УДК 519.63
MSC 65Y05

МЕТОДЫ ПАРАЛЛЕЛЬНОГО РЕШЕНИЯ СЛАУ НА СИСТЕМАХ С РАСПРЕДЕЛЕННОЙ ПАМЯТЬЮ В БИБЛИОТЕКЕ KRYLOV[1]

Д.С. Бутюгин, В.П. Ильин, Д.В. Перевозкин

PARALLEL METHODS FOR SLAE SOLUTION ON THE SYSTEMS WITH DISTRIBUTED MEMORY IN KRYLOV LIBRARY

D.S. Butyugin, V.P. Il'in, D.V. Perevozkin

В работе рассматривается подход к созданию итерационного black-box («черного ящика») параллельного решателя, использованный в библиотеке Krylov для систем линейных алгебраических уравнений (СЛАУ) с разреженными матрицами высокого порядка, возникающими при сеточных аппроксимациях многомерных краевых задач и представленными в сжатом строчном формате CSR. Предлагается вариант алгебраической одномерной декомпозиции СЛАУ. Алгоритм основан на обходе в ширину графа матрицы системы и позволяет привести ее к блочно-трехдиагональному виду. За основу алгебраического решателя системы взят аддитивный метод Шварца, который естественным образом ложится на архитектуру вычислительных систем с распределенной памятью. Полученные алгебраические системы в подпространстве следов, образованных переменными на внутренних границах подобластей, решаются с помощью обобщенного метода минимальных невязок. Вспомогательные системы в подобластях решаются с помощью прямого алгоритма PARDISO из библиотеки Intel MKL, использующего распараллеливание над общей памятью средствами OpenMP. Реализованные алгоритмы апробированы на численном решении ряда задач вычислительной математики, таких как задачи гидродинамики, диффузионно-конвективные уравнения, задачи электромагнетизма и др. Приведенные результаты численных экспериментов демонстрируют эффективность предлагаемых решений для многопроцессорных вычислительных систем с распределенной памятью.

Ключевые слова: итерационные алгоритмы, методы декомпозиции областей, распараллеливание, алгебраические системы, разреженные матрицы, численные эксперименты, аддитивный метод Шварца

The paper presents an approach to creation of black-box parallel iterative solver, which is used in Krylov library for solving systems of linear algebraic equations (SLAEs) with large sparse matrices in CSR format arising from discretization of multidimensional boundary value problems. A variant of one-dimensional algebraic decomposition method is offered. The algorithm is based on breadth-first search of SLAE's adjacency graph that allows to reduce the matrix to block-tridiagonal form. The algebraic solver is based on additive Schwarz method which naturally suits distributed memory computer systems. The generalized minimal residual method is used to solve the SLAEs arising from relations on subdomains' boundaries. Auxiliary subdomain systems are solved with Intel MKL's multithreaded direct solver PARDISO. Implemented algorithms were tested on the numerical solution of the series of computational mathematics problems, such as problems of hydrodynamics, diffusion-convection equations, problems of electromagnetism and others. Adduced numerical experiments results show the effectiveness of the presented algorithms for multiprocessor computational systems with distributed memory.

Keywords: iterative algorithms, domain decomposition methods, parallel computing, algebraic systems, sparse matrices, numerical experiments, additive Schwarz method

1. Введение

Системы линейных алгебраических уравнений с разреженной матрицей возникают при численном решении большого числа задач вычислительной математики. Вычислительная сложность таких задач часто требует кластерных расчетов. В настоящее время существует ряд подходов к декомпозиции задач и, соответственно, матриц систем, а также итерационных алгоритмов, позволяющих свести решение исходной задачи к серии решений задач в подобластях. К методам декомпозиции задачи можно отнести геометрические методы декомпозиции расчетной области и последующую аппроксимацию задачи в каждой из подобластей, а также алгебраические методы декомпозиции матрицы системы, например, реализованные в библиотеке METIS [1].

В рамках данной работы предлагается вариант алгебраической одномерной декомпозиции СЛАУ. Алгоритм основан на обходе в ширину графа матрицы системы и позволяет привести ее к блочно-трехдиагональному виду. За основу алгебраического решателя системы взят аддитивный метод Шварца, который естественным образом ложится на архитектуру вычислительных систем с распределенной памятью. К данному методу применяется крыловское ускорение [2], заключающееся в замене стационарного процесса $x_{n+1} = Tx_n + g$ решением системы $(I - T)x = g$, которое осуществляется при помощи итерационных методов в подпространствах Крылова, таких как GMRES. При этом итерации осуществляются в подпространстве следов, образованных переменными на границах подобластей. Решение систем в подобластях осуществляется параллельным прямым решателем разреженных систем PARDISO из библиотеки Intel® MKL [3]. Использование такого подхода позволяет реализовать black-box решатель, не требующий дополнительной информации для решения СЛАУ и имеющий небольшое число конфигурационных параметров. Реализация алгоритма на языке C++ с использованием средств MPI и высокоуровневой библиотеки линейной алгебры Eigen [4] включена в библиотеку Krylov [5].

