам можно найти в многочисленных зарубежных и отечественных монографиях и статьях (например, [9, 10]). Тем не менее, в рамках настоящего исследования представляется полезным перечислить некоторые работы, направленные на создание параллельных алгоритмов, использующих прямые методы. Подобные алгоритмы могут рассматриваться в качестве эффективной альтернативы широко используемым многосеточным методам или декомпозиции по методу Шварца.

Так, в статьях [11–13] предложен метод локальных коррекций для декомпозиции области. Метод основан на одновременном представлении потенциала в виде двух частей: дальнодействующего потенциала, который вычисляется с помощью схемы четвертого порядка во всей области на грубой сетке, и близкодействующего потенциала, который вычисляется с помощью метода второго порядка. Для сопряжения решений используется метод Джеймса [7]. В работах [14, 15] предложен алгоритм сгущающихся сеток (AMR), позволяющий специальным образом учитывать и устранять разрывы в нормальных производных потенциала, которые возникают на границе подобластей с разными уровнями детализации, и при этом сохранять второй порядок сходимости. Интересные результаты по созданию параллельного алгоритма для решения серий трехдиагональных и блочно-трехдиагональных линейных систем с одинаковыми матрицами и разными правыми частями получены в работах [16, 17]. В отличие от классического алгоритма [18], этот алгоритм хорошо масштабируется по крайней мере до нескольких тысяч процессоров. И наконец, нельзя не отметить популярный метод декомпозиции для уравнения Пуассона, основанный на алгоритме быстрого преобразования Фурье и транспозиции подобластей. Основной недостаток этого метода (необходимость передачи в некоторых случаях десятков и сотен мегабайт данных в межпроцессорных коммуникациях) уверенно компенсируется простотой реализации и хорошей производительностью для большинства реальных практических задач. Подробное обсуждение этого метода и его оптимизаций может быть найдено, например, в работах [19, 20]. Кроме того, программная реализация доступна во многих некоммерческих и коммерческих программных пакетах [21–23].

В настоящей статье предложен новый метод решения задачи Дирихле для двумерного уравнения Пуассона в прямоугольной декартовой области на равномерных сетках. В дальнейшем он может быть использован в качестве вспомогательного метода для решения трехмерного уравнения Пуассона. Для ускорения производительности параллельного алгоритма используется свойство нестационарности задачи. Известно, что это свойство является существенным фактором, который необходимо рассматривать при выборе метода решения и создании эффективного алгоритма. Так, в работах [6, 8] показано, что можно серьезно улучшить производительность итерационного метода решения уравнения Пуассона, если брать начальное приближение с предыдущего шага.

Представленный в нашей статье алгоритм использует прямой метод решения, в его основе лежат следующие идеи.

1. Пространственная декомпозиция области решения на непересекающиеся подобласти и независимое решение уравнения Пуассона для каждой подобласти.

2. Сопряжение двух смежных подобластей на основе метода вычисления потенциала граничного слоя по аналогии с методом [14].

3. Построение древовидной иерархии подобластей для эффективного вычисления потенциала граничного слоя для произвольного количества подобластей.

4. Для повышения производительности вычисления потенциала граничного слоя используется свойство нестационарности исходной задачи, что концептуально близко подходу, предложенному в [16] для параллельной трехдиагональной прогонки.

Статья имеет следующую структуру. В разделе 2 представлено описание разработанного алгоритма (далее этот алгоритм будет называться декомпозицией с прямым сопряжением подобластей, или сокращенно DDCS (Decomposition with Direct Coupling of Subdomains). Раздел 3 посвящен обсуждению

результатов тестовых экспериментов по измерению производительности, проведенных на суперкомпьютере МВС-100К в Межведомственном суперкомпьютерном центре РАН и суперкомпьютере НКС-30Т G6 Сибирского суперкомпьютерного центра. В разделе 4 приводятся выводы.

2. Декомпозиция области с прямым методом сопряжения подобластей. Пусть задана прямоугольная двумерная область Ω с границей Γ. Требуется решить задачу Дирихле для уравнения Пуассона (1) в предположении, что задача по нахождению гравитационного потенциала Φ должна решаться многократно для разных функций ρ:

$$\Delta\Phi(\boldsymbol{r}) = \rho(\boldsymbol{r}), \quad \Phi(\boldsymbol{r})\big|_{\Gamma} = 0, \quad \boldsymbol{r} = (x, y). \tag{1}$$

Введем равномерную сетку размером $L_x \times L_y$ и с пространственными шагами h_x, h_y. Для индексации узлов сетки будем использовать $i = 0, \ldots, L_x$ и $k = 0, \ldots, L_y$. В целях удобства изложения мы ограничимся аппроксимацией оператора Лапласа с помощью 5-точечного шаблона [24]. Вместе с тем, нет никаких препятствий к выбору любого другого компактного шаблона аппроксимации.

Тогда задача (1), записанная в разностном виде с сеточными функциями Φ и ρ, примет вид

$$\frac{\Phi_{i+1,k} - 2\Phi_{i,k} + \Phi_{i-1,k}}{h_x^2} + \frac{\Phi_{i,k+1} - 2\Phi_{i,k} + \Phi_{i,k-1}}{h_y^2} = \rho_{ik}, \quad \Phi_{i,k}\big|_{(i,k)\in\Gamma} = 0. \tag{2}$$

Предполагаем, что заданы однородные краевые условия для задачи Дирихле. Это не мешает рассматривать неоднородную задачу, которая может быть сведена к однородной стандартным способом [24].

2.1. Алгоритм вычисления потенциала выделенного слоя. Определим *слой сетки с индексом* i_0 как набор сеточных узлов (i_0, k), имеющих фиксированный индекс i_0 и $k = 0, \ldots, L_y$. Рассмотрим следующую подзадачу для задачи (2).

Задача 1. Вычисление потенциала выделенного слоя. Предполагаем, что плотность ρ равна нулю в сеточной области везде, кроме одного слоя с индексом i_0: $\rho_{i_0 k} \neq 0$. Требуется найти значения потенциала $\Phi_{i_1 k}$ на слое i_1 (рис. 1).

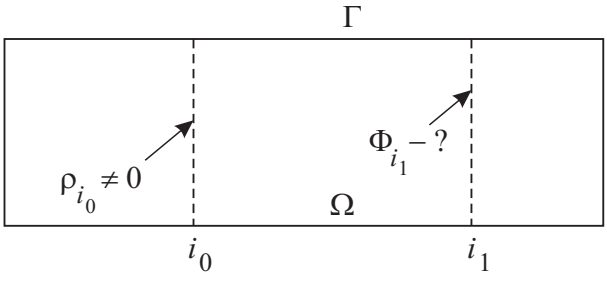

Рис. 1. Требуется вычислить потенциал на слое с индексом i_1, генерируемый зарядами с плотностью $\rho_{i_0 k}$, локализованными на слое i_0

Если решать эту задачу прямым методом разделения переменных [24], то вычислительная трудоемкость равна $O\big(L_x L_y(\log L_x + \log L_y)\big)$. Однако оказывается, что при определенных условиях трудоемкость может быть уменьшена на порядок — вплоть до $O(L_y \log_2 L_y)$. Для вывода соответствующих формул воспользуемся методом разделения переменных (методом преобразования Фурье). Представим функции Φ и ρ в виде разложения в ряд Фурье по синусам в направлении y:

$$\Phi_{ik} = \frac{1}{2L_y} \sum_{n=1}^{L_y-1} F_i(n) \sin\left(\frac{\pi n k}{L_y}\right), \quad \rho_{ik} = \frac{1}{2L_y} \sum_{n=1}^{L_y-1} G_i(n) \sin\left(\frac{\pi n k}{L_y}\right),$$

где $F_i(n) = \sum\limits_{k=1}^{L_y-1} \Phi_{ik} \sin\left(\frac{\pi n k}{L_y}\right)$ и $G_i(n) = \sum\limits_{k=1}^{L_y-1} \rho_{ik} \sin\left(\frac{\pi n k}{L_y}\right)$.

