УДК 519.612

РЕШЕНИЕ РАЗРЕЖЕННЫХ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ МЕТОДОМ ГАУССА С ИСПОЛЬЗОВАНИЕМ ТЕХНИКИ АППРОКСИМАЦИИ МАТРИЦАМИ МАЛОГО РАНГА

С. А. Соловьев[1]

Предложен алгоритм решения систем линейных алгебраических уравнений (СЛАУ), получаемых в результате дискретизации трехмерных уравнений математической физики. Алгоритм основан на методе исключения Гаусса с использованием вложенных сечений и аппроксимации матрицами малого ранга. Без ограничения общности алгоритм описан для случая симметричных положительно определенных матриц. Для хранения матрицы L в LU-разложении исходной матрицы используется крупноблочное представление, а также иерархический формат HSS (Hierarchically Semiseparable Structure). Для построения малоранговой аппроксимации предложено использование адаптивной крестовой аппроксимации, что более эффективно по сравнению с известными SVD- и QR-методами. Для повышения эффективности программной реализации алгоритма используются подпрограммы библиотек Intel MKL BLAS и LAPACK. На основе предлагаемого алгоритма и использования Intel MKL BLAS/LAPACK создана научно-исследовательская версия программного обеспечения для вычислительных систем с общей памятью. Приведены результаты тестирования, которые показали высокое качество предложенного алгоритма малоранговой/HSS-аппроксимации. Тестирование производительности и примененные способы использования памяти показывают более чем трехкратное превосходство предложенного подхода по сравнению с программным пакетом PARDISO библиотеки Intel MKL.

Ключевые слова: трехмерные задачи математической физики, алгоритмы решения разреженных линейных систем, метод Гаусса, аппроксимация матрицами малого ранга, HSS-формат матриц, итерационное уточнение.

Введение. Задача решения систем линейных алгебраических уравнений (СЛАУ) является одной из фундаментальных задач вычислительной математики. Зачастую такие системы возникают при решении задач математической физики, а именно в результате дискретизации уравнений в частных производных. Матрицы получающихся систем сеточных уравнений являются разреженными, т.е. с относительно небольшим числом ненулевых элементов. Число неизвестных в СЛАУ является достаточно большим, в настоящее время практический интерес представляет решение трехмерных задач с порядком матрицы от 10^6 до 10^9 и выше, однако общее число ненулевых элементов по отношению к порядку матрицы остается практически неизменным. В связи с этим возникают особые требования к разработке методов и алгоритмов решения таких систем сеточных уравнений.

Большинство методов решения СЛАУ можно разделить на два класса: итерационные методы, позволяющие найти приближенное решение системы с заданной точностью, и прямые методы, вычисляющие точное решение (с машинной точностью) за конечное число арифметических операций.

Преимущество итерационных методов заключается в минимальных затратах по использованию памяти. Основной недостаток состоит в том, что число итераций метода существенно зависит от числа обусловленности исходной матрицы A. Существуют различные подходы к построению итерационных методов, большинство из них — это проекционные методы, а также методы, основанные на подпространстве Крылова (GMRES, CG, BCG и т.п.). Кроме того, значимым направлением в развитии итерационных методов является техника построения предобусловливающих матриц, например методы ILU и Multigrid. Следует отметить, что метод Multigrid был предложен Р. П. Федоренко и Н. С. Бахваловым и под названием "многосеточный метод" [1, 4]. Наиболее полный обзор итерационных методов приведен в работах [17, 18].

Прямые методы, в отличие от итерационных, дают точное решение (с машинной точностью) с заранее известным числом арифметических операций и эффективны для решения СЛАУ со многими правыми частями. Указанные методы основаны на разложении (факторизации) матрицы на множители (чаще всего LU-разложение). Однако они более требовательны к размеру памяти и к временны́м ресурсам,

[1] Институт нефтегазовой геологии и геофизики им. А. А. Трофимука СО РАН, просп. Академика Коптюга, 3, 630090, г. Новосибирск; науч. сотр., e-mail: 511ssa@mail.ru

так как в процессе факторизации теряется свойство разреженности, а множители L и U содержат на несколько порядков больше ненулевых элементов, чем исходная матрица. К примеру, для двумерной задачи дискретизации методами конечных разностей (МКР), элементов (МКЭ) или объемов (МКО) на сетке размером $n \times n$ с общим числом неизвестных $N = O(n^2)$ число ненулевых элементов в матрице A равно $O(N)$, а в соответствующих ей множителях $L(U)$ уже $O(N^{3/2})$, при этом число арифметических операций для разложения равно $O(N^2)$. Для трехмерных задач на сетке $n \times n \times n$ с общим числом неизвестных $N = O(n^3)$ число ненулевых элементов составляет соответственно $O(N)$ и $O(N^{5/3})$, а число операций $O(N^{7/3})$. Эффективность прямых методов может быть повышена путем применения алгоритмов параллельных и вложенных сечений, уменьшающих это заполнение [2, 14]. В результате для двумерной задачи заполняемость $L(U)$-множителей уменьшается до $O(N \log_2 N)$, а число арифметических операций до $O(N^{3/2})$. Для трехмерных задач имеем соответственно $O(N^{4/3})$ и $O(N^2)$, что непозволительно много для практических задач.

Для дальнейшего повышения эффективности прямых методов в последнее время развивается техника аппроксимации обратной матрицы A^{-1}, а также множителей $L(U)$ блочно-малоранговой структурой. Пионерами в области построения блочно-малоранговых аппроксимаций являются российские ученые, в частности Е. Е. Тыртышников [19, 20]. Для хранения аппроксимированных матриц В. Хакбушем была предложена HSS-структура, а также HSS-арифметика для работы с ними [12, 13]. Кроме того, для ряда задач доказано [6], что множители L и U имеют блочную структуру со свойствами малого ранга для внедиагональных блоков. Диагональные блоки эффективно аппроксимируются H-матрицами и записываются в HSS-структуре [12]. После применения указанной техники малоранговой/HSS аппроксимации для двумерных задач размер памяти для хранения $L(U)$-множителей равен $O(N)$, число операций для построения множителей также равно $O(N)$. Для трехмерных задач размер памяти $O(N \log_2 N)$ (а для некоторых задач $O(N)$), а число операций $O(N^{4/3})$ [25]. Применение указанной техники в прямых методах широко используется в последнее время [15, 24, 26].

В настоящей статье предложен алгоритм решения СЛАУ, основанный на применении техники вложенных сечений и использовании малоранговой/HSS-аппроксимации. Приведено наглядное описание алгоритма отложенной модификации для построения множителя L, а также алгоритмов обратной и прямой подстановок в блочном виде. Универсальность указанных алгоритмов заключается в том, что блоки могут рассматриваться как в плотном, так и в малоранговом либо HSS-представлении, а арифметические операции могут производиться как в точной, так и в малоранговой арифметике.

Предложены следующие оригинальные приемы, позволяющие дополнительно уменьшить временны́е ресурсы и затраты по памяти.