Тестирование алгоритма проводилось на большом числе методических и практических задач вычислительной физики. Было произведено исследование быстродействия решателя

[1]Статья рекомендована к публикации программным комитетом международной научной конференции «Параллельные вычислительные технологии 2012»

и числа итераций в зависимости от различных параметрах решателя. К варьируемым параметрам относились параметры алгебраической декомпозиции, такие как перекрытие подобластей и их количество, а к параметрам времени исполнения — число потоков в PARDISO при решении задач в подобластях, а также число MPI процессов на узел. Кроме того, была продемонстрирована точность получаемых решений для представленных задач.

2. Алгоритмы решения СЛАУ

2.1. Метод алгебраической декомпозиции СЛАУ

Аппроксимация различных краевых задач соответствующих дифференциальных уравнений различными методами, такими как методы конечных разностей, конечных объемов и конечных элементов, приводит к системе линейных алгебраических уравнений вида

$$Ax = b, \quad x, b \in \mathcal{R}^N, \quad A \in \mathcal{R}^{N,N}, \quad \det|A| \neq 0. \tag{1}$$

Решение СЛАУ на кластере требует предварительной ее декомпозиции и распределения блоков матрицы, а также векторов неизвестных и правой части между узлами кластера. В работе предлагается использовать следующую модификацию метода одно-направленной алгебраической декомпозиции задачи [2], основными достоинствами которой являются простота, высокая скорость работы алгоритма и возможность получить распределенную блочно-трехдиагональную матрицу на выходе.

Сначала ищется псевдо-периферийная вершина v графа G, ассоциированного с матрицей системы (1).

Алгоритм 1. Поиск псевдо-периферийной вершины v графа G. Запускается обход в ширину [6] с произвольной вершины графа матрицы и находятся наиболее удаленные от нее вершины и расстояние до них. Далее снова запускается обход в ширину, начиная уже с произвольной вершины среди множества наиболее удаленных вершин, полученных на предыдущем шаге. Обходы в ширину повторяются до тех пор, пока увеличивается максимальное расстояние от стартовой вершины до остальных. Как только расстояние перестанет увеличиваться, алгоритм останавливается и в качестве псевдо-периферийной вершины v возвращает вершину, с которой начинался последний обход в ширину.

Время работы алгоритма 1 равно $O(E \cdot K)$, где E — количество ребер в графе матрицы. Для большинства конечных схем аппроксимации уравнений число ребер пропорционально количеству вершин сетки в расчетной области. Этот факт следует из того, что соответствующие конечные базисы финитны и, как правило, используются сетки, у которых имеются ограничения на минимальные величины плоских и телесных внутренних углов. Число K в оценке равно количеству итераций, совершенных алгоритмом 1. Для произвольных графов в худшем случае это число может быть довольно велико, однако для графов практических задач алгоритму обычно требуется всего лишь несколько итераций [2]. За счет этого на практике алгоритм поиска псевдо-периферийных вершин оказывается весьма эффективным. Более того, во многих случаях оказывается достаточно выбрать произвольную, например, первую вершину графа, и вообще не запускать алгоритм поиска псевдо-периферийных вершин.

После того, как выбрана некоторая вершина v, начиная с нее запускается обход графа G в ширину. Все вершины разделяются на множества — «фронты»— согласно расстоянию от них до вершины v. То есть, во фронт F_k ($k = 0 \ldots d$, где d — псевдо-диаметр графа G) попадают все вершины, равноудаленные от вершины v на расстояние k.

Далее рассмотрим способ объединения фронтов в алгебраические подобласти Ω_p. Мы

будем объединять только последовательные фронты в подобласти, так что

$$\Omega_p = \bigcup_{k=l_p}^{r_p} F_k,$$

где l_p и r_p — границы подобласти Ω_p. Пусть требуется построить P подобластей (например, по числу используемых узлов кластера) для системы (1) с N неизвестными. Желательно разбить неизвестные в подобласти приблизительно по N/P неизвестных. Однако поскольку мы объединяем в подобласти не отдельные вершины, а фронты, то точное разбиение неизвестных по N/P в каждой подобласти может не получиться. Тем не менее, мы можем при помощи бинарного поиска минимизировать максимальный размер подобласти либо максимизировать минимальный размер.

Рассмотрим, например, максимизацию минимального размера, которая может использоваться для того, чтобы, по возможности, более равномерно загрузить все узлы кластера. Легко видеть, что функция зависимости максимально возможного числа подобластей при фиксированном минимально допустимом размере монотонна. Действительно, при фиксированном минимальном размере разделение вершин графа разбиение на подобласти можно произвести при помощи следующего жадного [6] алгоритма.

