Библиотека параллельных алгебраических решателей Krylov*

Д.С. Бутюгин[1,2], Я.Л. Гурьева[1], В.П. Ильин[1], Д.В. Перевозкин[1], А.В. Петухов[1], И.Н. Скопин[1]

Институт вычислительной математики и математической геофизики СО РАН[1], Новосибирский государственный университет[2]

Описываются функциональные возможности и особенности программной реализации библиотеки параллельных алгоритмов Krylov, ориентированной на решение больших систем линейных алгебраических уравнений с разреженными симметричными и несимметричными матрицами (положительно определенными и знаконеопределенными), получаемых при сеточных аппроксимациях многомерных краевых задач для систем дифференциальных уравнений на неструктурированных сетках. Библиотека включает двухуровневые итерационные методы в подпространствах Крылова, предобуславливание которых осуществляется на основе сбалансированной декомпозиции расчетной области с различными размерами пересечений подобластей и краевых условий сопряжения на смежных границах. Повышение скорости сходимости итерационных крыловских процессов производится с помощью грубосеточной коррекции, алгоритмов дефляции и неполной факторизации матриц в покомпонентном и блочном вариантах. Программные реализации выполнены на типовых сжатых разреженных форматах матричных данных. Приводятся примеры численных экспериментов с демонстрацией эффективности распараллеливания для характерных плохо обусловленных задач.

1. Введение

Настоящая работа содержит описание функционального наполнения и технологических подходов библиотеки параллельных итерационных алгоритмов Krylov (см. предварительные публикации в [1, 2]), ориентированной на решение больших систем линейных алгебраических уравнений (СЛАУ с числом неизвестных до 10^{10} и выше) с разреженными матрицами, возникающими при конечно-объемных или конечно-элементных (МКО или МКЭ [3, 4]) аппроксимациях многомерных краевых задач для систем дифференциальных уравнений на неструктурированных сетках, на многопроцессорных вычислительных системах (МВС с количеством ядер в десятки и сотни тысяч). В данном случае имеется в виду отсутствие программных ограничений на проведение таких масштабных вычислительных экспериментов, а также достаточно высокая эффективность применяемых алгоритмов.

Основой применяемых в библиотеке Krylov вычислительных подходов являются двухуровневые итерационные процессы в подпространствах Крылова, предобуславливаемые с помощью аддитивного метода Шварца и декомпозиции расчетной области с пересечениями подобластей и разными типами краевых условий на смежных внутренних границах, см. [4 – 7] и цитируемые там работы.

Внешний итерационный процесс осуществляется распределенным по вычислительным узлам образом методом FGMRES [7] с динамическими (в общем случае) предобуславливателями, которые включают решение вспомогательных подсистем в расширенных подобластях с помощью или прямого решателя PARDISO из библиотеки MKL INTEL [8], или авторскими версиями алгоритмов BiCGStab и GMRES [7, 9], предобусловленных с помощью покомпонентной или “мелкоблочной” модификациями Айзенштата неполной факторизации. Для ускорения внешних итераций используются методы дефляции и грубосеточной

*Работа поддержана грантом РФФИ N 11-01-00205, а также грантами Президиума РАН N 15.9-4 и ОМН РАН N 1.3.3-4.

коррекции [10].

Коммуникации между процессорами на внешних итерациях выполняются средствами системы MPI, а "внутренние" вычисления при решении СЛАУ в подобластях реализуются параллельными потоками над общей памятью с помощью OpenMP. Библиотека Krylov включает также итерационные решатели из написанной на языке CUDA библиотеки CuSP NVIDIA [11], что позволяет решать на гетерогенном кластере внутренние подсистемы на GPU.

Особенностью программных реализаций является то, что решаемые СЛАУ представляются в стандартном сжатом строчном формате CSR [8], который является практически безальтернативным способом построения универсальных алгебраических решателей, но представляет значительные трудности для эффективного распараллеливания алгоритмов. В частности, возникает нетривиальная задача формирования CSR-форматов для вспомогательных расширенных СЛАУ в подобластях.

Библиотека Krylov является функциональным аналогом таких пакетов распределенных итерационных решателей, как например PETSc [13], Hypre [14], HIPS и pARMS [15] и др., однако в Krylov предложен ряд оригинальных методов, обсуждаемых в работе.