1. Основываясь на предварительном анализе, принимается индивидуальное решение для каждого блока множителя L: использовать его малоранговое представление либо сохранить его в плотном виде. Такой выбор позволяет перейти от строгого эвристического выбора критерия применимости малоранговой аппроксимации по уровням дерева исключений [22, 25–27] к более гибкому и универсальному выбору критерия использования малоранговой аппроксимации.

2. Применяя алгоритм адаптивной крестовой аппроксимации (АКА) вместо известного QR-разложения [23], удалось уменьшить временны́е затраты на построение малоранговой аппроксимации. Для упаковки в формат HSS предложено использование АКА-аппроксимации с последующей ортогонализацией методом QR, что также сокращает время счета. Таким образом, общее время на сжатие и факторизацию сравнительно небольших разреженных матриц с общим числом неизвестных порядка 10^6 в 1.5–2 раза меньше временны́х затрат прямого метода, реализованного в Intel MKL PARDISO, а при увеличении размера задач это преимущество увеличивается.

3. Опираясь на анализ двух алгоритмов HSS-факторизации плотных матриц [7], реализован алгоритм получения $L_H L_H^T$-разложения диагональных H-блоков в гауссовом исключении, являющийся более экономным по памяти, чем используемый в работах [25–27].

4. На основе использования метода итерационного уточнения удалось увеличить точность полученного решения до точности прямых методов. При этом временны́е затраты увеличились незначительно, а сложность реализации алгоритма итерационного уточнения минимальна по сравнению с использованием малорангового/HSS-разложения в качестве предобусловливателя.

Статья состоит из четырех разделов. В первом разделе описана специфика алгоритма решения разреженных СЛАУ на основе метода Гаусса; на двух- и трехмерных примерах описывается алгоритм вложенных сечений. Во втором разделе рассматривается аппроксимация нижнетреугольной матрицы L блоками малого ранга; описывается структура формата HSS и его применение для хранения диагональных блоков матрицы L. В третьем разделе представлен алгоритм метода Гаусса с использованием техники

малоранговой/HSS-аппроксимации; приведены алгоритмы построения матрицы L, алгоритмы прямой и обратной подстановок, а также алгоритм итерационного уточнения. Результаты численных экспериментов представлены в четвертом разделе. Тестовые расчеты показывает корректную работу алгоритма, а также подтверждают качество малоранговой аппроксимации. Тестирование производительности и применяемые способы использования памяти показывают высокую эффективность по сравнению с пакетом PARDISO из высокопроизводительной математической библиотеки Intel MKL.

1. Алгоритм решения разреженных СЛАУ на основе метода Гаусса. Рассмотрим систему линейных уравнений

$$Ax = b \tag{1}$$

с невырожденной матрицей $A \in R^{n\times n}$. В общем случае A — заполненная матрица, хотя ниже рассматривается алгоритм решения СЛАУ, оптимизированный под разреженные матрицы, т.е. в которых число ненулевых элементов на несколько порядков меньше общего числа элементов. Наиболее известная форма гауссова исключения та, в которой система (1) приводится к верхнетреугольному виду путем вычитания одних уравнений, умноженных на подходящие числа, из других уравнений; полученная треугольная система решается с помощью обратной подстановки. Математически это эквивалентно тому, что сначала строится разложение $A = LU$ (этап LU-факторизации), где L — нижнетреугольная матрица с единицами на главной диагонали (L-множитель), а U — верхнетреугольная матрица (U-множитель). Затем треугольные системы $Ly = b$ и $Ux = y$ решаются прямой и обратной подстановками (второй этап). Следует отметить, что построение разложения LU требует $O\big(n^3\big)$ арифметических операций, что значительно больше по сравнению со вторым этапом: $O\big(n^2\big)$ операций.

Если A — симметричная знаконеопределенная матрица, то часто используемой альтернативой гауссову исключению является $A = LDL^T$-разложение, где L — нижнетреугольная матрица с единицами на главной диагонали, а D — диагональная матрица. Чтобы завершить решение линейной системы $Ax = b$, надо выполнить прямую и обратную подстановки, а также решить систему с диагональной матрицей D: $Lz = b,\ Dy = z,\ L^Tx = y$.

Кроме того, следует отметить распространенный случай, когда A — симметричная положительно определенная матрица. Тогда применяется разложение Холесского $A = LL^T$ с нижнетреугольной матрицей L. Прямая и обратная подстановки будут иметь вид $Ly = b,\ L^Tx = y$.

Далее без ограничения общности и для простоты понимания рассматривается СЛАУ с симметричными положительно определенными матрицами. Следует отметить, что для решения знаконеопределенных систем принцип работы алгоритма остается прежним, а в реализации добавляется техника выбора ведущего элемента.

1.1. Особенности использования гауссова исключения для решения двух- и трехмерных задач математической физики. Матрицы, получаемые в результате дискретизации дифференциальных уравнений в частных производных как для двух-, так и для трехмерных задач, содержат большое количество нулевых элементов. Поэтому имеет смысл рассмотреть структуру (портрет) матрицы, т.е. информацию о том, в каких позициях матрицы хранятся ненулевые элементы.

В качестве ключевого момента в понимании особенностей таких матриц является решение двумерных краевых задач для дифференциальных уравнений второго порядка на примере уравнения Гельмгольца (в частности, Лапласа). Рассматривается прямоугольная расчетная область с пятиточечной разностной аппроксимацией на прямоугольной сетке, образованной семействами линий, параллельных границам исходной области (n_1 вертикальных и n_2 горизонтальных). Известно, что структура матрицы двумерной разностной задачи в значительной степени зависит от того, какая выбирается упорядоченность узлов сетки; соответственно определяются векторы неизвестных и правых частей. Наиболее простой является построчная нумерация, в которой нумеруются поочередно все узлы из 1-й строки, затем из 2-й и т.д. до последней. Отсюда следует, что матрица A является пятидиагональной и ее представление наиболее наглядно в блочном трехдиагональном виде, где каждый блок, расположенный на главной диагонали, — это квадратная трехдиагональная матрица порядка n_1, а два других блока, расположенные на соседних диагоналях, — квадратные диагональные, тоже порядка n_1. Каждая блочная строка матрицы A соответствует уравнениям одной линии сетки, а блочный порядок матрицы равен n_2 (рис.1а).

В трехмерном случае рассматривается семиточечная разностная схема на параллелепипедальной сетке, образованной вертикальными (n_1, n_2) и горизонтальными координатными плоскостями (n_3). Матрица такой системы имеет порядок $n_1n_2n_3$ и может быть представлена в блочно-трехдиагональной форме аналогично двумерному случаю. Узлы нумеруются по горизонтальным плоскостям: сначала все узлы из первой плоскости (порядок нумерации аналогичен двумерному случаю), затем из второй и т.д. до n_3. Не вдаваясь в подробности построения всей матрицы, следует отметить, что структура блоков, расположен-

ных на главной диагонали, аналогична структуре всей матрицы для двумерного случая и имеет порядок $n_1 n_2$; блоки, расположенные на соседних диагоналях, — квадратные диагональные порядка $n_1 n_2$. Каждая блочная строка матрицы A соответствует уравнениям одной плоскости сетки, а блочный порядок матрицы равен n_3 (рис. 1b).