Алгоритм 2. Алгоритм построения максимального числа подобластей. К первой подобласти добавляем последовательно фронты, начиная с первого, пока размер подобласти не станет больше либо равным минимально допустимому размеру. В этом момент можно прекратить увеличение первой подобласти и приступить к построению второй подобласти. Этот процесс продолжается до тех пор, пока все фронты не окажутся в какой-либо подобласти. При этом может оказаться так, что в конце окажется несколько фронтов, которых недостаточно для формирования отдельной подобласти. Такие фронты можно отнести к последней построенной подобласти.

Ясно, что приведенный выше способ строит максимальное число подобластей. Действительно, если бы можно было построить больше подобластей при заданном минимальном размере подобласти, то тогда, начиная с некоторого i, подобласть Ω_i в «жадном»разбиении и подобласть Ω'_i отличались бы. Рассмотрим минимальное такое i. Если в подобласти Ω'_i больше фронтов, чем в Ω_i, то тогда оптимальное разбиение можно перестроить таким образом, чтобы в подобласти i было столько же фронтов, что и в оптимальном. Значит, в подобласти Ω'_i должно быть меньше фронтов, чем в Ω_i. Однако Ω_i строилось таким образом, что в него поместили минимально возможное число фронтов при фиксированном начальном. Получаем противоречие, и значит, построенное алгоритмом разбиение является максимальным.

Из приведенных выше рассуждений становится ясно, что с увеличением минимального размера число подобластей не может увеличиться. Значит, функция зависимости их числа от минимального размера действительно монотонна, и можно использовать двоичный поиск максимального минимального размера подобласти, чтобы получить P подобластей. Кроме того, описанный выше жадный алгоритм позволяет собственно построить эти подобласти уже после определения минимального размера.

Представленный алгоритм генерирует подобласти без пересечений. Однако получить разбиение с пересечениями довольно легко — достаточно просто расширить каждую из подобластей на несколько фронтов слева и справа. Также можно при начале генерации следующей подобласти отступить от границы предыдущей внутрь необходимое число фронтов.

Заметим, что ребра в графе матрицы могут соединять вершины либо одного, либо двух соседних фронтов. Тогда получается, что граничными для подобласти p будут вершины, принадлежащие двум фронтам F_{l_p-1} и F_{r_p+1}. Можно считать, что эти фронты принадле-

жат соседним подобластям Ω_{p-1} и Ω_{p+1} соответственно. Получается, что если последовательно занумеровать переменные в каждой из подобластей, а в каждой из подобластей — последовательно в каждом из фронтов, мы получим следующую блочно-трехдиагональную СЛАУ:

$$\bar{A}\bar{x} = \begin{bmatrix} D_1 & U_1 & & \\ L_1 & D_2 & \ddots & \\ & \ddots & \ddots & U_{P-1} \\ & & L_{P-1} & D_P \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_P \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_P \end{bmatrix} = \bar{f}. \tag{2}$$

Будем полагать, что диагональные блоки D_p, соответствующие СЛАУ в подобластях Ω_p, не вырождены. При этом если разбиение исходной СЛАУ выполнялось с пересечениями, то в полученной СЛАУ (2) порядка N' окажется по несколько переменных, соответствующих одной и той же вершине графа, но лежащим в разных подобластях; в этом случае $N' > N$.

2.2. Крыловское ускорение аддитивного метода Шварца

После декомпозиции задачи на подобласти и пересылки блоков матрицы в соответствующие узлы кластера мы имеем распределенную блочную СЛАУ (2). Для решения таких СЛАУ широко используются методы Шварца. Одна итерация аддитивного метода Шварца в матричном виде может быть записана как (см. [2])

$$\bar{x}^{i+1} = \bar{x}^i + \sum_p R_p^T D_p^{-1} R_p (f - \bar{A}\bar{x}^i),$$

где $R_p : \mathcal{R}^{N'} \to \mathcal{R}^{N_p}$ — оператор сужения на подобласть Ω_p. Здесь N' — размерность СЛАУ (2), а N_p — количество неизвестных в подобласти Ω_p. С использованием блочного вида СЛАУ итерацию метода Шварца можно переписать следующим образом:

$$x_p^{i+1} = x_p^i + D_p^{-1}(f_p - L_p x_{p-1}^i - U_p x_{p+1}^i),$$

где для $p = 1$ и $p = P$ полагаем, что $L_p x_{p-1}^i = 0$ и $U_p x_{p+1}^i = 0$ соответственно. В общем случае члены $L_p x_{p-1}^i$ и $U_p x_{p+1}^i$ являются аналогами условий Дирихле для краевых задач.