В п. 2 мы описываем функциональное наполнение библиотеки Krylov, в п. 3 излагаются особенности программных реализаций алгебраических решателей, а в последнем разделе приводятся примеры численных экспериментов для характерных практических задач.

2. Функциональное наполнение библиотеки Krylov

Рассматривается решение вещественной СЛАУ

$$Au = f, \;\; A = \{a_{i,j}\} \in \mathcal{R}^{N,N}, \tag{1}$$

матрица которой, возможно, предварительно разбита на блочные строки $A = \{A_p \in \mathcal{R}^{N_p,N}, \; p = 1, ..., P\}, \;\; N_1 + ... + N_p = N$, с приблизительно равным числом строк $N_p \gg 1$ в каждой. Соответствующим образом также разбиваются векторы искомого решения и правой части $u = \{u_p\}, \;\; f = \{f_p\}$, так что система уравнений (1) может быть записана в форме совокупности P подсистем

$$A_{p,p}u_p + \sum_{\substack{q=1 \\ q \neq p}}^{P} A_{p,q}u_q = f_p, \;\; p = 1, ..., P, \tag{2}$$

где $A_{p,q} \in \mathcal{R}^{N_p,N_q}$ – прямоугольные матричные блоки, получаемые при разбиении каждой блочной строки (и всей матрицы A) на блочные столбцы.

Будем считать, что СЛАУ (1) представляет собой систему сеточных уравнений, так что каждая компонента векторов u, f соответствует узлу сетки, общее число которых в расчетной сеточной области $\Omega^h = \bigcup\limits_{p=1}^{P} \Omega_p$ равно N. При этом блочное разбиение матриц и векторов соответствует разбиению (декомпозиции) Ω^h на P сеточных непересекающихся подобластей Ω_p, в каждой из которых находится N_p узлов.

Описанная декомпозиция Ω^h не использует узлов – разделителей, т.е. условные границы подобластей не проходят через узлы сетки. Формальности ради можно считать, что внешность расчетной области есть неограниченная подобласть Ω_0 с числом узлов $N_0 = 0$.

Сеточное уравнение для i-го узла сетки может быть записано в виде

$$a_{i,i}u_i + \sum_{\substack{j \in \omega_i \\ j \neq i}} a_{i,j}u_j = f_i, \;\; i \in \Omega^h, \tag{3}$$

где через ω_i обозначается сеточный шаблон i-го узла, т.е. совокупность номеров всех узлов, участвующих в i-м уравнении.

Предполагается, что сеточные узлы и соответствующие переменные пронумерованы следующим образом: сначала идут подряд все узлы 1-й подобласти Ω_1 (неважно в каком внутреннем порядке), затем – узлы второй подобласти и т.д. Отметим, что взаимнооднозначного соответствия алгебраической и геометрической (сеточной) интерпретации структуры СЛАУ может и не существовать. Типичный пример – одному узлу сетки может соответствовать $m > 1$ переменных в алгебраической системе. Если для каждого узла такие "кратные" переменные нумеруются подряд, то структура СЛАУ приобретает "мелкоблочный" ($m \ll N$) вид (в (3) u_i и f_i означают не числа, а подвекторы порядка m соответственно, $a_{i,j} \in \mathcal{R}^{m,m}$) и на таких случаях мы остановимся в последующем особо.

Пусть матрица (1) задана в сжатом разреженном CSR-формате [8] с указанием числа строк N_p в каждой из P подсистем (2) в соответствии с разбиением матрицы A на блочные строки A_p. Естественно предполагается, что строки исходной матрицы пронумерованы подряд от 1 до N, так что номера строк каждого блока A_p меняются от $1 + \sum\limits_{i=1}^{p-1} N_i$ до $\sum\limits_{i=1}^{p} N_i$ (эти номера назовем глобальными).

На основе блочного представления СЛАУ (2) стандартным образом строится аддитивный метод Шварца, на алгебраическом языке представляющий блочный итерационный метод Якоби. Однако известно, что скорость сходимости итераций возрастает, если декомпозицию расчетной области сделать с пересечением областей.