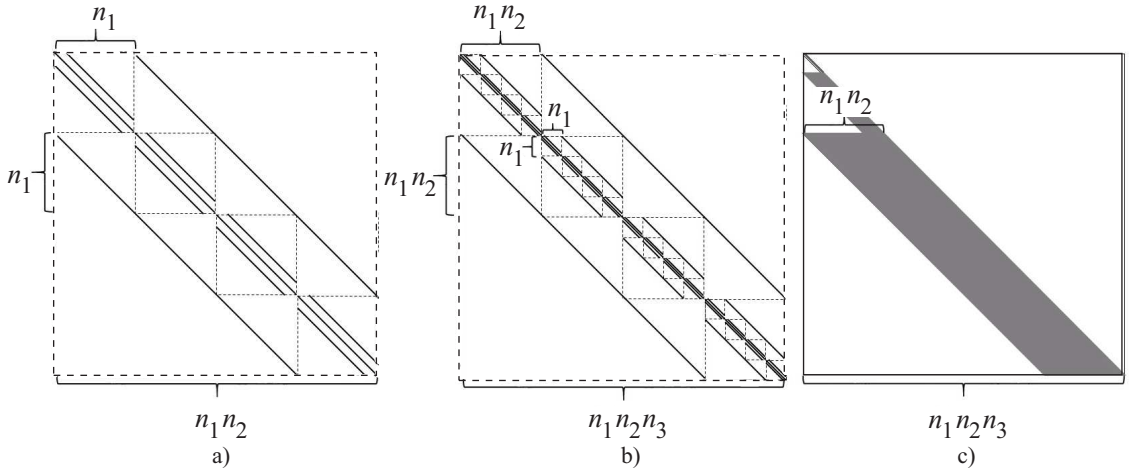

Рис. 1. Портрет исходной матрицы A для двумерной (a) и трехмерной (b) областей, а также нижнетреугольной матрицы L (c)

Для оценки трудоемкости метода Гаусса, а также затрат по памяти для хранения данных считается, что число расчетных узлов по каждому направлению одинаково: $n = n_1 = n_2 = n_3$. Если же учесть свойство симметричности рассмотренных матриц, то в двумерном случае достаточно хранить одну главную диагональ и две кодиагонали ($3n^2 - 2n$ ненулевых элементов); в трехмерном — одну главную диагональ и три кодиагонали ($4n^3 - 3n^2$ ненулевых элементов).

Несмотря на такой экономный способ хранения, после применения алгоритма факторизации Холесского получаются ленточные заполненные матрицы с шириной ленты n в двумерном и n^2 в трехмерных случаях (рис. 1c). Исходя из того что число арифметических операций для LL^T-разложения ленточной матрицы порядка m и шириной ленты k равно $O(k^2 m)$, а число ненулевых элементов равно $O(km)$, получается, что число арифметических операций Q_1 и объем оперативной памяти P_1 в двумерном случае приблизительно равны $O(n^4)$ и $O(n^3)$, но в трехмерном — $O(n^7)$ и $O(n^5)$ соответственно. Это непозволительно много, особенно для трехмерных задач, так как для расчетной сетки 100^3 необходимо порядка 80 GB для хранения матрицы L с удвоенной точностью.

Одним из путей повышения эффективности алгоритма гауссова исключения и уменьшения затрат по памяти является использование алгоритмов переупорядочивания элементов исходной матрицы, эквивалентного перенумерации узлов расчетной сетки, с последующим применением алгоритма гауссова исключения. Переупорядочивание эквивалентно умножению исходной матрицы A слева и справа на соответствующую матрицу перестановок P.

В результате решение исходной системы сводится к перестановке элементов правой части b, перестановке строк и столбцов матрицы A, решению системы с новой матрицей $\widehat{A}$, а также к обратной перестановке элементов в полученном решении $\widehat{x}$:

$$\widehat{b} = Pb, \quad \widehat{A} = PAP^T, \quad \widehat{A}\widehat{x} = \widehat{b}, \quad x = P^T\widehat{x}. \tag{2}$$

Следует отметить, что матрица перестановок P хранится в виде вектора перестановок длины N. Кроме того, матрица $\widehat{A}$, как и матрица A, обладает свойством симметричности и положительной определенности, что позволяет использовать LL^T-разложение.

1.2. Метод вложенных сечений для уменьшения числа ненулевых элементов в LL^T-разложении. Наиболее эффективный алгоритм переупорядочивания основывается на методе вложенных сечений (Nested Dissection method; ND-method), позволяющем на порядок увеличить эффективность алгоритма гауссова исключения [10, 11]. Ниже приводится описание алгоритма нумерации узлов в указанном методе на примере двумерной квадратной сетки. Как упоминалось выше, естественная строчно-столбцовая нумерация дает блочно-трехдиагональный портрет матрицы A (рис. 1) и ленточный для L-множителя.

Первым шагом ND-алгоритма является выбор множества узлов Ω_3, лежащих на среднем столбце сетки. Пусть Ω_1 — множество узлов, находящихся слева от Ω_3, а Ω_2 — узлы справа (рис. 2).

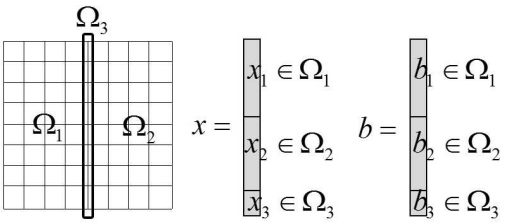

Рис. 2. Нумерация узлов сетки после первого шага алгоритма вложенных сечений

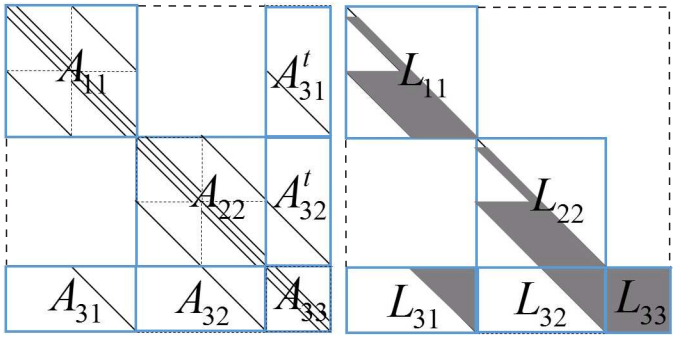

Рис. 3. Портреты матриц $\widehat{A}$ и L после 1-го шага переупорядочивания алгоритмом вложенных сечений

Далее по столбцам нумеруются все узлы из Ω_1, затем из Ω_2 и Ω_3. В матричном представлении это эквивалентно тому, что в векторе правой части b и решении x компоненты располагаются поочередно из областей Ω_1, Ω_2 и Ω_3, а матрица имеет следующий портрет.

Блоки A_{11}, A_{22} и A_{33} отвечают за сеточную аппроксимацию дифференциальной задачи внутри областей Ω_1, Ω_2 и Ω_3 соответственно. Блок A_{31} — за связи между областями Ω_1 и Ω_3; A_{32} — между Ω_2 и Ω_3. Так как изначально был выбран пятиточечный шаблон аппроксимации, то никакие два узла из разных областей Ω_1 и Ω_2 не имеют между собой связей и соответствующие блоки в матрице состоят из нулей. При выборе более сложных шаблонов, например при аппроксимации более высокого порядка, в качестве области Ω_3 необходимо использовать два сеточных столбца (или более) таким образом, чтобы области Ω_1 и Ω_2 оставались не связанными используемым шаблоном.