Заметим, что на самом деле итерировать достаточно только переменные, образующие внешние границы подобластей, поскольку, зная граничные значения, легко определить значения внутри подобластей, решив соответствующую подзадачу. Для этого в произведениях $L_p x_{p-1}^i$ и $U_p x_{p+1}^i$ достаточно использовать только части векторов из подобластей Ω_{p-1} и Ω_{p+1}, соответствующие фронтам $l_p - 1$ и $r_p + 1$. Эти переменные и будут образовывать внешнюю границу области Ω_p.

Обозначим за u^b вектор граничных переменных для всех подобластей, и введем оператор сужения на граничные переменные $\hat{R} : \mathcal{R}^{N'} \to \mathcal{R}^{N_b}$, где N_b — размерность пространства следов. Тогда итерация метода Шварца в подпространстве следов выглядит следующим образом:

$$u_{i+1}^b = u_i^b + \hat{R} \sum_p R_p^T D_p^{-1} R_p (f - \bar{A}\hat{R}^T u_i^b).$$

Эту итерацию можно представить в виде $u_{i+1}^b = S(u_i^b) = T u_i^b + g$, где

$$T = I - \hat{R} \sum_p R_p^T D_p^{-1} R_p \bar{A} \hat{R}^T, \quad g = \hat{R} \sum_p R_p^T D_p^{-1} R_p f.$$

Сходимость стационарных итераций имеется при спектральном радиусе $\rho(T) < 1$. В то время как для эллиптических уравнений это условие выполнено, для произвольных уравнений это

не так. Однако можно заметить, что решение u_*^b должно удовлетворять $u_*^b = Tu_*^b + g$, что равносильно системе

$$[I - T]\, u^b = g. \tag{3}$$

Данную систему можно решать итерационными методами в подпространствах Крылова, например обобщенным методом минимальных невязок (GMRES) [2]. Для этого достаточно, чтобы матрица $[I-T] = \hat{R} \sum R_p^T D_p^{-1} R_p \bar{A} \hat{R}^T$ была невырожденной, что является существенно более слабым условием, по сравнению с условием $\rho(T) < 1$. Заметим также, что матрица $[I - T]$ не является симметричной, поэтому нельзя использовать экономичные методы в подпространствах Крылова, такие как метод сопряженных градиентов, ориентированные на решение симметричных СЛАУ и использующие короткие рекурсии для построения базиса подпространств Крылова.

Метод GMRES рассчитан на решение произвольных невырожденных СЛАУ вида $Ax = b$, при этом он минимизирует невязку $r_j = b - Ax_j$ в подпространстве Крылова

$$K_m(A, r_0) = \mathrm{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{m-1} r_0\}.$$

Псевдокод алгоритма следующий:

- $r_0 \leftarrow b - Ax_0$, $\beta = ||r_0||$, $q_0 \leftarrow r_0/||r_0||$, $\xi \leftarrow (1, 0, \ldots, 0)^T$
- $j \leftarrow 0, 1, \ldots$ while $||r_j|| > \varepsilon\beta$
  - $\tilde{q}_{j+1} \leftarrow Aq_j$
  - For $k \in [0, \ldots, j]$
    - $H_{k,j} \leftarrow (\tilde{q}_{j+1}, q_k)$
    - $\tilde{q}_{j+1} \leftarrow \tilde{q}_{j+1} - H_{k,j} q_k$
  - $H_{j+1,j} \leftarrow ||\tilde{q}_{j+1}||$, $q_{j+1} \leftarrow \tilde{q}_{j+1}/H_{j+1,j}$
  - For $k \in [0, \ldots, j-1]$
    - $\begin{bmatrix} H_{k,j} \\ H_{k+1,j} \end{bmatrix} \leftarrow \begin{bmatrix} c_k & s_k \\ -\bar{s}_k & c_k \end{bmatrix} \begin{bmatrix} H_{k,j} \\ H_{k+1,j} \end{bmatrix}$
  - $c_j \leftarrow |H_{j,j}|/\sqrt{|H_{j,j}|^2 + |H_{j+1,j}|^2}$
  - $\bar{s}_j \leftarrow c_j H_{j+1,j}/H_{j,j}$
  - $\begin{bmatrix} \xi_j \\ \xi_{j+1} \end{bmatrix} \leftarrow \begin{bmatrix} c_j & s_j \\ -\bar{s}_j & c_j \end{bmatrix} \begin{bmatrix} \xi_j \\ \xi_{j+1} \end{bmatrix}$
  - $H_{j,j} \leftarrow c_j H_{j,j} + s_j H_{j+1,j}$, $H_{j+1,j} \leftarrow 0$
  - $||r_{j+1}|| \leftarrow \beta|\xi_{j+1}|$
- $y_j \leftarrow \beta H^{-1} \xi$
- $x_j \leftarrow x_0 + [q_0 \ldots q_{j-1}] y_j$.