В силу этого мы используем определение расширенной сеточной подобласти $\bar{\Omega}_p \supset \Omega_p$, имеющей пересечения с соседними подобластями, величину которых мы будем определять в терминах количества околограничных сеточных слоев, или фронтов. Обозначим через $\Gamma_p^0 \in \Omega_p$ множество внутренних околограничных узлов из Ω_p, т.е. таких узлов $P_i \in \Omega_p$, у которых один из соседей не лежит в $\Omega_p (P_j \notin \Omega_p,\ j \in \omega_i,\ j \neq i))$. Обозначим далее через Γ_p^1 множество узлов, соседних с узлами из Γ_p^0, но не принадлежащих Ω_p, через Γ_p^2 – множество узлов, соседних с узлами из Γ_p^1, но не принадлежащих объединению $\Gamma_p^1 \bigcup \Omega_p$, через Γ_p^3 – множество узлов, соседних с Γ_p^2, но не принадлежащих $\Gamma_p^2 \bigcup \Gamma_p^1 \bigcup \Omega_p$, и т.д. Соответственно эти множества назовем первым внешним слоем (фронтом) узлов, вторым слоем, третьим и т.д. Получаемое объединение узлов

$$\bar{\Omega}_p \equiv \Omega_p \bigcup \Gamma_p^1 ... \bigcup \Gamma_p^{\Delta}$$

будем называть расширенной p-й сеточной подобластью, а целую величину Δ определяем как величину расширения, или пересечения (в терминах количества сеточных слоев). Случай $\Delta = 0$ фактически означает декомпозицию области Ω^h на подобласти без пересечений ($\Omega_p^0 = \Omega_p$).

На формальном алгебраическом языке каждой расширенной подобласти можно сопоставить подсистему уравнений

$$(\bar{A}_{p,p} + \theta \bar{D}_p)\bar{u}_p = \bar{f}_p - \sum_{\substack{q=1 \\ q \neq p}}^{P} \bar{A}_{p,q}\bar{u}_q + \theta \bar{D}_p = \bar{r}_p, \tag{4}$$

где $\bar{u}_p, \bar{f}_p, \bar{r}_p \in \mathcal{R}^{\bar{N}_p}$, а $\bar{D}_p$ – диагональная матрица, определяемая соотношением

$$\bar{D}_p e = \sum_{\substack{q=1 \\ q \neq p}}^{P} A_{p,q} e, \ \ e = (1, ..., 1)^T \in \mathcal{R}^{\bar{N}_p}.$$

Здесь $\theta \in [0, 1]$ – некоторый итерационный параметр, который при $\theta = 0$ соответствует итерационному краевому условию Дирихле на смежных границах подобластей, при $\theta = 1$ – условию Неймана, а при $0 < \theta < 1$ – условию 3-го рода, или Робена.

Переходя теперь к полному вектору u и вводя матрицы

$$B_p = \bar{A}_{p,p} + \theta \bar{D}_p, \tag{5}$$

из (4) получаем итерационный аддитивный процесс Шварца в виде

$$u^n = u^{n-1} + B^{-1}(f - Au^{n-1}) = u^{n-1} + B^{-1}r^n, \tag{6}$$

где B – предобуславливающая матрица, определяемая следующим образом:

$$B^{-1} = B_{AS}^{-1} = \sum_{p=1}^{P} \bar{B}_p^{-1}, \;\; \bar{B}_p^{-1} = W_p^T B_p^{-1} W_p. \tag{7}$$

Здесь $W_p : \mathcal{R}^N \to \mathcal{R}^{\bar{N}_p}$ есть матрица сужения полного вектора в подвектор из $\bar{\Omega}_p$, а W_p^T – транспонированная матрица продолжения (расширения вектора из $\bar{\Omega}_p$ в Ω).

Выполнение каждой итерации в (6) требует параллельного решения внутренних СЛАУ в подобластях $\bar{\Omega}_p$, что может производиться или прямым методом, или итерационным алгоритмом крыловского типа. В последнем случае фактически реализуются переменные (динамические) предобуславливатели B_n.