Таким образом, структура L-множителя (рис. 3b) обладает двумя основными свойствами:

1) процесс гауссова исключения для A_{22} и A_{32} выполняется независимо (параллельно) от исключения для A_{11} и A_{31};

2) число ненулевых элементов в полученном L-множителе меньше, чем в L для исходной матрицы.

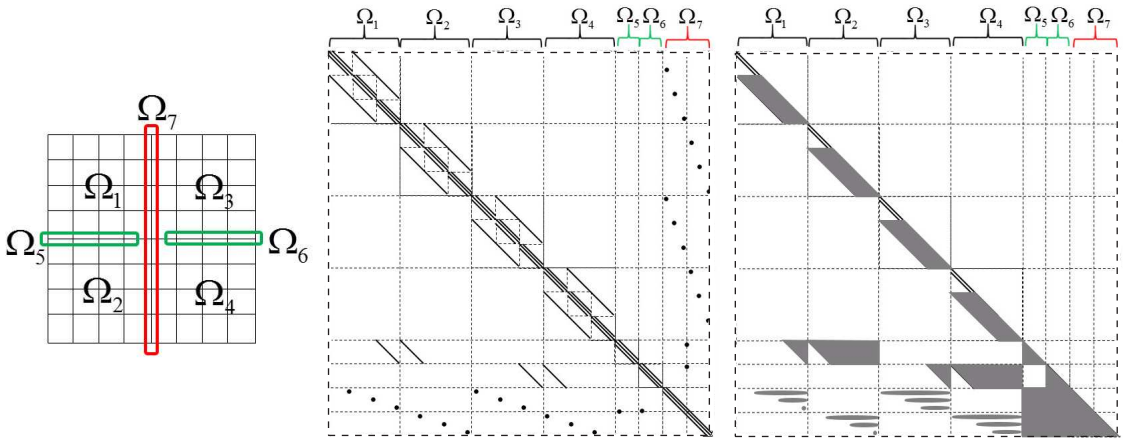

Рис. 4. Второй шаг алгоритма ND: a) разделение расчетной сетки на подобласти; портреты матриц $\widehat{A}$ (b) и L (c), соответствующие полученному разделению

Для большего понимания соответствия между сеточным представлением и получаемой матрицей $\widehat{A}$ отметим, что при удалении области Ω_3 из расчетной области получаются две независимые области Ω_1 и Ω_2 и матрица вырождается в блочно-двухдиагональную, а система $\widehat{A}\widehat{x} = \widehat{b}$ распадается на две независимые системы $A_{11}x_1 = b_1$ и $A_{22}x_2 = b_2$, соответствующие двум полученным областям.

Второй шаг алгоритма ND — дальнейшее разбиение области на подобласти (рис. 4).

Нумерация узлов следующая: первыми рассматриваются поочередно узлы из областей Ω_1, Ω_2, Ω_3 и Ω_4. Далее нумеруются узлы в подстроках Ω_5 и Ω_6, а также в столбце Ω_7. Шаблон матрицы A после такой нумерации сложнее, однако, как и после первого шага, он обеспечивает уменьшение числа ненулевых эле-

ментов в L-множителе, а также независимость процессов гауссова исключения для подматриц из каждой из подобластей Ω_1, Ω_2, Ω_3 и Ω_4. После факторизации указанных областей возможно также независимое рассмотрение областей Ω_5 и Ω_6.

Предложенное разбиение удобно представить в виде двоичного дерева T. Гауссово исключение для сеточных узлов из некоторой вершины дерева выполняется после выполнения факторизации его левого и правого поддерева. Гауссово исключение для сеточных узлов из разных вершин дерева, расположенных на одном уровне, выполняется независимо друг от друга (рис. 5).

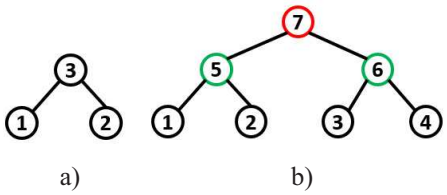

Рис. 5. Деревья зависимостей для процесса факторизации после 1-го (a) и 2-го (b) шагов алгоритма ND

При продолжении разбиения расчетной сетки получается вложенная древовидная структура подобластей Ω_i с соответствующей им матрицей $\widehat{A}$ и L-множителем (см. рис. 6).

В трехмерном случае нумерация узлов в алгоритме ND аналогична двумерному. В качестве разделителей области задаются вертикальные и горизонтальные плоскости, состоящие из узлов расчетной сетки.

Матрицу $\widehat{A}$ можно представить в крупноблочном виде. Каждому разделителю Ω_i расчетной сетки соответствует набор столбцов матрицы A. Эти столбцы разбиваются по горизонтали следующим образом. Диагональный блок $\widehat{A}_{ii}$ соответствует сеточным связям внутри разделителя Ω_i. Внедиагональные блоки $\widehat{A}_{ji}$ — это связи между узлами сетки разделителя Ω_i с узлами сетки из Ω_j, родителем либо предком для узла i в дереве зависимостей T. Аналогичным образом записывается крупноблочное представление матрицы L.

Отметим, что если в матрице $\widehat{A}$ ненулевые блоки сильно разрежены (переупорядочивание элементов исходной матрицы не меняет общего числа ненулевых элементов), то в L-множителе указанные блоки, разреженные для первых столбцов, довольно быстро становятся более плотными по мере выполнения факторизации оставшейся части матрицы. В качестве примера на рис. 6 светло-серые блоки — разреженные, серые — уплотненные, черные — плотные. На практике, можно сказать, основной объем L-множителей приходится на хранение указанных уплотненных и плотных блоков.