Из псевдокода видно, что для алгоритма требуется только операция умножения матрицы системы на вектор. Запишем результат умножения $I - T$ на вектор v через результат одной итерации метода аддитивного Шварца $S(v)$:

$$[I - T]\, v = v - Tv = v - S(v) + g, \quad g = S(0).$$

Таким образом, для решения СЛАУ (3) методом GMRES достаточно реализовать обычный метод аддитивного Шварца. Для решения задач в подобластях $x_p = D_p^{-1} b_p$ можно использовать какой-нибудь прямой решатель для разреженных систем. При этом LU-разложение матриц D_p достаточно вычислить всего один раз.

Оценим эффективность данного алгоритма для простого случая — решения трехмерного уравнения Пуассона в параллелепипеде со структурированной сеткой. Пусть имеется сетка с n вершинами по каждой из координатных осей. Тогда матрица системы будет иметь порядок $N = n^3$. При решении такой системы прямыми методами требуется $O(N^{5/3})$ памяти и $O(N^{7/3})$ операций [7], поскольку ширина ленты матрицы k в этом случае оказывается пропорциональной n^2.

Предположим теперь, что декомпозиция задачи на P подобластей выполнена равномерно с небольшими перехлестами, то есть порядок СЛАУ в каждой подобласти есть $O(N/P)$. Размерность пространства следов будем считать равной $O(P \cdot n^2) = O(P \cdot N^{2/3})$. Пусть для решения СЛАУ (3) требуется K итераций. Считая, что в подобластях сохраняются оценки для прямых решателей, получим, что общие затраты памяти M будут равны

$$M = O(P \cdot (N/P)^{5/3} + K \cdot P \cdot N^{2/3}) = O(N^{5/3}/P^{2/3} + K \cdot P \cdot N^{2/3}),$$

или, в пересчете на один узел (при условии использования распределенного GMRES),

$$M_p = O((N/P)^{5/3} + K \cdot N^{2/3}).$$

Время работы решателя T будет складываться из максимального времени факторизации t_{fact}, времени работы итерационного метода t_{iter} и времени на коммуникации t_{comm} ($T = t_{fact} + t_{iter} + t_{comm}$). Для них можно получить следующие оценки:

$$\begin{aligned} t_{fact} &= O((N/P)^{7/3}), \\ t_{iter} &= O(K \cdot (N/P)^{5/3} + K^2 \cdot N^{2/3}), \\ t_{comm} &= O(K \cdot P \cdot N^{2/3}). \end{aligned}$$

Данные оценки справедливы для распределенного GMRES. При использовании GMRES на одном узле оценка для t_{iter} увеличиваются до

$$t_{iter} = O(K \cdot (N/P)^{5/3} + K^2 \cdot P \cdot N^{2/3}).$$

Тогда имеем

$$T = O((N/P)^{7/3} + K \cdot (N/P)^{5/3} + K^2 \cdot P \cdot N^{2/3}).$$

Поскольку, как правило, K растет с ростом P, получаем, что при фиксированном N для T имеется оптимум при некотором значении P. Этот оптимум достигается, с одной стороны, за счет уменьшения времени факторизации блоков, и, с другой стороны, за счет увеличения временны́х затрат на ортогонализацию векторов и межпроцессные коммуникации.

3. Технологии реализация решателя

При выборе языка программирования, сторонних библиотек и технологий для реализации решателя учитывалась специфика используемых в решателе алгоритмов. Необходим был высокоуровневый язык и набор библиотек, хорошо подходящий для реализации различных алгоритмов на графах, манипуляций с векторами и разреженными матрицами. Дополнительное требование заключалось в том, чтобы компиляторы языка генерировали высокоэффективный код. В итоге выбор был сделан в пользу языка C++ и использования стандартной библиотеки STL, из которой использовались различные контейнеры, а также библиотеки линейной алгебры Eigen [4]. Последняя имеет в своем составе классы разреженных

матриц и динамических векторов, предоставляет удобный API для манипуляции ими, а также имеет встроенную векторизацию алгоритмов с использованием инструкций SSE.

Для решения задач в подобластях требуется эффективный решатель для СЛАУ с разреженными матрицами невысокого порядка, причем необходимо решать СЛАУ с различными правыми частями. Кроме того, обусловленность диагональных матриц D_p может быть довольно плохой. Для решения таких систем хорошо подходят прямые решатели разреженных систем. В качестве такого решателя был выбран PARDISO (PARallel DIrect SOlver) из библиотеки Intel ® MKL [3]. Данный решатель является многопоточным, что позволяет реализовать гибридный подход и повысить производительность: на каждом используемом узле кластера можно решать задачи для одной подобласти, причем факторизацию СЛАУ, выполняющуюся при инициализации решателя, и решение серии систем с разными правыми частями можно выполнять в многопоточном режиме с использованием всех доступных процессорных ядер.