Аналогичным образом представим функции $F_i(n)$ и $G_i(n)$, разложив их в ряд Фурье по синусам в направлении x:

$$F_i(n) = \frac{1}{2L_x} \sum_{m=1}^{L_x-1} H(m, n) \sin\left(\frac{\pi m i}{L_x}\right), \quad G_i(n) = \frac{1}{2L_x} \sum_{m=1}^{L_x-1} R(m, n) \sin\left(\frac{\pi m i}{L_x}\right),$$

где $H(m,n) = \sum\limits_{i=1}^{L_x-1} F_i(n) \sin\left(\frac{\pi m i}{L_x}\right)$ и $R(m,n) = \sum\limits_{i=1}^{L_x-1} G_i(n) \sin\left(\frac{\pi m i}{L_x}\right)$.

Учитывая, что плотность ρ_{ik} равна нулю везде, кроме слоя i_0 : $\rho_{i_0 k} \neq 0$, заключаем, что функция $G_i(n)$ тоже равна нулю везде, кроме слоя i_0: $G_{i_0}(n) \neq 0$. Следовательно, последнее уравнение можно записать в виде

$$R(m, n) = G_{i_0}(n) \sin\left(\frac{\pi m i_0}{L_x}\right) = \sin\left(\frac{\pi m i_0}{L_x}\right) \sum_{k=1}^{L_y-1} \rho_{i_0 k} \sin\left(\frac{\pi n k}{L_y}\right).$$

В соответствии со стандартными формулами для решения уравнения Пуассона методом разделения переменных запишем $H(m,n) = R(m,n)S(m,n)$, где $S(m,n) = \left[-\dfrac{4\sin^2\left(\dfrac{\pi m}{2L_x}\right)}{h_x^2} - \dfrac{4\sin^2\left(\dfrac{\pi n}{2L_y}\right)}{h_y^2} \right]^{-1}$.

Подставим $R(m,n)$ в эту формулу для $H(m,n)$: $H(m,n) = S(m,n)\sin\left(\dfrac{\pi m i_0}{L_x}\right)\sum\limits_{k=1}^{L_y-1}\rho_{i_0 k}\sin\left(\dfrac{\pi n k}{L_y}\right)$. Далее подставим $H(m,n)$ в формулу для $F_i(n)$:

$$F_i(n) = \frac{1}{2L_x}\sum_{m=1}^{L_x-1} H(m,n)\sin\left(\frac{\pi m i}{L_x}\right) = \frac{1}{2L_x}\sum_{k=1}^{L_y-1}\rho_{i_0 k}\sin\left(\frac{\pi n k}{L_y}\right)\sum_{m=1}^{L_x-1} S(m,n)\sin\left(\frac{\pi m i_0}{L_x}\right)\sin\left(\frac{\pi m i}{L_x}\right). \quad (3)$$

Введем обозначения: $A_{i_0}(n) = \sum\limits_{k=1}^{L_y-1}\rho_{i_0 k}\sin\dfrac{\pi n k}{L_y}$ и $B_{i,i_0}(n) = \sum\limits_{m=1}^{L_x-1} S(m,n)\sin\dfrac{\pi m i_0}{L_x}\sin\dfrac{\pi m i}{L_x}$.

Поскольку необходимо вычислить потенциал только для конкретного слоя $i = i_1$, то из (3) следует $F_{i_1}(n) = \dfrac{1}{2L_x}A_{i_0}(n)B_{i_1,i_0}(n)$. Подставляя $F_{i_1}(n)$ в исходную формулу для $\Phi_{i_1 k}$, получим

$$\Phi_{i_1 k} = \frac{1}{2L_y}\sum_{n=1}^{L_y-1} F_{i_1}(n)\sin\frac{\pi n k}{L_y} = \frac{1}{2L_x}\frac{1}{2L_y}\sum_{n=1}^{L_y-1} A_{i_0}(n)\,B_{i_1,i_0}(n)\sin\frac{\pi n k}{L_y}. \quad (4)$$

Заметим, что выражение $B_{i_1,i_0}(n)$ зависит от области решения, параметров сетки и схемы аппроксимации, но не зависит от функции ρ. Поэтому значения $B_{i_1,i_0}(n)$ могут быть посчитаны заранее для заданной сетки и пары значений i_1, i_0. Сложность вычисления $B_{i_1,i_0}(n)$ составляет $O(L_x L_y)$, а объем памяти, требуемый для их хранения, составляет L_y значений, что незначительно по сравнению с объемом памяти, требуемой для хранения двумерных сеточных функций.

Таким образом, с учетом предвычисленных значений $B_{i_1,i_0}(n)$ трудоемкость вычисления $\Phi_{i_1 k}$ составляет $O(L_y \log L_y)$, и можно сформулировать следующий алгоритм для решения задачи 1.

Алгоритм 1. Вычисление потенциала выделенного слоя.

1. Для заданных i_1, i_0 и заданной сетки перед началом расчета вычисляем значения $B_{i_1,i_0}(n)$. Требуется $O(L_x L_y)$ действий, а размер памяти для хранения этих значений равен L_y.
2. Вычисляем $A_{i_0}(n)$ методом быстрого преобразования Фурье. Требуется $O(L_y \log L_y)$ действий.
3. Вычисляем $F_{i_1}(n) = \dfrac{1}{2L_x}A_{i_0}(n)\,B_{i_1,i_0}(n)$. Требуется $O(L_y)$ действий.
4. Вычисляем $\Phi_{i_1 k}$ по формуле (4) методом быстрого преобразования Фурье. Требуется $O(L_y \log L_y)$ действий.

Шаг 1 необходимо выполнить один раз в начале расчета, а шаги 2–4 могут для новых значений плотности, которые могут переопределяться на слое i_0. Вычислительная сложность шагов 2–4 составляет $O(L_y \log L_y)$ действий.

2.2. Алгоритм сопряжения двух подобластей. Вернемся к задаче (2). Рассмотрим следующую подзадачу.

Задача 2. Область Ω декомпозируется на две подобласти Ω_1, Ω_2 с границами Γ_1, Γ_2 и общей границей γ (рис. 2). Необходимо независимо и параллельно в каждой подобласти вычислить потенциал Φ, удовлетворяющий (2).