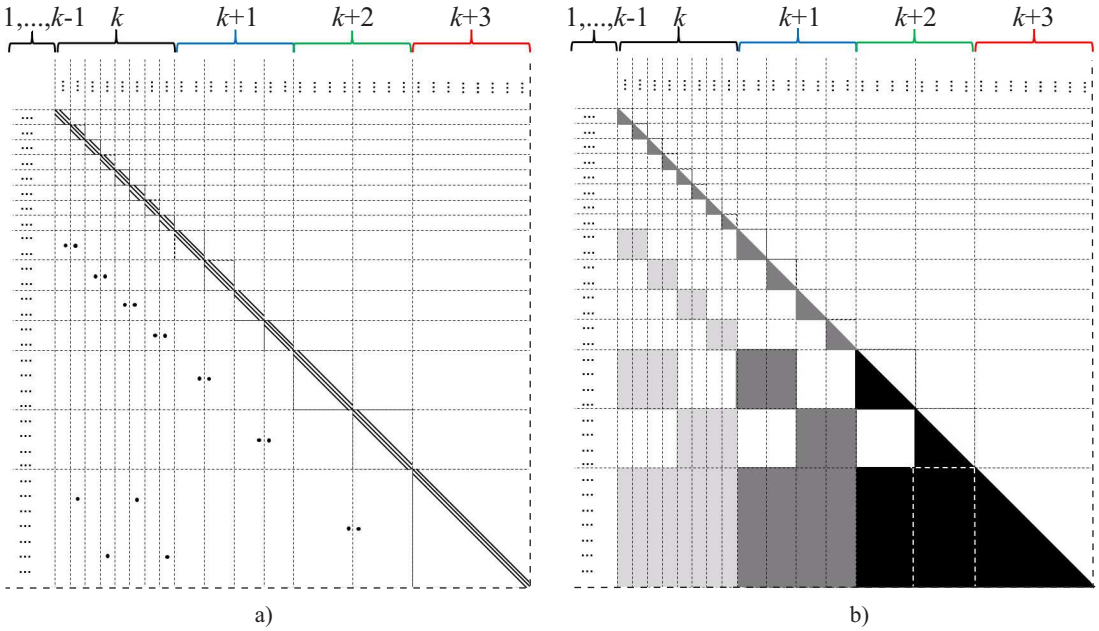

Рис. 6. Портреты матриц $\widehat{A}$ (a) и L-фактор (b) после выполнения алгоритма ND, число уровней дерева равно $k+3$

Применение алгоритма вложенных сечений позволяет повысить эффективность гауссова исключения на порядок. Если N — общее число неизвестных, то для двумерного случая объем оперативной памяти P_1 и число арифметических операций Q_1 приблизительно равны $O\big(N\log(N)\big)$ и $O\big(N^{3/2}\big)$, а в трехмерном — $O\big(N^{4/3}\big)$ и $O\big(N^2\big)$ соответственно [10]. Таким образом, в двумерном случае прямой метод на основе LU-разложения является конкурентоспособным с итерационными. Для трехмерных задач это до сих пор открытый вопрос, и дальнейшее повышение эффективности алгоритма гауссова исключения связано с

возможностью использования техники аппроксимации матрицами малого ранга, а также HSS-формата применительно к плотным внедиагональным и диагональным блокам в L-множителях.

2. Аппроксимация нижнетреугольной матрицы L блоками с малым рангом. Дальнейшее повышение эффективности алгоритма гауссова исключения основано на использовании техники аппроксимации блоками малого ранга (low-rank), а также блочного HSS-формата применительно к плотным недиагональным и диагональным блокам в L-множителях. Таким образом строится блочно-малоранговое приближение к исходной матрице L с любой наперед заданной точностью.

2.1. Малоранговая аппроксимация плотных недиагональных блоков матрицы L. Отправной точкой для исследования аппроксимации матрицами малого ранга является сингулярное разложение матрицы (SVD-разложение), которое покажет, можно ли матрицу B аппроксимировать матрицей малого ранга $\widetilde{B}$ и в каком виде можно представить эту аппроксимацию. Использование SVD-анализа удобно для теоретических представлений и оценок. В практической же реализации наиболее эффективными способами нахождения малоранговой аппроксимации являются QR-, а также АКА-алгоритмы. Впервые метод АКА был предложен в работах Е. Е. Тыртышникова [21] и М. Бебендорфа [5].

Для произвольной плотной матрицы $B \in R^{m\times n}$ (пусть число строк m больше числа столбцов n) результатом SVD-разложения является произведение:

$$B = UKV^*, \tag{3}$$

где $U = [u_1, \ldots, u_n] \in R^{m\times n}$ и $V = [v_1, \ldots, v_n] \in R^{n\times n}$ — ортогональные матрицы, $\{u_i\}_{i=1}^n$ — левые сингулярные векторы, $\{v_i\}_{i=1}^n$ — правые сингулярные векторы, а матрица $K = \{\sigma_i\}_{i=1}^n$ — диагональная с сингулярными числами σ_i на диагонали, упорядоченными по убыванию.

Кроме того, SVD-разложение можно представить в виде суммы матричного произведения левых и правых сингулярных векторов, а также сингулярных чисел σ_i:

$$B = \sum_{i=1}^{n} \sigma_i u_i v_i^* = \sum_{i=1}^{n} \sigma_i [u_i][v_i^*]. \tag{4}$$

В дальнейших рассуждениях общее количество сингулярных чисел (нормированных на старшее сингулярное число σ_1), бóльших либо равных заданному порогу ε, обозначается через $\mathrm{rank}_\varepsilon$ (ε-ранг матрицы A). Поэтому для любого заданного ε сумма первых $k = \mathrm{rank}_\varepsilon$ элементов ряда (4) является матрица $\widetilde{B} \in R^{m\times n}$ ранга k. Несложно заметить, что $\dfrac{\|B - \widetilde{B}\|}{\|B\|} < \varepsilon$, где в качестве нормы используется евклидова норма (максимальное сингулярное число матрицы).

В данном случае матрица $\widetilde{B}$ является аппроксимацией малым рангом матрицы B. Преобразованием суммы (4) в произведение двух матриц малоранговую аппроксимацию можно представить в виде

$$\widetilde{B} = \widetilde{U}\widetilde{V}^*, \quad \widetilde{U} = [u_1, \ldots u_k] \in R^{m\times k}, \quad \widetilde{V} = [v_1\sigma_1, \ldots v_k\sigma_k] \in R^{n\times k}. \tag{5}$$

В практических применениях вместо матрицы B используются сохраненные указанные матрицы $\widetilde{U}$ и $\widetilde{V}$.

Для хранения $\widetilde{U}$ и $\widetilde{V}$ число элементов составляет $(m+n)k$, для хранения B — mn, а коэффициент сжатия $KC = \dfrac{k(m+n)}{mn}$.

Вместо матрично-векторного произведения $y = Bx$ используется два последовательных умножения:

$$z = \widetilde{V}^* x, \quad y = \widetilde{U} z, \quad x \in R^n, \quad y \in R^m, \quad z \in R^k. \tag{6}$$

Число арифметических операций также равно $(m+n)k$ вместо mn для умножения B, а коэффициент сжатия KC составляет $\dfrac{k(m+n)}{mn}$. Если ε-rank равен полному рангу матрицы B, то коэффициенты компрессии и ускорения больше единицы, и в данном случае аппроксимация неэффективна. Однако если $k < \dfrac{mn}{m+n}$, особенно когда сингулярные числа матрицы убывают достаточно быстро, то малоранговая аппроксимация высокоэффективна: матрица B аппроксимируется матрицей малого ранга $\widetilde{B} = \widetilde{U}\widetilde{V}^*$.

Свойство малоранговой аппроксимации внедиагональных блоков матрицы L, как и дополнения Шура, доказано для различных классов задач в работе [6], широко используется и в других работах. Для трехмерной задачи Пуассона с нулевыми граничными условиями Дирихле это свойство подтверждается численными экспериментами. В частности, на рис. 7 в логарифмической шкале показано экспоненциальное убывание сингулярных чисел для недиагональных блоков, соответствующих связям предпоследнего и последнего сеточного разделителя.