Для организации взаимодействия процессов на узлах кластера был использован интерфейс MPI, который широко используется при написании распределенных программ. При этом решатель был построен на основе master-slave архитектуры, с выделенным основным узлом (с рангом MPI, равным 0). На этом узле решателю передается матрица, выполняется декомпозиция на подобласти, а также работает решатель GMRES. Остальные MPI процессы работают как ведомые, принимая от мастер-процесса матрицу системы на этапе инициализации и граничные переменные в процессе работы для решения задач в подобластях. Граничные для соседних подобластей переменные после решения отсылаются назад. Использование такого подхода позволяет на этапе итерационного решения ограничиться вызовами трех функций MPI на одну итерацию: MPI_scatter и MPI_gather для рассылки и сбора граничных переменных и MPI_Bcast для управления ведомыми процессами.

Представленный решатель был включен в состав библиотеки Krylov [5]. Интерфейс решателя, позволяющий вызывать его из процедур, написанных на различных языках программирования, представлен на рис. 1.

Рисунок 1

Интерфейс решателя

```
void cschwarz_solver(MPI_Comm *pcomm, const int *n,
            const double *a, const int *ia, const int *ja,
            double *u, const double *f,
            const double *eps, const int *ovl,
            const int *nmax, const int *nres, int *niter,
            double *time, int *err);
void cschwarz_client(MPI_Comm *pcomm);
```

Параметры решателя следующие:

pcomm — MPI коммуникатор, процессы которого участвуют в решении СЛАУ; процедура cschwarz_solver(...) должна вызываться в процессе с рангом 0, в остальных процессах необходимо вызывать cschwarz_client(...);

n — порядок СЛАУ;

a, ia, ja — матрица системы в формате CSR (Compressed Sparse Row format)[3];

u — вектор размера **n**, в который записывается полученное решение системы;

f — правая часть СЛАУ, должна иметь размер **n**;

eps — критерий останова итераций ε (**eps**) для решателя GMRES, решение считается найденным, если

$$||g - [I - T]u^b|| < \varepsilon||g||;$$

ovl — параметр пересечения подобластей; после построения разбиения на подобласти без пересечений каждая подобласть расширяется на **ovl** фронтов влево и вправо, соответственно пересечение соседних подобластей состоит из $2 \cdot$ **ovl** фронтов;

nmax — максимальное число итераций решателя GMRES;

nres — число итераций решателя GMRES до рестарта, позволяет ограничить длину последовательности q_j базисных векторов подпространства Крылова; после выполнения каждых **nres** итераций, даже если не была достигнута необходимая точность решения, текущее приближение пересчитывается по формулам

$$y_j = \beta H^{-1}\xi, \quad u_j^b \leftarrow u_0^b + [q_0 \dots q_{j-1}]y_j,$$

и новое значение u_j^b используется в качестве начального приближения для решателя;

niter — выходной параметр, количество совершенных решателем итераций;

time — время, затраченное на решение системы;

err — код возврата решателя; ненулевое значение свидетельствует об ошибке.

4. Численные эксперименты

В последующих таблицах приняты следующие обозначения: N — порядок системы, N_{nz} — количество ненулевых элементов в исходной матрице, N_b — размерность пространства следов, t_{fac} — время (в секундах) факторизации матриц D_p, t_{tot} — общее время решения и n — количество итераций. Эксперименты проводились на вычислительных узлах HP BL2x220c G6 кластера НКС-30Т [8], каждый из которых оборудован двумя четырехъядерными процессорами Intel Xeon E5540 и 16-ю гигабайтами памяти. В целях увеличения объема памяти, доступной PARDISO для хранения факторов матриц D_p, запускалось по одному MPI-процессу на узел, а для ускорения факторизации были задействованы все имеющиеся на узлах ядра. Параметр пересечения подобластей при решении всех задач был установлен таким образом, чтобы подобласти пересекались максимум по 8 фронтам, за исключением второй модельной задачи, где пересечение составляет 4 фронта. Параметр ε в критерии останова итераций был равен 10^{-7}. Рестарты итерационного процесса не производились, то есть значение параметра **nres** было выбрано достаточно большим.

4.1. Модельная задача для диффузионно-конвективного уравнения

В качестве первой модельной задачи была выбрана задача Дирихле для линейного диффузионно-конвективного уравнения

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + p\frac{\partial u}{\partial x} + q\frac{\partial u}{\partial y} + r\frac{\partial u}{\partial z} = f(x,y,z), \quad (x,y,z) \in \Omega,$$
$$u|_\Gamma = g(x,y,z),$$

в единичном кубе $\Omega = [0;1]^3$ на равномерной кубической сетке. Дискретизация задачи проводилась с помощью семиточечной схемы экспоненциального типа [7]. Функции f и g выбирались соответственно точному решению $u(x,y,z) = x^2 + y^2 + z^2$, а в качестве начального

приближения была взята функция $u^0(x, y, z) = 0$. В таблицах 1–3 приведены результаты решения данной модельной задачи при различных значениях p, q и r.