Запишем задачу Дирихле для уравнения Пуассона в каждой подобласти, считая, что значение потенциала на общей границе γ равно нулю, а сеточные функции ρ_1, ρ_2 совпадают с сеточной функцией ρ в своих подобластях:

$$\Delta\Phi_1 = \rho_1, \quad \Phi_1\big|_{\Gamma_1} = 0, \qquad \Delta\Phi_2 = \rho_2, \quad \Phi_2\big|_{\Gamma_2} = 0. \quad (5)$$

Решим задачи для обеих подобластей и объединим решения Φ_1 и Φ_2: $\Phi_0(\boldsymbol{r}) = \begin{cases} \Phi_1(\boldsymbol{r}), & \boldsymbol{r} \in \Omega_1, \\ \Phi_2(\boldsymbol{r}), & \boldsymbol{r} \in \Omega_2, \\ 0, & \boldsymbol{r} \in \gamma. \end{cases}$

Очевидно, что $\Phi_0(\boldsymbol{r})$ не будет удовлетворять исходной задаче (2). Во-первых, не учитывается ненулевой слой исходной плотности на границе подобластей γ (обозначим его ρ_γ). Во-вторых, функция Φ_0, являясь непрерывной в области Ω, имеет разрыв нормальной производной при пересечении границы γ: $\left.\dfrac{\partial\Phi_1}{\partial\boldsymbol{n}}\right|_\gamma \neq -\left.\dfrac{\partial\Phi_2}{\partial\boldsymbol{n}}\right|_\gamma$.

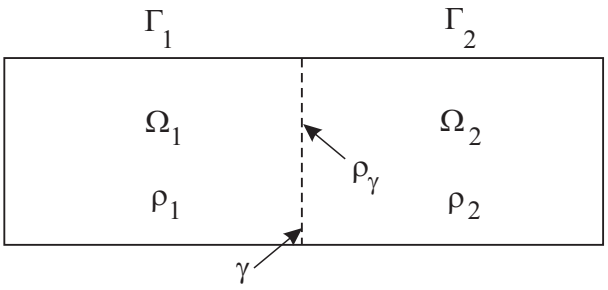

Рис. 2. Декомпозиция 2D области на две подобласти

Этот разрыв производной порождает так называемый слой "экранирующих" зарядов q^{scr}, плотность которых может быть вычислена с помощью применения разностного оператора Лапласа к значениям потенциала на границе: $\rho^{\text{scr}} := \Delta\Phi_0(\boldsymbol{r}),\ \boldsymbol{r} \in \gamma$.

Предположим теперь, что мы вычислили потенциал Φ^{scr} во всей области Ω, создаваемый плотностью $\rho_\gamma - \rho^{\text{scr}}$ и являющийся решением задачи

$$\Delta\Phi^{\text{scr}} = \rho_\gamma - \rho^{\text{scr}}, \quad \Phi^{\text{scr}}\big|_\Gamma = 0. \tag{6}$$

Тогда сумма решений $\Phi = \Phi_0 + \Phi^{\text{scr}}$ является решением задачи (2). Действительно:

$$\Delta\Phi(\boldsymbol{r}) = \Delta\Phi_0(\boldsymbol{r}) + \Delta\Phi^{\text{scr}}(\boldsymbol{r}) = \begin{cases} \Delta\Phi_1(\boldsymbol{r}) = \rho_1(\boldsymbol{r}) = \rho(\boldsymbol{r}), & \boldsymbol{r} \in \Omega_1, \\ \Delta\Phi_2(\boldsymbol{r}) = \rho_2(\boldsymbol{r}) = \rho(\boldsymbol{r}), & \boldsymbol{r} \in \Omega_2, \\ \rho^{\text{scr}} + \rho_\gamma - \rho^{\text{scr}} = \rho_\gamma \equiv \rho(\boldsymbol{r}), & \boldsymbol{r} \in \gamma. \end{cases}$$

Кроме того, потенциал Φ по своему построению удовлетворяет краевым условиям задачи (2).

Подобный подход к сопряжению подобластей для численного решения уравнения Пуассона в том или ином виде использовался во многих работах [7, 11, 12, 14, 18]. В нашем методе существенно новым является применение эффективного метода решения задачи (6). Действительно, поскольку потенциал Φ^{scr} необходимо искать во всей области Ω, непосредственное использование стандартных методов к решению задачи (6) оказывается не более экономным, чем решение исходной задачи (2).

Вместе с тем, если применить описанный в предыдущем подразделе алгоритм 1, то можно быстро вычислить потенциал Φ^{scr}_γ на слое γ, который создается зарядами с плотностью $\rho_\gamma - \rho^{\text{scr}}$, расположенными на том же слое. Далее, решая независимо задачи Дирихле для уравнения Лапласа в каждой подобласти, вычисляем Φ^{scr}_1 и Φ^{scr}_2:

$$\Delta\Phi^{\text{scr}}_1 = 0, \quad \Phi_1\big|_{\Gamma_1\setminus\gamma} = 0, \quad \Phi_1\big|_\gamma = \Phi^{\text{scr}}_\gamma \qquad \Delta\Phi^{\text{scr}}_2 = 0, \quad \Phi_2\big|_{\Gamma_2\setminus\gamma} = 0, \quad \Phi_2\big|_\gamma = \Phi^{\text{scr}}_\gamma. \tag{7}$$

В итоге приходим к следующему алгоритму для решения задачи 2.

Алгоритм 2. Сопряжение двух подобластей.

1. С помощью прямого метода решаем две задачи (5) независимо для каждой подобласти Ω_1 и Ω_2.
2. Определяем потенциал Φ_0 в Ω как объединение потенциалов Φ_1 в Ω_1 и Φ_2 в Ω_2. При параллельном решении задач на двух разных процессорах реальное конструирование Φ_0 на одном процессоре не требуется.
3. Вычисляем плотность экранирующих зарядов на границе γ с помощью применения разностного оператора Лапласа к функции Φ^0. Если используется компактная схема для оператора Лапласа (5-точечная или 9-точечная), то потребуется передача между процессорами того слоя, который является смежным со слоем γ. Иными словами, если слой γ имеет индекс i_0, то потребуется передача $\Phi_{0(i_0-1,k)}$ с первого процессора на второй и передача $\Phi_{0(i_0+1,k)}$ со второго процессора на первый.
4. Вычисляем Φ^{scr}_γ с помощью алгоритма 1.
5. Решаем независимо две задачи (7) и находим решения $\Delta\Phi^{\text{scr}}_1$ и $\Delta\Phi^{\text{scr}}_2$.
6. Для каждой подобласти конструируем решение исходной задачи в виде $\Phi = \Phi_0 + \Phi^{\text{scr}}$.

Накладными расходами этого алгоритма декомпозиции, возникающими в дополнение к вычислительной трудоемкости последовательного решения задачи (2), являются необходимость двукратного решения

задачи Дирихле (для уравнения Пуассона и уравнения Лапласа) в каждой подобласти, передача массива размера L_y между процессорами и вычисление потенциала $\Phi_\gamma^{\mathrm{scr}}$.