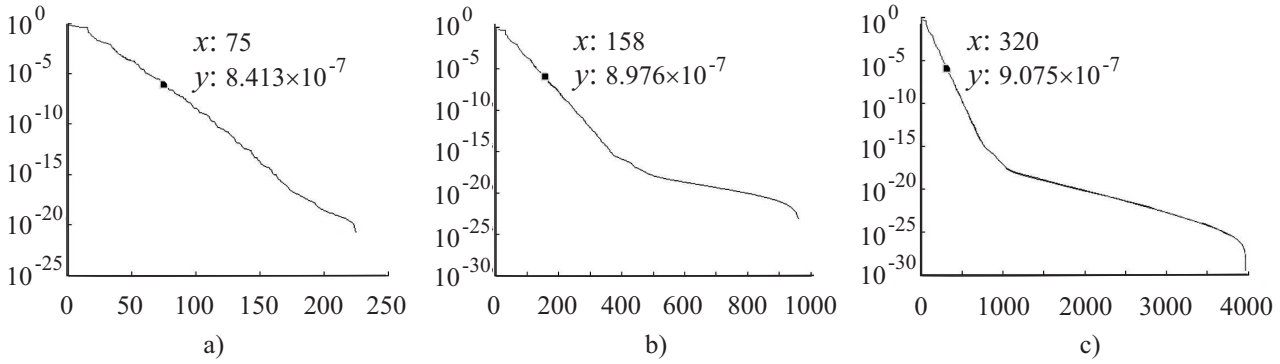

Рис. 7. Сингулярные числа недиагональных блоков матрицы L для связей предпоследнего и последнего сеточного разделителя на примере трех расчетных областей: a) 31^3, b) 63^3 и c) 127^3

Так как разделитель в трехмерном случае является плоскостью, то его размерность растет квадратично: $O\big(n^2\big)$, где n — число точек по каждому направлению. Значение $\mathrm{rank}_\varepsilon$ растет линейно: $O(n)$. Это также подтверждает идею, упомянутую в работах [23, 27].

При оптимизации метода гауссова исключения для хранения недиагональных блоков матрицы L, обладающих коэффициентом сжатия меньшим 1, используется их малоранговая аппроксимация, т.е. сохраненные матрицы $\widetilde{U}$ и $\widetilde{V}$, а все необходимые арифметические действия с недиагональным блоком заменяются на операции с этими двумя матрицами.

2.2. Использование HSS-формата для хранения диагональных блоков матрицы L. Диагональные блоки матрицы L не обладают свойством малоранговой аппроксимации. В противном случае она содержала бы близкие к нулю сингулярные числа и исходная система (1) была бы близка к вырожденной, однако это не так. Для диагональных блоков эффективно применяется техника HSS-аппроксимации и их представление в HSS-формате [8].

Рассмотрим произвольный диагональный блок F матрицы L, соответствующий некоторому разделителю Ω_i расчетной области.

Поскольку L — нижнетреугольная матрица, предполагается, что блок $F \in R^{n\times n}$ симметричный с отражением нижней части на верхнюю. Кроме того, без ограничения общности, допускается, что $n = 2^p$.

Опишем алгоритм представления блока F в HSS-формате последовательно для разной глубины разбиения матрицы (l-уровень — параметр HSS-формата).

1. *Уровень* $l = 1$. Считается, что матрица уже в формате HSS в виде плотного блока $D = F$.

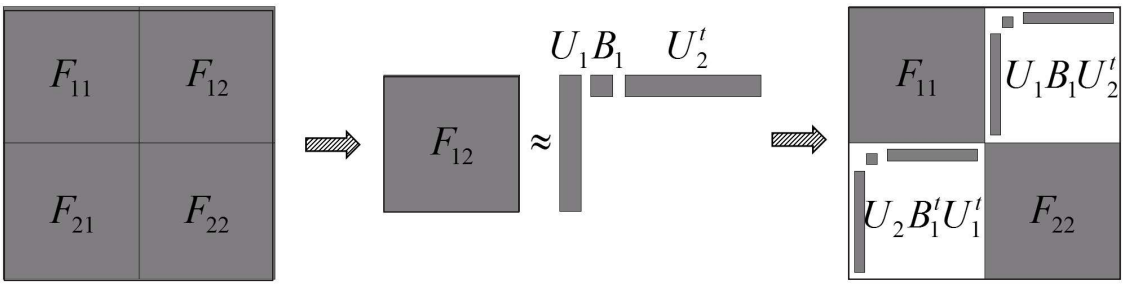

Рис. 8. Представление матрицы F в HSS-формате с глубиной рекурсивного разбиения l, равной 2

2. *Уровень* $l = 2$. Матрица F представляется в блочном виде 2×2 (рис. 8, слева). Недиагональный блок F_{12} может быть аппроксимирован матрицей малого ранга. Этот блок представляется в виде произведения ортогональной матрицы U_1, квадратной матрицы B_1 и ортогональной матрицы U_2 (рис. 8, центр). Это может быть выполнено с применением SVD- или QR-разложения. Как упоминалось выше, более эффективным способом является применение АКА-аппроксимации (5) с последующей ортогонализацией векторов $\widetilde{U}$ и $\widetilde{V}$. Здесь и далее считается, что $\mathrm{rank}_\varepsilon = k$, а диагональные блоки F_{ii} переобозначаются через D_i.

3. *Уровень* $l = 3$. Диагональные блоки F_{ii} представляются в блочном виде 2×2. Полученные внедиагональные блоки обладают свойством малоранговой аппроксимации. С учетом новой нумерации блоков матрица F будет иметь вид, представленный на рис. 9. Такая, на первый взгляд, сложная нумерация является естественной, если матрицу F представить в виде HSS-дерева T_H. Каждому листу j этого дерева

соответствует диагональный блок F_{jj}, а каждой промежуточной вершине i соответствует диагональный блок F_{ii}, который состоит из блоков $F_{c_1c_1}$, $F_{c_2c_2}$, $F_{c_1c_2}$, $F^T_{c_1c_2}$, где c_1 и c_2 — номера вершин левого и правого поддерева вершины i в дереве T_H (рис. 10). Из такой нумерации получаем $n_i = n_{c_1} + n_{c_2}$, где n_i, n_{c_1} и n_{c_2} — размерности блоков F_{ii}, $F_{c_1c_1}$ и $F_{c_2c_2}$ соответственно.

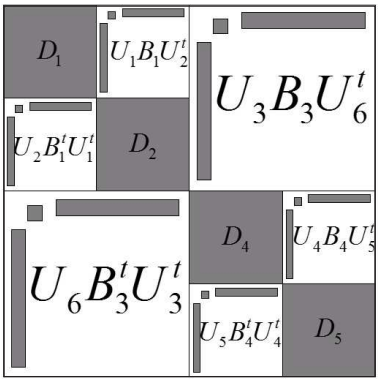

Рис. 9. Представление матрицы F в HSS-формате с глубиной рекурсивного разбиения l, равной 3

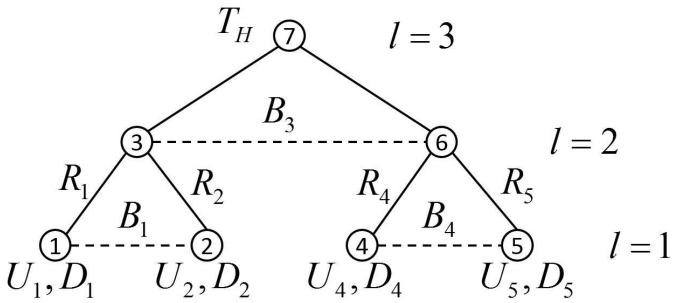

Рис. 10. Древовидное представление HSS-формата

Далее вводятся следующие определения:

— HSS-строка F_i^- — блочная строка матрицы F без диагонального блока F_{ii}: $F_i^- \in R^{n_i \times (n-n_i)}$;

— HSS-столбец $F_i^|$ — блочный столбец матрицы F без диагонального блока F_{ii}: $F_i^| \in R^{(n-n_i)\times n_i}$;

— блок F_{ij} — пересечение строки i со столбцом j: $F_{ij} = F_i^- \cap F_j^|$.