Максимальная погрешность решения первой модельной задачи в равномерной норме составила $7,7 \cdot 10^{-7}$.

Таблица 1

Решение первой модельной задачи при $p = q = r = 0$

Сетка	N	N_{nz}		Количество узлов			
				2	4	8	16
64^3	262144	1810432	N_b	6104	16196	34688	71786
			n	12	16	21	32
			t_{fac}	3,31	0,94	0,46	0,30
			t_{tot}	4,65	1,74	1.18	1,26
128^3	2097152	14581760	N_b	24536	65508	139183	285485
			n	17	23	29	40
			t_{fac}	219	33,3	8,36	2,56
			t_{tot}	245	44,5	15,6	9,25

Таблица 2

Решение первой модельной задачи при $p = q = r = 16$

Сетка	N	N_{nz}		Количество узлов			
				2	4	8	16
64^3	262144	1810432	N_b	6104	16196	34688	71786
			n	10	11	14	24
			t_{fac}	3,29	0,95	0,50	0,30
			t_{tot}	4,44	1,67	1,11	1,08
128^3	2097152	14581760	N_b	24536	65508	139183	285485
			n	14	16	20	29
			t_{fac}	233	34,7	8,73	2,60
			t_{tot}	255	43,9	14,8	8,23

Таблица 3

Решение первой модельной задачи при $p = q = r = -16$

Сетка	N	N_{nz}		Количество узлов			
				2	4	8	16
64^3	262144	1810432	N_b	6104	16196	34688	71786
			n	10	11	15	25
			t_{fac}	3,33	1,00	0,47	0,44
			t_{tot}	4,48	1,97	1,14	1,33
128^3	2097152	14581760	N_b	24536	65508	139183	285485
			n	14	17	21	30
			t_{fac}	226	33,9	8,67	3,01
			t_{tot}	248	43,5	15,2	8,77

4.2. Модельная задача с волноводом

Вторым примером является решение трехмерной краевой задачи для уравнения Гельмгольца

$$\nabla \times \left(\frac{1}{\mu_r} \nabla \times \vec{E} \right) - k_0^2 \dot{\varepsilon}_r \vec{E} = 0, \quad k_0 = \omega \sqrt{\varepsilon_0 \mu_0},$$

дискретизация которой выполнялась с помощью конечно-элементных аппроксимаций с базисными функциями Неделека второго порядка [9, 10]. Расчетной областью является волновод с линейными размерами $a = 72$, $b = 34$, $c = 200$ мм. Параметры задачи: $\mu_r = 1$, $\dot{\varepsilon}_r = 1 - 0,1i$, частота $\omega = 6\pi \cdot 10^9$ Гц. Граничные условия: на грани $z = 200$ мм задавалась касательная компонента поля

$$\vec{E}_0 \times \vec{n} = \vec{e}_y \sin(\pi x/a) \times \vec{n},$$

а на остальных гранях — условие металлической стенки ($\vec{E}_0 = 0$). Аналитическое решение в данном случае имеет вид

$$\vec{E} = \vec{e}_y \sin\left(\frac{\pi x}{a}\right) \frac{\sin \gamma z}{\sin \gamma c}.$$

В табл. 4 приводятся результаты решения указанной задачи при различных размерностях сетки.

Таблица 4

Решение второй модельной задачи

Сетка	N	N_{nz}		Количество узлов			
				2	4	8	16
$8 \times 4 \times 20$	21664	423392	N_b	1540	4678	10708	23052
			n	9	13	21	258
			t_{fac}	0,19	0,17	0,15	0,36
			t_{tot}	0,51	0,47	0,67	6,19
$15 \times 7 \times 40$	149874	3117779	N_b	5210	16646	37670	78446
			n	13	18	25	38
			t_{fac}	2,56	1,26	0,63	0,40
			t_{tot}	5,88	4,00	3,12	3,15
$29 \times 14 \times 80$	1196026	25795767	N_b	20562	67084	151032	318108
			n	18	26	34	49
			t_{fac}	108	39,9	14,9	5,29
			t_{tot}	173	85,6	43,3	32,94

Наличие точного решения позволяет проверить сходимость итерационного процесса. Данные тесты продемонстрировали ошибку порядка $O(h^2)$, что согласуется с теорией.