При этом реально вместо (6) используется универсальный "гибкий" метод обобщенных минимальных невязок FGMRES, оптимизирующий невязку в подпространствах Крылова. Дальнейшее ускорение этого процесса осуществляется с помощью "улучшения" крыловских подпространств, что означает добавление новых базисных векторов и может быть интерпретировано как аддивное расширение предобуславливающих матриц:

$$B_n^{-1} = B_{AS,n}^{-1} + B_c^{-1} + B_d^{-1}. \tag{8}$$

3. Особенности параллельной реализации алгоритмов

Описанные выше параллельные алгоритмы реализованы в форме библиотеки Krylov, которая ориентирована на решение сверхбольших СЛАУ на системах с распределенной памятью, включающих в себя большое количество вычислительных узлов. В связи с этим реализация итерационных решателей СЛАУ и предобуславливателей имеет определенные особенности. В частности, требуется такая организация и структура методов, которая хорошо отображается на архитектуры имеющихся вычислительных систем.

3.1. Общая организация двухуровневых итераций

Решатели в библиотеке Krylov построены на основе широко распространенного стандарта интерфейса обмена данными MPI (Message Passing Interface). Базой для итерационного решения систем уравнений в библиотеке Krylov является метод FGMRES, распределенный по различным MPI-процессам. Данный метод совместно с различными предобуславливателями типа Шварца используется для решения заданной распределенной СЛАУ.

Псевдокод алгоритма следующий (см. [7]):

- $r_0 \leftarrow f - Au_0$, $\beta = ||r_0||$, $q_0 \leftarrow r_0/||r_0||$, $\xi \leftarrow (1, 0, \ldots, 0)^T$
- $n \leftarrow 0, 1, \ldots$ while $||r_n|| > \varepsilon\beta$
  - $z_n \leftarrow B_n^{-1} q_n$
  - $\tilde{q}_{n+1} \leftarrow Az_n$
  - For $k \in [0, \ldots, n]$
    - $H_{k,n} \leftarrow (\tilde{q}_{n+1}, q_k)$
    - $\tilde{q}_{n+1} \leftarrow \tilde{q}_{n+1} - H_{k,n} q_k$
  - $H_{n+1,n} \leftarrow ||\tilde{q}_{n+1}||$, $q_{n+1} \leftarrow \tilde{q}_{n+1}/H_{n+1,n}$

- For $k \in [0, \ldots, n-1]$
  - $\begin{bmatrix} H_{k,n} \\ H_{k+1,n} \end{bmatrix} \leftarrow \begin{bmatrix} c_k & s_k \\ -\bar{s}_k & c_k \end{bmatrix} \begin{bmatrix} H_{k,n} \\ H_{k+1,n} \end{bmatrix}$
- $c_n \leftarrow |H_{n,n}| / \sqrt{|H_{n,n}|^2 + |H_{n+1,n}|^2}$
- $\bar{s}_n \leftarrow c_n H_{n+1,n} / H_{n,n}$
- $\begin{bmatrix} \xi_n \\ \xi_{n+1} \end{bmatrix} \leftarrow \begin{bmatrix} c_n & s_n \\ -\bar{s}_n & c_n \end{bmatrix} \begin{bmatrix} \xi_n \\ \xi_{n+1} \end{bmatrix}$
- $H_{n,n} \leftarrow c_n H_{n,n} + s_n H_{n+1,n}$, $H_{n+1,n} \leftarrow 0$
- $||r_{n+1}|| \leftarrow \beta |\xi_{n+1}|$

- $y_n \leftarrow \beta H^{-1} \xi$
- $x_n \leftarrow x_0 + [z_0 \ldots z_{n-1}] y_n$.

Анализ вычислительной схемы алгоритма показывает, что основными операциями в методе FGMRES являются

- векторно-векторные операции (вычисление скалярных произведений, норм, и т.п.);
- матрично-векторные операции (умножение матрицы на вектор); в первых двух пунктах операции выполняются в полном N-мерном пространстве, где N – размерность решаемой СЛАУ;
- применение предобуславливателя (методы Шварца и грубосеточной коррекции, включающие решение вспомогательных СЛАУ относительно малого размера);
- QR-разложение матрицы Хессенберга, формируемой в процессе построения базисных векторов.

Все остальные операции требуют $O(1)$ времени и не вносят существенного вклада в производительность решателя. Более того, вклад QR-разложения во время работы решателей также можно игнорировать в предположении, что порядок системы много больше количества итераций. Такое разложение также не требует большого количества памяти, поэтому с целью уменьшения коммуникационных затрат оно выполняется на каждом MPI процессе.