Нумерация для матриц U_l и B_l вводится так, что любой блок F_{ij} представляется в виде $F_{ij} = U_iB_iU_j^T$, $U_i \in R^{n_i\times k}$, $B_i \in R^{k\times k}$, $U_j \in R^{n_j\times k}$ (рис. 10). Такое представление не нарушает свойства симметрии F, т.е. $F_{ij} = F_{ji}^T$.

Одним из замечательных свойств HSS-аппроксимации является то, что матрицы U_3 и U_6 не хранятся в явном виде, а выражаются посредством матриц перехода R_1, R_2 и R_4, R_6, что позволяет дополнительно экономить память:

$$U_3 = \begin{pmatrix} U_1R_1 \\ U_2R_2 \end{pmatrix}, \quad U_6 = \begin{pmatrix} U_4R_4 \\ U_5R_5 \end{pmatrix}, \quad R_1,\, R_2,\, R_4,\, R_5 \in R^{k\times k}. \tag{7}$$

4. *Уровень* $l \geqslant 4$. Диагональные блоки D_1–D_5 разбиваются на (2×2)-подблоки, и выполняются аналогичные предыдущему пункту действия.

В результате получается HSS-представление, состоящее из матриц U_i, R_i, B_i, D_i, в котором матрицы U_i и D_i хранятся только для листьев HSS-дерева T_H; B_i — для левых соседей, а R_i — для всех узлов, кроме узлов из двух верхних уровней дерева T_H (рис. 10).

Рекурсивный процесс можно выполнять, пока размер блоков не станет равным 1. Однако из общих соображений необходимо остановить разбиение, если значение n_i стало соизмеримым с ε-рангом k недиагональных блоков. Максимальная эффективность HSS-сжатия равна $O(nk)$ и достигается при $l = \log_2(n-k)$.

Следует отметить, что произвольный блок F_{ij} несложно выражается через матрицы U_l, R_l, B_l посредством древовидного представления (10) как произведение указанных матриц, расположенных на кратчайшем пути от вершины i до j. Необходимо учесть то, что при направлении вниз по дереву, а также справа налево используется транспонирование R^T и B^T. Произведение матриц всегда начинается с U_i, а заканчивается транспонированием U_j^T. Пример для листьев дерева T_H: $F_{12} = U_1B_1U_2^T$, $F_{21} = U_2B_1^TU_1^T$, $F_{14} = U_1R_1B_3R_4^TU_4^T$.

Пример для двух внутренних узлов дерева T_H: $F_{36} = U_3B_3U_6^T$.

Для выполнения математических операций разработаны специализированные алгоритмы, применяющие сохраненные в HSS-формате матрицы. Эти алгоритмы не используют явного представления (вычисления) блоков F_{ij} и всей матрицы F, а также матриц U_i для промежуточных узлов i дерева T_H.

Алгоритм конвертации плотной матрицы в HSS-формат в различных вариантах описан во многих работах, например [7, 25]. Указанные алгоритмы основаны на предположении о том, что заранее известно число уровней l для HSS-структуры. Построение HSS-структуры начинается снизу вверх, т.е. для листьев $i \in T_H$ задаются диагональные блоки $D_i = F_{ii}$, вычисляются матрицы U_i, R_i и B_i. Далее для второго уровня и т.д. до предпоследнего от вершины дерева вычисляются матрицы R_i и B_i.

Временны́е затраты на указанные алгоритмы составляют $O(kn^2)$ (табл. 1).

Таблица 1
Оценка памяти HSS-формата и производительности HSS-алгоритмов

Формат хранения	Построение HSS-формата	Нахождение разложения $F = LL^T$	Обращение системы $LL^Tx = b$	Объем памяти для хранения L
Плотный	—	$O(n^3)$	$O(n^2)$	$O(n^2)$
H-матрица	$O(kn^2)$	$O(n^2)$	$O(kn)$	$O(kn)$

Алгоритм умножения H-матрицы на вектор также хорошо известен и описан в различных работах, например [8, 12]. Не вдаваясь в подробности реализации, следует обратить внимание на число арифметических операций: $O(kn)$ вместо $O(n^2)$ для плотной матрицы.

Для построения LL^T-разложения H-матрицы существуют два подхода. Первый (Fast Factorization) основан на использовании древовидной H-структуры T_H; его производительность (число арифметических операций) равна $O(n^2)$ [7].

Второй подход (Super Fast Factorization) основан на предварительном уменьшении размерности исходной матрицы путем умножения H-матрицы F_H слева и справа на блочно-диагональную с ортогональными матрицами S_i на диагонали: $\widehat{F} = SFS^T$. Последние столбцы матрицы S_i состоят из столбцов U_i, что позволяет разбить полученную после умножения матрицу на независимые, для процесса факторизации, блоки. Производительность алгоритма выше, чем у первого, и равна $O(k^2n)$. Однако необходимо хранить дополнительные ортогональные блоки S_i, что нежелательно, если есть ограничения по памяти.

Два разных алгоритма дают разные представления $L_HL_H^T$-разложения, что, в свою очередь, порождает два разных алгоритма для их обращения. Для первого алгоритма структура L_H уже представлена в формате H-матриц, в которой, предполагается, что верхняя часть равна нулю. Для второго структура L_H должна дополнительно содержать ортогональные матрицы S_i.

Временны́е затраты на алгоритмы, использующие HSS-формат, а также затраты по памяти для хранения H-матриц, представлены в табл. 1.

Для получения $L_HL_H^T$-разложения диагональных блоков в матрице L используется первый алгоритм (Fast Factorization). Такой выбор обусловлен тем, что указанный алгоритм более экономичный по использованию памяти. То, что он по временны́м затратам уступает второму алгоритму, не существенно, так как построение HSS-формата, а также выполнение расчетов дополнения Шура при выполнении LL^T-разложения исходной матрицы A занимают значительно больше времени, чем HSS-факторизация диагональных блоков матрицы L.

3. Алгоритм метода Гаусса с использованием техники малоранговой аппроксимации и HSS-формата. Как упоминалось выше, решение СЛАУ прямым методом Гаусса состоит из двух этапов: нахождения нижнетреугольной L (симметричный случай) и решения системы $LL^Tx = b$ — этап решения.