По сути же это означает, что за счет двукратного увеличения вычислительной нагрузки удается снизить объем межпроцессорных коммуникаций к пересылке только граничных значений. Заметим, что для решения линейных систем, возникающих в задачах (5) и (7), можно применять любые прямые методы.

2.3. Алгоритм сопряжения N подобластей. Предположим, что область Ω разделена в одном направлении на N подобластей Ω_n, $n = 0, \ldots, N-1$, каждая из которых назначена для обработки своему собственному процессору. Далее мы будем предполагать, что количество подобластей является степенью двойки: $N = 2^{N_l}$.

Алгоритм сопряжения произвольного количества подобластей не является прямым следствием алгоритма сопряжения двух подобластей. Действительно, если рассмотреть N подобластей (рис. 3), то окажется, что на границе каждой подобласти возникает выделенный слой зарядов, которые будут порождать потенциал в любой другой подобласти. Чтобы учесть это влияние, каждый процессор должен N раз применить алгоритм 1 и затем передать полученные данные соответствующим процессорам. Уже для небольшого количества процессоров ($N \sim 16$) падение производительности будет катастрофическим.

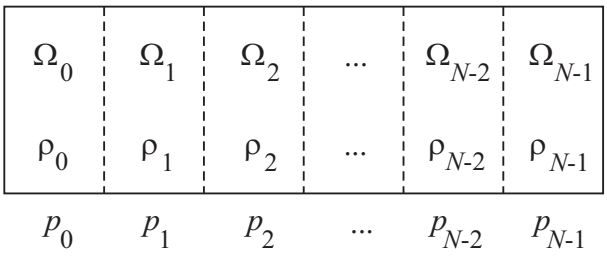

Рис. 3. Подразделение 2D области на N подобластей в направлении x

Для устранения этого недостатка разработан алгоритм, основанный на построении иерархии подобластей в виде двоичного (бинарного) дерева. Общая схема дерева для $N = 8$ представлена на рис. 4.

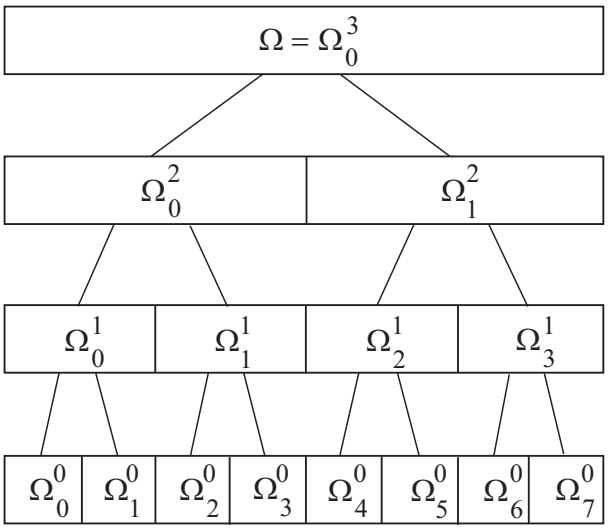

Рис. 4. Построение иерархии подобластей ($N = 8$) в виде двоичного дерева

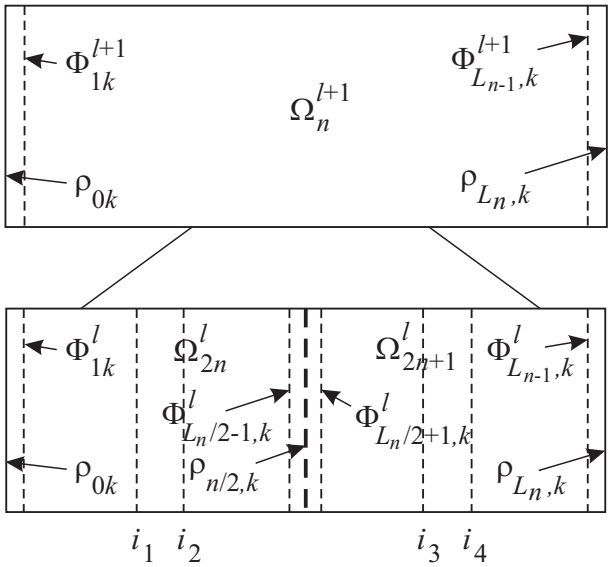

Рис. 5. Шаг перехода на следующий уровень дерева для двух подобластей

Корнем дерева является исходная область решения Ω. Будем индексировать узлы дерева (подобласти) в виде Ω_n^l, где верхний индекс l является порядковым номером уровня (глубины) подобласти в дереве, а нижний индекс n обозначает порядковый номер подобласти на данном уровне. Так, корень дерева записывается в виде $\Omega = \Omega_0^{N_l-1}$ (нумерация уровней в дереве начинается с листьев, имеющих индекс $l = 0$). Дочерними элементами корня дерева являются две одинаковых подобласти $\Omega_0^{N_l-2}$ и $\Omega_1^{N_l-2}$, а их дочерними элементами соответственно $\Omega_0^{N_l-3}$, $\Omega_1^{N_l-3}$, $\Omega_2^{N_l-3}$ и $\Omega_3^{N_l-3}$. Построение дерева продолжается до того момента, пока все N подобластей Ω_n не станут листьями в дереве. Они будут обозначаться Ω_n^0, $n = 0, \ldots, N-1$.

Пусть Φ_n^l — потенциал, который был получен на уровне l. Таким образом, Φ_n^0 является решением задач (5) для каждой подобласти-листа, а $\Phi_0^{N_l-1}$ является искомым потенциалом во всей области Ω, т.е. решением задачи (2). Нашей целью является создание алгоритма, выполняющего проход по дереву от листьев к корню и вычисляющего потенциал $\Phi_0^{N_l-1}$.

Проход по дереву начинаем с листьев (уровень $l = 0$): решаем задачи (5) для каждой подобласти и выполняем сопряжение двух соседних подобластей Ω^0_{2m} и Ω^0_{2m+1}, $m = 0, N/2 - 1$, в соответствии с алгоритмом 2, после чего переходим на уровень дерева $l = 1$. Чтобы формализовать переход от произвольного уровня l к уровню $l = 1$, рассмотрим следующую задачу (рис. 5).

Задача 3. Выполнить шаг перехода с одного уровня двоичного дерева подобластей на следующий (верхний) уровень.

Пусть даны две смежные подобласти Ω^l_{2n} и Ω^l_{2n+1} с общим родителем Ω^{l+1}_n. Обозначим через L_n число узлов подобласти Ω^{l+1}_n в направлении x. Тогда размер родительской подобласти будет равен $L_n \times L_y$, а размеры ее подобластей равны $\dfrac{L_n}{2} \times L_y$. Далее будем считать, что сеточные узлы в направлении x для первой подобласти Ω^l_{2n} проиндексированы в виде $i = 0, \ldots, L_n/2$, а для второй подобласти Ω^l_{2n+1} в виде $i = L_n/2\,, \ldots, L_n$.