Векторно-векторные операции в итерационных решателях распараллеливаются естественным образом за счет разбиения векторов, порождаемого декомпозицией матрицы на блочные строки. Единственная особенность реализации этих операций заключается в том, что для вычисления скалярного произведения и нормы необходима будет точка синхронизации и обмен данными, однако, к примеру, в интерфейсе MPI имеется требуемая функция **MPI_Allreduce**, реализация которой оптимизируется поставщиком библиотеки MPI.

Для проведения матрично-векторных операций решатель строит для каждой подобласти списки узлов, являющихся граничными для соседних подобластей. В дальнейшем, при умножении матрицы на вектор или при применении итерации Шварца к вектору неизвестных решатель использует полученную информацию для пересылки частей вектора между процессами. Ключевым моментом в таком подходе является уменьшение объема пересылаемых данных — необходимо пересылать в соседние подобласти только граничные коэффициенты вектора вместо того, чтобы пересылать вектор целиком. В случае, если разбиение на подобласти было построено подходящим способом, количество граничных вершин будет

существенно меньше размера подобласти (например, для двумерных задач количество граничных вершин N_p будет пропорционально квадратному корню из общего числа вершин в подобласти, а для трехмерных $N_\Gamma \sim N^{2/3}$).

Применение предобуславливателей, основанных на методе Шварца, требует на каждой итерации внешнего алгоритма решения подзадач в подобластях. Достоинством метода аддитивного Шварца является то, что решение в подобластях может проводиться независимо друг от друга и не требует коммуникаций. Действительно, как уже отмечалось выше, итерацию Шварца можно представить в виде (6), где предобуславливатель B составлен из блоков (5). Отсюда видно, что применение предобуславливателя B^{-1} в подобласти p не требует знания переменных, не принадлежащих ей.

3.2. Реализация внутренних итераций в подобластях

Для решения задач $A_{p,p}v_p = b_p$ в подобластях вида можно использовать какой-нибудь прямой решатель для разреженных систем, например решатель PARDISO из библиотеки Intel® MKL. Однако время, требуемое PARDISO для разложения матриц, в общем случае растет практически как $O(N^3/P^3)$, поэтому такой метод подходит только для решения не очень больших систем на большом числе процессоров. Однако действие обратных несимметричных матриц $A_{p,p}^{-1}$ можно вычислять приближенно при помощи предобусловленных методов в подпространствах Крылова, таких как GMRES и BiCGStab. В качестве предобуславливателя предлагается метод USOR в модификации Айзенштата, в том числе его мелкоблочный вариант. При этом пользователю предоставляется возможность выбора решателя для подобластей — либо PARDISO из Intel® MKL, либо одного из перечисленных выше итерационных решателей.

Рассмотрим более подробно предобуславливатель USOR в модификации Айзенштата. Пусть B — предобуславливающая матрица, записанная в форме

$$B = (G - L)G^{-1}(G - U) = D - L - U + LG^{-1}U, \tag{9}$$

где $A = D - L - U$, $D = block\text{-}diag\{D_k\}$, $D_k \in \mathcal{R}^{m,m}$, $k = 1, ..., M = N/m$ есть блочно-диагональная невырожденная матрица с диагональными блоками одинакового порядка m, причем порядок СЛАУ кратен m, а M — это блочный порядок матрицы A. Матрица G – некоторая пока не конкретизируемая невырожденная матрица, для которой вычислимо разложение $G = L_G U_G$ с треугольными множителями L_G и U_G.

С помощью обозначений

$$\bar{D} = L_G^{-1} D U_G^{-1},\ \bar{L} = L_G^{-1} L U_G^{-1},\ \bar{U} = L_G^{-1} U U_G^{-1} \tag{10}$$

предобусловленная матрица системы записывается в форме

$$\bar{A} = (I - I)^{-1} + (I - \bar{U})^{-1} + (I - \bar{L})^{-1}(\bar{D} - 2I)(I - \bar{U})^{-1}, \tag{11}$$

где I есть единичная матрица.