4.3. Примеры решения практических задач

В заключение данного пункта мы приведем результаты решения трех СЛАУ (обозначаемых ниже в табл. 5 как Г1, Г2, Г3), возникающих из актуальных практических краевых задач, описываемых трехмерными уравнениями гидродинамики, которые аппроксимируются с помощью неявных конечно-объемных схем на сложных неструктурированных сетках.

Таблица 5

Решение практических задач гидродинамики

Задача	N	N_{nz}		Количество узлов			
				2	4	8	16
Г1	4875172	26455278	N_b	180487	439384	895041	1831917
			n	40	58	85	162
			t_{fac}	101	64	49,8	19,7
			t_{tot}	166	117	119	201
Г2	1726272	11984539	N_b	6912	18533	42476	90000
			n	52	109	207	334
			t_{fac}	54,4	26,0	9,55	3,26
			t_{tot}	79,0	116	83,0	64,0
Г3	475000	13672497	N_b	2500	7500	17500	37110
			n	24	25	36	53
			t_{fac}	6,12	2,62	1,08	0,53
			t_{tot}	10,1	5,45	3,31	2,88

Следует отметить, что приведенные результаты параллельных вычислений дают значительный выигрыш в сравнении с однопроцессорными расчетами, в которых некоторые из рассмотренного типа СЛАУ с помощью «стандартных»предобусловленных крыловских алгоритмов решаются часами.

5. Заключение

Рассмотренные результаты позволяют сделать следующие выводы.

- С увеличением количества подобластей число итераций увеличивается, но (за одним исключением) не более чем линейно. В этом плане улучшение сходимости можно ожидать за счет включения других типов граничных условий на смежных границах, вместо используемого условия Дирихле.

- Применение прямого решателя PARDISO для подобластей является достаточно экономичным. В частности, с увеличением числа MPI-процессов относительные временные расходы на факторизацию вспомогательных подсистем значительно уменьшаются.

- Размерности СЛАУ в подпространствах следов (N_b) значительно меньше исходных порядков N. Это позволяет экономично реализовать с помощью GMRES итерации по подобластям.

- Описанная программная реализация позволяет проводить оптимизацию алгоритмов за счет выбора различных счетных параметров (величина перехлестов, варьирования числа ядер на подобласти, количества MPI-процессов на вычислительный узел и т.д.).
- При увеличении числа подобластей и используемых процессоров до нескольких тысяч целесообразен, по-видимому, переход от используемой 1-D декомпозиции к 2-D или даже 3-D декомпозиции.

Работа выполнена при поддержке грантов РФФИ №11-01-00205 и Президиума РАН №2.5.

Литература

1. Karypis G. A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs / Karypis G., Kumar V. // SIAM Journal on Scientific Computing. – 1999. – Vol. 20, № 1. – P. 359–392.
2. Saad Y. Iterative Methods for Sparse Linear Systems, Second Edition. – SIAM, 2003.
3. Intel (R) Math Kernel Library from Intel: сайт URL: http://software.intel.com/en-us/articles/intel-mkl/
4. Eigen: сайт URL: http://eigen.tuxfamily.org/index.php?title=Main_Page
5. Ильин В.П. Krylov: библиотека алгоритмов и программ для решения СЛАУ / Ильин В.П., Бутюгин Д.С., Ицкович Е.А и др. // Современные проблемы математического моделирования. Математическое моделирование, численные методы и комплексы программ. Сборник трудов Всероссийских научных молодежных школ. – Ростов-на-Дону: Изд-во Южного федерального университета, 2009. – С. 110–128.
6. Кормен Т. Алгоритмы: построение и анализ / Кормен Т., Лейзерсон Ч., Ривест Р. – М., МЦНМО: БИНОМ. Лаборатория знаний, 2004.
7. Ильин В.П. Методы и технологии конечных элементов. – Новосибирск: Изд-во ИВМиМГ СО РАН, 2007.
8. Кластер НКС-30Т: сайт URL: http://www2.sscc.ru/НКС-30Т/НКС-30Т.htm
9. Monk P. Finite Element Methods for Maxwell's Equations. – Oxford University Press, 2003.
10. Ingelstrom P. A new set of H(curl)-conforming hierarchical basis functions for tetrahedral meshes // IEEE Transactions on Microwave Theory and Techniques. – 2006. – Vol. 54, № 1. – P. 160–114.

Дмитрий Сергеевич Бутюгин, младший научный сотрудник, Институт Вычислительной Математики и Математической Геофизики СО РАН, аспирант, Новосибирский Государственный Университет, dm.butyugin@gmail.com.

Валерий Павлович Ильин, доктор физико-математических наук, профессор, главный научный сотрудник, Институт Вычислительной Математики и Математической Геофизики СО РАН, ilin@sscc.ru.

Данил Валерьевич Перевозкин, младший научный сотрудник, Институт Вычислительной Математики и Математической Геофизики СО РАН, dperevozkin@mail.ru.

Поступила в редакцию