Подобласть Ω^l_{2n} обрабатывается некоторым процессором с номером P_{j_1}, который хранит в своей памяти значения сеточной функции потенциала на приграничных слоях $i = 1$ и $i = L_n/2 - 1$: $\Phi^l_{1,k}$ и $\Phi^l_{L_n/2-1,k}$ (хранение сеточной функции Φ^l_{2n} для всей подобласти Ω^l_{2n} не требуется), а также исходную функцию плотности для граничных слоев: $\rho_{0,k}$ и $\rho_{L_n/2,k}$.

Вторая подобласть Ω^l_{2n+1} обрабатывается процессором с номером P_{j_2}, который хранит значения сеточной функции потенциала на приграничных слоях $i = L_n/2 + 1$ и $i = L_n - 1$: $\Phi^l_{L_n/2+1,k}$ и $\Phi^l_{L_n-1,k}$, а также исходную функцию плотности для граничных слоев: $\rho_{L_n/2,k}$ и $\rho_{L_n,k}$.

Задача состоит в том, чтобы выполнить сопряжение подобластей, воспользовавшись алгоритмами 1 и 2, так, чтобы каждый из процессоров хранил всю необходимую информацию о подобласти Ω^{l+1}_n, достаточную для возможности аналогичного сопряжения подобластей на уровне дерева $l + 1$. Затем требуется вычислить значения потенциала в подобласти Ω^{l+1}_n для приграничных слоев ($\Phi^{l+1}_{1,k}$ и $\Phi^{l+1}_{L_n-1,k}$), а затем вычислить значения потенциала Φ^{l+1} на некоторых других заранее определенных слоях с индексами i_1 и i_2 внутри подобласти Ω^l_{2n} и на слоях с заранее заданными индексами i_3 и i_4 внутри Ω^l_{2n+1}.

Алгоритм 3. Шаг перехода с одного уровня дерева на следующий уровень.

1. Выполняем межпроцессорную коммуникацию приема–передачи, соответствующую MPI-функции SendRecv: процессор P_{j_2} посылает процессору P_{j_1} массивы $\Phi_{L_n/2+1,k}$, $\Phi_{L_n-1,k}$, $\rho_{L_n,k}$ и принимает от него массивы $\Phi_{L_n/2-1,k}$, $\Phi_{1,k}$, $\rho_{0,k}$. Заметим, что плотность на границе подобластей $\rho_{L_n/2,k}$ уже хранится на каждом процессоре.

2. Пользуясь алгоритмом 1 и алгоритмом 2, вычисляем одновременно на каждом процессоре:

   (a) плотность экранирующих зарядов ρ^{scr} на общей границе подобластей, расположенной на слое под номером $L_n/2$;

   (b) значения потенциала Φ^{scr}, создаваемого экранирующими зарядами и плотностью $\rho_{L_n/2,k}$: $\Phi^{\text{scr}}_{1,k}$ и $\Phi^{\text{scr}}_{L_n-1,k}$;

   (c) процессор P_{j_1} вычисляет значения потенциала на слоях i_1 и i_2: $\Phi^{\text{scr}}_{i1,k}$ и $\Phi^{\text{scr}}_{i2,k}$;

   (d) процессор P_{j_2} вычисляет значения потенциала на слоях i_3 и i_4: $\Phi^{\text{scr}}_{i3,k}$ и $\Phi^{\text{scr}}_{i4,k}$.

3. Определим величины $\Phi^{l+1}_{1,k} := \Phi^l_{1,k} + \Phi^{\text{scr}}_{1,k}$ и $\Phi^{l+1}_{L_n-1,k} := \Phi^l_{L_n-1,k} + \Phi^{\text{scr}}_{L_n-1,k}$.

Перейдем теперь к описанию основного алгоритма для решения уравнения Пуассона с N процессорами, работающего на определенной выше древовидной структуре из N подобластей и предназначенного для решения исходной задачи (2).

Алгоритм 4. Декомпозиция области на N подобластей с прямым методом сопряжения.

1. Начинаем с листьев дерева (уровень дерева $l = 0$).

   (a) В каждой подобласти $\Omega^0_n,\ n = 0, \ldots, N - 1$, решаем задачу Дирихле для уравнения Пуассона с нулевыми граничными условиями.

   (b) Определяем коммуникационную пару процессоров (эти процессоры обмениваются между собой данными на следующем уровне дерева): для $l = 1$ — это процессоры с индексами P_{2m}, P_{2m+1}, $m = 0, N/2 - 1$.

(c) Пусть $\Phi_n\big|_{\Gamma_{\text{left}}}$, $\Phi_n\big|_{\Gamma_{\text{right}}}$ — значения потенциала на левой и правой границах подобласти Ω_n^0. Каждый процессор P_n будет хранить эти значения и накапливать в них значения потенциала, создаваемого зарядами, расположенными в других подобластях. В соответствии с поставленной задачей Дирихле, на первом шаге $\Phi_n\big|_{\Gamma_{\text{left}}}$ и $\Phi_n\big|_{\Gamma_{\text{right}}}$ устанавливаются равными нулю.

2. Применяем алгоритм 3 для каждой пары подобластей Ω_{2m}^l и Ω_{2m+1}^l, $m = 0, \dfrac{N}{2^{l+1}-1}$, с заданного уровня и пары процессоров, составляющих ранее определенную коммуникационную пару.

   (a) В качестве специально заданных слоев i_1 и i_2 для процессора P_n выбираем левую и правую границу соответствующей ему подобласти-листа. Вычисляем $\Phi_n^{l+1}\big|_{\Gamma_{\text{left}}}$ и $\Phi_n^{l+1}\big|_{\Gamma_{\text{right}}}$.

   (b) Определяем $\Phi_n\big|_{\Gamma_{\text{left}}} := \Phi_n\big|_{\Gamma_{\text{left}}} + \Phi_n^{l+1}\big|_{\Gamma_{\text{left}}}$ и $\Phi_n\big|_{\Gamma_{\text{right}}} := \Phi_n\big|_{\Gamma_{\text{right}}} + \Phi_n^{l+1}\big|_{\Gamma_{\text{right}}}$.

   (c) В итоге оба процессора P_m и P_{m+1} хранят всю необходимую информацию для дальнейшей обработки подобласти Ω_m^{l+1}. Для шага l число процессоров, обрабатывающих подобласть Ω_m^l, будет равно $2^{l+1} - 1$.

3. Определяем новую коммуникационную пару процессоров для уровня $l + 1$ следующим образом. Поскольку число процессоров, способных обрабатывать подобласть Ω_{2m}^{l+1}, равно числу процессоров, способных обрабатывать подобласть Ω_{2m+1}^{l+1}, то устанавливаем между ними взаимно однозначное соответствие. В результате будет выполнено условие, что межпроцессорное взаимодействие на каждом уровне выполняется только между парами процессоров.

4. Если достигнут корень дерева, то переходим на следующий шаг. Иначе $l := l + 1$ и переходим на шаг 2.

5. В каждой подобласти Ω_n^0 решаем задачу Дирихле для уравнения Лапласа с граничными условиями $\Phi_n\big|_{\Gamma_{\text{left}}}$ и $\Phi_n\big|_{\Gamma_{\text{right}}}$.

6. Решение исходной задачи получаем в виде суммы решений, полученных на шагах 1a и 5.