Отсюда предобусловленную СЛАУ можно представить как

$$\bar{A}\bar{u} = \bar{f} \equiv (L_G - L U_G^{-1})^{-1} f,\ \ \bar{u} = (U_G - L_G^{-1} U)u. \tag{12}$$

Важно отметить, что умножение $\bar{A}$ на некоторый вектор v можно представить в удобной для вычисления форме

$$\bar{A}v = (I - \bar{L})^{-1}[v + (\bar{D} - 2I)w] + w,\ \ w = (I - \bar{U})^{-1}v, \tag{13}$$

составляющей суть модификации Айзенштата.

Конкретизируем теперь выбор матрицы G. Мы используем

$$G = \frac{1}{\omega} D, \tag{14}$$

где ω – релаксационный числовой параметр. Очевидно, что такая матрица G сохраняет блочно-диагональную структуру D. В этом случае треугольное разложение G сводится к соответствующему разложению ее диагональных блоков:

$$G = block\text{-}diag\{G_k = L_{G,k} U_{G,k}\}, \tag{15}$$

а реализация умножения вектора на матрицу $\bar{A}$ упрощается за счет блочного представления матриц

$$\bar{L} = \{\bar{L}_{k,l} = L_{G,k}^{-1} L_{k,l} U_{G,l}^{-1}\}, \;\; \bar{U} = \{\bar{U}_{k,l} = L_{G,k}^{-1} U_{k,l} U_{G,l}^{-1}\},$$

$$k, l = 1, ..., M.$$

Более конкретно, при использовании блочного представления векторов $v = \{v_k\}, w = \{w_k\}, k = 1, ..., M$, решения фактически участвующих в (13) треугольных систем

$$(I - \bar{U})w = v, \;\; (I - \bar{L})y = z \equiv v + (\bar{D} - 2I)w$$

осуществляются фактически по следующим рекуррентным формулам:

$$\begin{aligned} &w_M = v_M, \;\; w_k = v_k + \sum_{l=k+1}^{M} \bar{U}_{k,l} v_l, \;\; k = M-1, ..., 1, \\ &y_1 = z_1, \;\; y_k = z_k + \sum_{l=1}^{k-1} \bar{L}_{k,l} z_l, \;\; k = 2, ..., M. \end{aligned} \tag{16}$$

Отметим, что используемые модификации неполной факторизации являются экономичными, но плохо распараллеливаемыми, вследствие необходимости решения вспомогательных СЛАУ с треугольными матрицами. Однако имеются различные подходы к распараллеливанию треугольных решателей, например метод планирования вычислений по уровням, см. [7]. Кроме того, умножение на "мелкоблочные" матрицы $U_{k,l}$ и $L_{k,l}$, если они вычислены заранее до итераций, для каждого фиксированного k позволяют обеспечить существенное ускорение средствами OpenMP.

4. Примеры численных экспериментов

Мы приведем результаты некоторых численных экспериментов по решению ряда практических задач с помощью библиотеки Krylov. В данных расчетах внешние итерации проводились с помощью метода GMRES с критерием окончания итераций по условию на евклидовую норму невязки

$$||r^n||_2 \le \varepsilon ||f||_2, \;\; \varepsilon = 10^{-7}.$$

Начальные приближения для искомых векторов всегда выбирались $u^0 = 0$.

Вспомогательные СЛАУ в подобластях решались либо с помощью прямого решателя PARDISO, причем наиболее трудоемкий этап – LU-разложение матрицы – выполнялся один раз до итераций, либо с помощью итерационного метода BiCGStab с предобуславливателем Айзенштата. Расчеты проводились на различном количестве P вычислительных узлах, с формированием такого же числа MPI-процессов. В нижеследующих таблицах N и NZ обозначают порядок решаемых СЛАУ, а также количество ненулевых элементов матрицы, которые характеризуют трудоемкость задачи.

Рассмотрим результаты экспериментов по решению трех несимметричных СЛАУ, возникающих из сеточных аппроксимаций практических задач гидро-газодинамики. Характеристики соответствующих плохо обусловленных матриц даны в табл. 1.

Табл. 1. Характеристики "гидро-газодинамических" матриц

Наименование	$N \cdot 10^{-6}$	$NZ \cdot 10^{-6}$
Г2	1.73	12.0
Г3	0.48	13.7
Г4	9.38	50.4

В табл. 2–4 приведены результаты расчетов с использованием PARDISO в качестве решателя в подобластях. В строках таблиц сверху вниз представлено общее время счета t в секундах и количество внешних итераций n_e.

Здесь и далее используются следующие обозначения: N_{nod} – число используемых вычислительных узлов; N_{mpi} – количество MPI-процессов на одном узле; $P = N_{nod} \cdot N_{mpi}$ – общее число MPI-процессов, равное количеству подобластей; Δ – параметр пересечения в декомпозиции (количество расширений сеточной подобласти); N_{th} – количество формируемых вычислительных потоков в одном MPI-процессе; T_1, – время решения СЛАУ без распараллеливания.

Табл. 2. Значения t и n_e для случая использования PARDISO в подобластях при $N_{th} = 4$, СЛАУ Г3 ($T_1 = 10.1$)

$P \setminus \Delta$	0	1	2	3	4	5
	9.45	7.46	7.23	6.80	6.75	6.52
5	82	41	29	22	18	15
	5.64	4.69	4.32	4.21	4.27	4.31
10	114	59	41	32	27	23
	6.03	4.09	3.72	3.76	3.77	3.59
20	164	84	58	44	37	32
	6.83	4.29	3.86	3.63	3.78	3.68
30	183	94	67	52	43	38

Табл. 3. Результаты решения СЛАУ Г4 с помощью PARDISO ($T_1 = 399.4$)

$P \setminus \Delta$	0	1	2	3	4	5
	280.4	199.4	153.9	146.5	140.6	143.5
5	235	119	85	67	55	47
	127.9	73.6	61.3	56.3	51.9	49.6
10	311	161	116	92	76	66
	147.4	71.2	51.9	47.0	41.3	40.6
20	331	175	127	101	84	72
	158.3	76.9	59.2	52.7	49.5	48.4
30	355	186	135	109	95	80

Табл. 4. Результаты экспериментов для СЛАУ Г2 с помощью PARDISO ($T_1 = 87.1$)

$P \setminus \Delta$	0	1	2	3	4	5
	60.6	39.9	33.9	30.9	28.5	27.4
5	235	116	78	59	46	37
	58.4	32.9	24.6	21.0	19.1	17.4
10	486	244	167	124	99	79
	118.5	47	28.6	22.6	18.3	16.0
20	934	479	330	246	194	157
	230.5	74.5	53.9	33.4	27.0	20.8
30	1352	690	473	355	280	228

Как видно из этих результатов, использование пересечений с ростом Δ дает стабильное уменьшение числа внешних итераций, а время счета уменьшается в два и в большее число раз (максимальный коэффициент ускорения – свыше 10). С ростом P время вычислений сначала уменьшается, но после $P > 20$ начинает увеличиваться, так как растет число внешних итераций. Для исправления ситуации, очевидно, надо использовать ускорение типа грубосеточной коррекции, которое в данных экспериментах не применялось.

Второй набор СЛАУ для численных экспериментов был представлен в блочно-строчном распределенном формате CSR, а заданное количество блочных строк указано в табл. 5. Там же указано, что первые две СЛАУ данного набора имеют “мелкоблочную” структуру с размером блоков, равным $m = 5$. В таблице также приведено количество подобластей P, на которые были разбиты соответствующие СЛАУ. Разбиение СЛАУ осуществлено без пересечений, так что, фактически, в данном наборе тестов $\Delta = 0$.

Табл. 5. Характеристики блочных распределенных СЛАУ

Наименование	$N \cdot 10^{-6}$	$NZ \cdot 10^{-6}$	P	m
Г5	9.5	330	48	5
Г6	3.7	126.8	36	5
Г7	6.5	44.4	20	1
Г8	0.35	1.74	20	1

В табл. 6 приведены результаты экспериментов для четырех СЛАУ из второго набора тестируемых задач. В столбцах таблицы указано общее время счета t, а также число внешних итераций n_e.

Для внутренних СЛАУ при использовании итерационных решателей фактически использовались ограничения числа итераций n^i_{max}. Для прямого решателя PARDISO в скобках указано число запускаемых им потоков при факторизации и решении СЛАУ. В качестве предобуславливателя для внутренних итерационных решателей использовался "мелкоблочный" предобуславливатель USOR в модификации Айзенштата. Для предобуславливателя USOR представлены результаты без использования средств распараллеливания OpenMP.