Из определения алгоритма видно, что каждому процессору необходимо один раз решить уравнение Пуассона и Лапласа, а также сделать $N_l = \log_2 N$ шагов, на каждом шаге выполняя межпроцессорные коммуникации с одним и только одним процессором (передачу и получение массива данных размерами $3L_y$) и применяя четыре раза алгоритм 1. Таким образом, если для решения уравнения Пуассона и Лапласа применять двукратное быстрое преобразование Фурье, то итоговая вычислительная сложность алгоритма 4 составит

$$T_{\text{calc}}(L_x, L_y, N) = O\big(L_x L_y (\log(L_x/N + \log L_y)/N\big) + O\left(L_y \log L_y \log N\right). \tag{8}$$

Коммуникационная сложность для каждого процессора (тот объем данных, который необходимо отправить и получить каждому процессору) составляет $T_{\text{comm}}(L_x, L_y, N) = \big(\text{Send}(3L_y) + \text{Recv}(3L_y)\big) \log_2 N$. Таким образом, если $N \ll L_x$ (например, $N \leqslant 32\, L_x$), то объем коммуникаций относительно невелик в сравнении с объемом вычислительной задачи, решаемой на каждом процессоре. Вместе с тем, если сопоставлять производительность параллельного алгоритма 4 с производительностью последовательной версии, то накладные расходы будут заключаться не только в появлении коммуникаций, но и в том, что задача Дирихле решается дважды: один раз для уравнения Лапласа и второй раз для уравнения Пуассона.

Заметим, что алгоритм 4 может быть применен также и к решению задачи Дирихле для экранированного уравнения Пуассона, которая записывается в виде

$$\Delta\Phi(\boldsymbol{x}) - a^2\Phi(\boldsymbol{x}) = \rho(\boldsymbol{x}), \quad \Phi(\boldsymbol{x})\big|_\Gamma = 0.$$

Это уравнение важно с точки зрения применения метода к декомпозиции области в трехмерном случае. Действительно, используя метод разделения переменных для решения уравнения Пуассона, мы получим независимые линейные системы, представляющие собой разностные аналоги экранированного уравнения Пуассона для Фурье-гармоник потенциала. Каждая из таких систем может решаться параллельно с помощью алгоритма 4. Вместе с тем, эффективная реализация декомпозиции трехмерной области по двум направлениям не может быть получена как тривиальное обобщение двумерной и является предметом дальнейших исследований.

3. Тестовые эксперименты по оценке производительности. Метод декомпозиции с прямым сопряжением подобластей (DDCS) дает то же самое решение линейной системы (2), что и другие прямые методы (отличаясь лишь в последних знаках мантиссы вещественного числа с двойной точностью за счет изменения порядка арифметических операций). В связи с этим мы не приводим здесь результаты тестов по проверке работоспособности метода (такие тесты для разнообразного набора функций, сеточных параметров и количества процессоров проводились на этапе реализации алгоритма и продемонстрировали корректность программной реализации).

Тестовые эксперименты по оценке производительности программной реализации алгоритма выполнялись на суперкомпьютерах МВС-100К в Межведомственном суперкомпьютерном центре РАН, Москва (МСКЦ, www.jscc.ru) и НКС-30Т G6 в Сибирском суперкомпьютерном центре, Новосибирск (ССКЦ, www2.sscc.ru). Для выполнения быстрого преобразования Фурье использовалась одна и та же версия библиотеки FFTW 3.1.4 [25], установленная на обоих суперкомпьютерах. Количество процессоров варьировалось от 4 до 512.

При разработке параллельных алгоритмов для высокопроизводительных вычислений принято разделять два типа масштабируемости. Под сильной масштабируемостью (*strong scaling*) понимается то, как время решения изменяется при увеличении числа процессоров и сохранении общего размера задачи. Под слабой масштабируемостью (*weak scaling*) — то, как время решения изменяется при увеличении числа процессоров и сохранении размера задачи, решаемой одним процессором (т.е. в этом случае предполагается, что общий размер задачи растет пропорционально числу процессоров).

С точки зрения реальных численных экспериментов это означает, что сильная масштабируемость параллельного алгоритма важна тогда, когда задача требуемого размера помещается в оперативную память некоторого количества процессоров, но при этом ее решение занимает неприемлемо большое время. Свойство слабой масштабируемости алгоритма важно тогда, когда задача некоторого размера решается за приемлемое время и требуется решать задачи большего размера, сохраняя время счета.

При замерах производительности общее время выполнения алгоритма 4 разбивалось на три логических части: T_{calc} (суммарное время решения уравнения Пуассона и Лапласа), T_{comm} (межпроцессорные коммуникации: передача массивов данных между процессорами MPI процедурой MPI_SendRecv), T_{prop} (вычисление потенциала выделенного слоя и граничных условий; соответствует второму члену в формуле (8).

Кроме того, делались замеры времени предвычисления всех значений, необходимых для применения алгоритма 1 (T_{init}).

Поскольку производительность алгоритма на обоих суперкомпьютерах (МСКЦ и ССКЦ) отличалась незначительно, мы приводим лишь те результаты, которые получены на ССКЦ. Однако тестовый эксперимент для сетки размером 65536×65536 показал, что коммуникационные расходы при оперировании с данными большого размера на большом количестве процессоров в МСКЦ заметно выше, чем в ССКЦ. Такое отличие, по-видимому, связано с разными версиями MPI-библиотек (в МСКЦ использовалась более старая некоммерческая MVAPICH-1.2, а в ССКЦ относительно новая коммерческая Intel MPI 4.1) и различиями в аппаратной части, отвечающей за межпроцессорные коммуникации. Результаты для этого эксперимента приведены для обоих суперкомпьютеров.

При оценке слабой масштабируемости есть два варианта увеличения размера двумерной задачи: по одному направлению x (по этому же направлению область разбивается на подобласти) и по двум направлениям x, y. Поскольку оба таких варианта имеют практическую значимость, были проведены эксперименты, соответствующие каждому из вариантов. В табл. 1 представлены результаты тестовых экспериментов (суперкомпьютер НКС-Т30 G6, ССКЦ) по оценке слабой масштабируемости при увеличении размеров расчетной области и сетки по одному направлению. Видно, что расходы на коммуникации с увеличением процессоров растут слабо, а расходы на вычислительную часть остаются одинаковыми.

В табл. 2 представлены результаты экспериментов (суперкомпьютер НКС-Т30 G6, ССКЦ) по оценке слабой масштабируемости при увеличении размера области по двум направлениям. В этом случае объем пересылок увеличивается уже не только из-за увеличения числа процессоров и добавления новых уровней при конструировании дерева подобластей, но и из-за увеличения числа узлов L_y. Кроме того, объем вычислений T_{prop} тоже растет в точном соответствии с формулой (8).

В табл. 3 и 4 приведены результаты по оценке сильной масштабируемости для задач размерами 16384×16384 и 65536×65536 и различного количества процессоров. Кроме того, в табл. 4 указаны сравнительные результаты для суперкомпьютеров ССКЦ и МСКЦ.