Табл. 6. Сравнение прямых и итерационных решателей для внутренних СЛАУ

Метод	Г5		Г6		Г7		Г8	
PARDISO ($N_{th} = 8$)	58.9	38	18.9	21	121.6	351	3.8	189
PARDISO ($N_{th} = 2$)	149.6	38	42.4	21	178.8	351	4.7	189
BBiCGStab ($n^i_{max} = 10$)	49.3	39	13.6	21	498.5	368	14.6	252
BBiCGStab ($n^i_{max} = 20$)	70.5	38	19.7	21	843.9	360	24.9	206

По результатам численных экспериментов можно сделать следующие выводы:

- на рассматриваемых СЛАУ с $m = 1$ прямой решатель PARDISO имеет существенное преимущество перед итерационными, причем увеличение в нем количества используемых потоков N_{th} от 2 до 8 дает коэффициент ускорения от 1.5 до 3;
- прямой решатель PARDISO проигрывает "мелкоблочному" итерационному алгоритму для СЛАУ с $m = 5$, хотя текущая реализация даже не использует средства OpenMP для распараллеливания на общей памяти;
- расчеты подтверждают ожидаемый факт, что при использовании итерационных внутренних решателей для СЛАУ в подобластях нет смысла добиваться высокой точности на различных внешних итерациях; более конкретно, при увеличении количества внутренних итераций n^i_{max} с 10 до 20 число внешних итераций несколько падает, но каждая из них делается "дороже" и общее время решения увеличивается.

Литература

1. Бутюгин Д.С., Ильин В.П., Ицкович Е.А и др. Krylov: библиотека алгоритмов и программ для решения СЛАУ // Современные проблемы математического моделирования. Математическое моделирование, численные методы и комплексы программ. Сборник трудов Всероссийских научных молодёжных школ. Ростов-на-Дону: Изд-во Южного федерального университета, 2009, 110-128.

2. Бутюгин Д.С., Ильин В.П., Перевозкин Д.В. Методы параллельного решения СЛАУ на системах с распределенной памятью в библиотеке Krylov.// Вестник ЮУрГУ. Серия “Вычислительная математика и информатика”, т. 47, N 306, 2012, 5-19.

3. Ильин В.П. Методы конечных разностей и конечных объемов для эллиптических уравнений.–Новосибирск, изд. ИВМиМГ СО РАН, 2001.

4. Ильин В.П. Методы и технологии конечных элементов.–Новосибирск, изд. ИВМиМГ СО РАН, 2007.

5. Ильин В.П. Параллельные методы и технологии декомпозиции областей.// Вестник ЮУрГУ. Серия “Вычислительная математика и информатика”, 2012, N 46(305), 31-44.

6. Ильин В.П., Кныш Д.В. Параллельные методы декомпозиции в пространствах следов.//Вычислительные методы и программирование, изд. МГУ, т. 12, N 1, 2011, 100-109.

7. Saad Y. Iterative Methods for Sparse Linear Systems, Second Edition.SIAM, 2003.

8. Intel Math Kernel Library. Reference Manual:
URL: http://software.intel.com/sites/products
/documentation/hpc/composerxe/enus/mklxe/mkl_manual_win_mac/index.htm.

9. Ильин В.П. Методы бисопряженных направлений в подпространствах Крылова.–СибЖИМ, т. 11, N 4(36), 2008, 47-60.

10. Nabben R., Vuik C. A comparison of abstract versions of deflation, balancing and additive coarse grid correction preconditioners.–Num. Lin. Alg. with Appl., v. 15, 2008, 355-372.

11. Bell N., Garland M. Cusp: Generic Parallel Algorithms for Sparse Matrix and Graph Computations, 2012. URL: http://cusp-library.googlecode.com

12. Кластер НКС-30Т. URL: http://www2.sscc.ru/HKC-30T/HKC-30T.htm

13. PETSc: Home Page. URL: http://www.mcs.anl.gov/petsc/

14. Hypre. URL: http://acts.nersc.gov/hypre/

15. Yousef Saad – Software. URL: http://www-users.cs.umn.edu/~saad/software/