4. Заключение. Разработан и реализован параллельный алгоритм для решения двумерного уравнения Пуассона в контексте нестационарных задач. Основным его преимуществом является относительно

Таблица 1

Время решения (секунды) для сетки N 2048×2048

N	$L_x \times L_y$	T_{init}	T_{all}	T_{calc}	T_{comm}	T_{prop}
4	8192×2048	1.28	1.02	0.98	0.04	0.0004
8	16384×2048	3.13	0.98	0.97	0.01	0.0008
16	32768×2048	7.31	0.99	0.98	0.01	0.0011
32	65536×2048	15.50	0.98	0.97	0.01	0.0014
64	131072×2048	35.02	0.99	0.97	0.02	0.0017
128	262144×2048	82.22	1.07	0.98	0.09	0.0020
256	524288×2048	190.94	1.08	0.98	0.10	0.0023
512	1048576×2048	414.01	1.07	0.97	0.09	0.0025

Таблица 2

Время решения (секунды) для сетки N $2048 \times \sqrt{N}\,2048$

Np	$L_x \times L_y$	T_{init}	T_{all}	T_{calc}	T_{comm}	T_{prop}
4	4096×4096	1.27	1.01	1.00	0.01	0.001
16	8192×8192	7.32	1.03	1.00	0.02	0.005
64	16384×16384	30.61	1.11	1.07	0.02	0.019
256	32768×32768	124.37	1.37	1.15	0.16	0.062

Таблица 3

Время решения (секунды) для сетки 16384×16384

N	Узлов сетки на проц.	T_{init}	T_{all}	T_{calc}	T_{comm}	T_{prop}
16	16 777 216	29.25	5.74	5.66	0.07	0.01
32	8 388 608	28.42	2.32	2.20	0.10	0.02
64	4 194 304	30.61	1.11	1.07	0.02	0.02
128	2 097 152	30.16	0.62	0.53	0.07	0.02
256	1 048 576	31.11	0.34	0.26	0.05	0.03
512	524 288	30.81	0.20	0.13	0.04	0.03

Таблица 4

Время решения (секунды) для сетки 65536×65536

N	Узлов сетки на процессор	T_{init}	T_{all}	T_{calc}	T_{comm}	T_{prop}
256 (МСКЦ)	16 777 216	537.88	7.54	6.73	0.52	0.28
512 (МСКЦ)	8 388 608	535.73	4.20	3.33	0.55	0.32
256 (ССКЦ)	16 777 216	505.01	6.79	6.29	0.35	0.15
512 (ССКЦ)	8 388 608	499.62	3.10	2.63	0.30	0.17

небольшой объем межпроцессорных коммуникаций, а накладные расходы состоят в начальном предвычислении вспомогательных величин в фурье-разложении потенциала и двукратном решении задачи Дирихле для уравнения Лапласа и Пуассона. Показано, что алгоритм хорошо масштабируется в слабом и сильном смысле по крайней мере до 512 процессоров и сеток вплоть до 4 миллиардов узлов.

В дальнейшем этот алгоритм может быть адаптирован для решения трехмерного уравнения Пуассона на очень больших сетках, вплоть до 10^{12} узлов с использованием десятков тысяч процессоров.

Работа выполнена при поддержке РФФИ (проекты 14–01–31088, 14–07–00241) и Интеграционного проекта СО РАН № 130.

СПИСОК ЛИТЕРАТУРЫ

1. *Снытников В.Н., Вшивков В.А., Кукшева Е.А., Неупокоев Е.С., Никитин С.А., Снытников А.В.* Трехмерное численное моделирование нестационарной гравитирующей системы многих тел с газом // Письма в астрономический журнал. 2004. **30**, № 2. 146–160.
2. *Кинг А.Р.* Введение в классическую звездную динамику. М.: Едиториал УРСС, 2002.
3. *Березин Ю.А., Вшивков В.А.* Метод частиц в динамике разреженной плазмы. Новосибирск: Наука, 1980.
4. *Hockney R.W., Eastwood J.W.* Computer simulation using particles. New York: McGraw-Hill, 1981.
5. *Hockney R.W.* Gravitational experiments with a cylindrical galaxy // Astrophysical Journal. 1967. **150**. 797–806.
6. *Вшивков В.А., Снытников В.Н., Снытников Н.В.* Моделирование трехмерной динамики вещества в гравитационном поле на многопроцессорных ЭВМ // Вычислительные технологии. 2006. **11**, № 2. 15–27.
7. *James R.A.* The solution of Poisson's equation for isolated source distributions // Journal of Computational Physics. 1977. **25**, N 2. 71–93.
8. *Вшивков В.А., Снытников А.В.* Построение эффективного параллельного метода решения уравнения Пуассона для моделирования эволюции протопланетного диска // Вычислительные методы и программирование. 2009. **10**. 116–122.
9. *Лаевский Ю.М., Мацокин А.М.* Методы декомпозиции решения эллиптических и параболических краевых задач // Сиб. журн. вычисл. матем. 1999. **2**, № 4. 361–372.
10. *Ильин В.П.* Методы конечных разностей и конечных объемов для эллиптических уравнений. Новосибирск: Институт математики РАН им. С.Л. Соболева, 2000.
11. *Anderson C.R.* Domain decomposition techniques and the solution of Poisson's equation in infinite domains // Domain Decomposition Methods. Philadelphia: SIAM, 1989. 129–139.
12. *Balls G.T., Colella P.* A finite difference domain decomposition method using local corrections for the solution of Poisson's equation // J. Comp. Physics. 2002. **180**, N 1. 25–53.
13. *McCorquodale P., Colella P., Balls G.T., Baden S.B.* A local corrections algorithm for solving Poisson's equation in three dimensions // Comm. App. Math. And Comp. Sci. 2007. **2**, N 1. 57–81.
14. *Huang J., Greengard L.* A fast direct solver for elliptic partial differential equations on adaptively refined meshes // SIAM J. Sci. Comput. 1999. **21**, N 4. 1551–1566.
15. *Ricker P.M.* A direct multigrid Poisson solver for oct-tree adaptive meshes // The Astrophysical Journal Supplement Series. 2008. **176**, N 1. 293–300.
16. *Terekhov A.V.* Parallel dichotomy algorithm for solving tridiagonal system of linear equations with multiple right-hand sides // Parallel Computing. 2010. **36**, N 8. 423–438.
17. *Terekhov A.V.* A fast parallel algorithm for solving block-tridiagonal systems of linear equations including the domain decomposition method // Parallel Computing. 2013. **39**, N 6/7. 245–258.
18. *Яненко Н.Н., Коновалов А.Н., Бугров А.Н., Шустов Г.В.* Об организации параллельных вычислений и "распараллеливании" прогонки // Численные методы механики сплошной среды. Том 9, вып. 7. Новосибирск: Институт теоретической и прикладной механики РАН, 1978. 139–146.
19. *Ayala O., Wang L.-P.* Parallel implementation and scalability analysis of 3D Fast Fourier Transform using 2D domain decomposition // Parallel Computing. 2013. **39**, N 1. 58–77.
20. *Duy T.V.T., Ozaki T.* A decomposition method with minimum communication amount for parallelization of multi-dimensional FFTs // Computer Physics Communications. 2014. **185**, N 1. 153–164.
21. *Springel V., Yoshida N., White S.D.M.* GADGET: a code for collisionless and gasdynamical cosmological simulation // New Astronomy. 2001. **6**, N 2. 79–117.
22. *Dubey A., Antypas K., Ganapathy M.K., et al.* Extensible component-based architecture for FLASH, a massively parallel, multiphysics simulation code // Parallel Computing. 2009. **35**, N 10/11. 512–522.
23. IntelRMath Kernel Library 10.0 — Overview (https://software.intel.com/en-us/intel-mkl).
24. *Самарский А.А., Николаев Е.С.* Методы решений сеточных уравнений. М.: Наука, 1978.
25. *Frigo M., Johnson S.G.* FFTW software (http://www.fftw.org).

Поступила в редакцию
09.12.2014

A Parallel Algorithm for Solving 2D Poisson's Equation in the Context of Nonstationary Problems

N. V. Snytnikov [1]

[1] *Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch of Russian Academy of Sciences; prospekt Lavrentyeva 6, Novosibirsk, 630090, Russia; Ph.D., Scientist, e-mail: nik@ssd.sscc.ru*

Received December 9, 2014

Abstract: A new parallel method to solve the Dirichlet problem for Poisson's equation in the context of nonstationary problems of mathematical physics is proposed. This method is based on a decomposition of a rectangular Cartesian domain in one direction, on a direct method of solving Poisson's equation in each subdomain, and on the coupling of the subdomains using a fast procedure for evaluating a single layer potential. A number of test experiments conducted on supercomputers installed at Joint Supercomputing Center of Russian Academy of Sciences and at Siberian Supercomputing Center show a good weak and strong scalability of the parallel algorithm.

Keywords: Poisson's equation, Dirichlet problem, domain decomposition, gravitational potential, stellar dynamics, parallel programming, scalability of algorithms.

References

1. V. N. Snytnikov, V. A. Vshivkov, E. A. Kuksheva, et al., "Three-Dimensional Numerical Simulation of a Nonstationary Gravitating N-Body System with Gas," Pis'ma v Astron. Zh. **30** (2), 146–160 (2004) [Astron. Lett. **30** (2), 124–137 (2004)].

2. I. R. King, *An Introduction to Classical Stellar Dynamics* (Univ. California, Berkeley, 1994; Editorial URSS, Moscow, 2002).

3. Yu. A. Berezin and V. A. Vshivkov, *The Particle Method in the Dynamics of a Rarefied Plasma* (Nauka, Novosibirsk, 1980) [in Russian].

4. R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles* (McGraw-Hill, New York, 1981).

5. R. W. Hockney, "Gravitational Experiments with a Cylindrical Galaxy," Astrophys. J. **150**, 797–806 (1967).

6. V. A. Vshivkov, V. N. Snytnikov, and N. V. Snytnikov, "Simulation of the Three-Dimensional Dynamics of Matter in the Gravitational Field Using Multiprocessor Computers," Vychisl. Tekhnol. **11** (2), 15–27 (2006).

7. R. A. James, "The Solution of Poisson's Equation for Isolated Source Distributions," J. Comput. Phys. **25** (2), 71–93 (1977).

8. V. A. Vshivkov and A. V. Snytnikov, "Development of an Efficient Parallel Poisson Equation Solver for the Simulation of Protoplanetary Disk Evolution," Vychisl. Metody Programm. **10**, 116–122 (2009).

9. Yu. M. Laevsky and A. M. Matsokin, "Decomposition Methods for the Solution to Elliptic and Parabolic Boundary Value Problems," Sib. Zh. Vychisl. Math. **2** (4), 361–372 (1999).

10. V. P. Il'in, *Finite Difference and Finite Volume Methods for Elliptic Equations* (Sobolev Inst. Math., Novosibirsk, 2000) [in Russian].

11. C. R. Anderson, "Domain Decomposition Techniques and the Solution of Poisson's Equation in Infinite Domains," in *Domain Decomposition Methods* (SIAM Press, Philadelphia, 1989), pp. 129–139.

12. G. T. Balls and P. Colella, "A Finite Difference Domain Decomposition Method Using Local Corrections for the Solution of Poisson's Equation," J. Comput. Phys. **180** (1), 25–53 (2002).

13. P. McCorquodale, P. Colella, G. T. Balls, and S. B. Baden, "A Local Corrections Algorithm for Solving Poisson's Equation in Three Dimensions," Commun. Appl. Math. Comput. Sci. **2** (1), 57–81 (2007).

14. J. Huang and L. Greengard, "A Fast Direct Solver for Elliptic Partial Differential Equations on Adaptively Refined Meshes," SIAM J. Sci. Comput. **21** (4), 1551–1566 (1999).

15. P. M. Ricker, "A Direct Multigrid Poisson Solver for Oct-Tree Adaptive Meshes," Astrophys. J. Suppl. Ser. **176** (1), 293–300 (2008).

16. A. V. Terekhov, "Parallel Dichotomy Algorithm for Solving Tridiagonal System of Linear Equations with Multiple Right-Hand Sides," Parallel Comput. **36** (8), 423–438 (2010).

17. A. V. Terekhov, "A Fast Parallel Algorithm for Solving Block-Tridiagonal Systems of Linear Equations Including the Domain Decomposition Method," Parallel Comput. **39** (6/7), 245–258 (2013).

18. N. N. Yanenko, A. N. Konovalov, A. N. Bugrov, and G. V. Shustov, "On Organization of Parallel Computations and Parallelization of the Tridiagonal Matrix Algorithm," in *Numerical Methods of Continuum Mechanics* (Inst. Theor. Appl. Mech., Novosibirsk, 1978), Vol. 9, Issue 7, pp. 139–146.

19. O. Ayala and L.-P. Wang, "Parallel Implementation and Scalability Analysis of 3D Fast Fourier Transform Using 2D Domain Decomposition," Parallel Comput. **39** (1), 58–77 (2013).

20. T. V. T. Duy and T. Ozaki, "A Decomposition Method with Minimum Communication Amount for Parallelization of Multi-Dimensional FFTs," Comput. Phys. Commun. **185** (1), 153–164 (2014).

21. V. Springel, N. Yoshida, and S. D. M. White, "GADGET: A Code for Collisionless and Gasdynamical Cosmological Simulation," New Astr. **6** (2), 79–117 (2001).

22. A. Dubey, K. Antypas, M. K. Ganapathy, et al., "Extensible Component-Based Architecture for FLASH, a Massively Parallel, Multiphysics Simulation Code," Parallel Comput. **35** (10/11), 512–522 (2009).

23. IntelRMath Kernel Library 10.0: Overview. https://software.intel.com/en-us/intel-mkl. Cited January 8, 2015.

24. A. A. Samarskii and E. S. Nikolaev, *Numerical Methods for Grid Equations* (Nauka, Moscow, 1978; Birkhäuser, Basel, 1989).

25. M. Frigo and S. G. Johnson, "FFTW software," http://www.fftw.org. Cited January 8, 2015.