При дальнейшем описании алгоритмов используется блочное представление матриц A и L. Эффективность блочных методов обусловлена спецификой машинной архитектуры (ограниченной размерностью памяти быстрого доступа — *cash*, возможностью процессора выполнять векторные операции, а также пропускной способностью шины память–процессор).

При использовании HSS-техники часть недиагональных или диагональных плотных блоков представляется в виде малоранговой или HSS-аппроксимации. Соответствующие им арифметические операции выполняются в приближенной малоранговой/HSS-арифметике, описанной в предыдущем разделе. Результатом HSS-аппроксимации является малоранговое приближение $\widetilde{L}$ для матрицы L. В случае недостаточной точности решения системы $\widetilde{L}\widetilde{L}^Tx = b$ используется третий этап — итерационное уточнение.

3.1. Построение нижнетреугольной матрицы L. Существуют различные подходы организации LL^T-разложения. Остановимся на алгоритме *отложенной модификации* и его блочной модификации — алгоритме Донгарры–Айзенштата [3]. Преимущество использования алгоритма *отложенной модификации* будет видно далее.

Описание алгоритма. Разложение LL^T производим по столбцам. Полученные в процессе разложенные столбцы и оставшаяся часть матрицы хранятся как одно целое, т.е. изначальная матрица A трансформируется в L.

Для примера рассматривается организация матрицы L (см. 8.в) для плотной симметричной матрицы

A, разбитой на блоки 4×4:

$$\begin{pmatrix} A_{11} & A_{21}^T & A_{31}^T & A_{41}^T \\ A_{21} & A_{22} & A_{32}^T & A_{42}^T \\ A_{31} & A_{32} & A_{33} & A_{43}^T \\ A_{41} & A_{42} & A_{43} & A_{44} \end{pmatrix} \Rightarrow \begin{pmatrix} L_{11} & L_{21}^T & L_{31}^T & L_{41}^T \\ L_{21} & S_{22} & S_{32}^T & S_{42}^T \\ L_{31} & S_{32} & A_{33} & A_{43}^T \\ L_{41} & S_{42} & A_{43} & A_{44} \end{pmatrix} \Rightarrow \begin{pmatrix} L_{11} & L_{21}^T & L_{31}^T & L_{41}^T \\ L_{21} & L_{22} & L_{32}^T & L_{42}^T \\ L_{31} & L_{32} & L_{33} & L_{43}^T \\ L_{41} & L_{42} & L_{43} & L_{44} \end{pmatrix} \tag{8}$$

а б в

1. 1-й столбец. Гауссово исключение для блока A_{11}: $L_{11}L_{11}^T = A_{11}$. Для оставшейся части столбца получаем $L_{i1} = A_{i1}L_{11}^{T^{-1}}$, $i = 2, 3, 4$.

2. 2-й столбец. Второй столбец матрицы (8.в) — это преобразованный 2-й столбец матрицы (8.а): $S_{i2} = A_{i2} - L_{i2}L_{21}^T$, $i = 2, 3, 4$, где блоки S_{i2} называют *дополнением Шура* для блоков A_{i2} (8.б). Второй столбец для L получается как гауссово исключение для S_{22}: $L_{22}L_{22}^T = S_{22}$ и формирование недиагональных блоков $L_{i2} = S_{i2}L_{22}^{T^{-1}}$, $i = 3, 4$.

После вычисления последнего столбца для L получаем LL^T-разложение исходной матрицы A (8.в).

В случае матрицы произвольного размера $m \times m$ считается, что первые i столбцов разложены. Дополнение Шура вычисляется по формуле

$$S_{ji+1} = A_{ji+1} - \sum_{k=1}^{i} L_{jk}L_{i+1,k}^T, \quad j = i+1, \ldots, m. \tag{9}$$

Результат наглядно представляется в матричном виде (10). Гауссово исключение вычисляется по формуле (11). Гауссово исключение для столбцов $i+1, \ldots, m$ вычисляется аналогичным образом. В результате получается нижнетреугольная плотная матрица L в блочном формате.

Предложенный алгоритм несложно распространяется на разреженные матрицы, полученные в результате применения алгоритма перенумерации элементов (вложенных сечений). Следует отметить, что блочные операции в формулах (9) и (11) выполняются только для ненулевых блоков, достигая тем самым экономии вычислений.

Как уже упоминалось выше, плотные блоки матрицы L могут быть аппроксимированы матрицами малого ранга или H-матрицами со значительным уменьшением объема использованной машинной памяти. Однако даже при таком подходе необходимо заранее вычислить и хранить в памяти всю матрицу L с плотными блоками (6), что невозможно при решении указанных задач.

Чтобы устранить этот недостаток, предлагается использование многоуровневого гауссова исключения [25], в котором упаковка матрицы L в HSS-формате производится в процессе вычисления самой матрицы L:

$$\begin{pmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \ldots & L_{kk} & \ldots & L_{ik}^T & L_{i+1,k}^T & L_{i+2,k}^T & \ldots & L_{jk}^T & \ldots & L_{mk}^T \\ \ldots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \ldots & L_{ik} & \ldots & L_{ii} & L_{i+1,i+1}^T & L_{i+2,i}^T & \ldots & L_{ji}^T & \ldots & L_{mi}^T \\ \ldots & L_{i+1,k} & \ldots & L_{i+1,i} & S_{i+1,i+1} & S_{i+2,i+1}^T & \ldots & S_{j,i+1}^T & \ldots & S_{m,i+1}^T \\ \ldots & L_{i+2,k} & \ldots & L_{i+2,i} & S_{i+2,i+1} & A_{i+2,i+2} & \ldots & A_{j,i+2}^T & \ldots & A_{m,i+2}^T \\ \ldots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \ldots & L_{jk} & \ldots & L_{ji} & S_{j,i+1} & A_{j,i+2} & \ldots & A_{jj} & \ldots & A_{mj}^T \\ \ldots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \ldots & L_{mk} & \ldots & L_{mi} & S_{m,i+1} & A_{m,i+2} & \ldots & A_{mj} & \ldots & A_{mm} \end{pmatrix} \tag{10}$$

$$L_{j,i+1} = S_{j,i+1}L_{i+1,i+1}^{T^{-1}}, \quad j = i+2, \ldots, m. \tag{11}$$

Описание алгоритма. В процессе формирования L-матрицы первые блочные столбцы находятся приведенным выше алгоритмом отложенной модификации Донгарры–Айзенштата (9) и (11). Начиная со

столбца, соответствующего определенному заранее уровню дерева исключений (5), выполняем малоранговую/HSS-аппроксимацию блоков.

1. Для текущего разреженного столбца A_i выполняется его модификация традиционным способом (9). Если в операциях матричного умножения участвуют блоки L_{ij}, представленные в виде малоранговой аппроксимации $L_{ij} = U_{ij}V_{ij}^T$, то эти операции производятся без промежуточного построения полного блока L_{ij}, тем самым экономя на числе арифметических операций (6). Следует отметить тот факт, что несмотря на сильно разреженную структуру матрицы A, заполняемость L-матрицы увеличивается от столбца к столбцу. Таким образом, далее считается, что дополнение Шура состоит как из нулевых, так и из плотных блоков.

2. Диагональный блок S_{ii} аппроксимируется H-матрицами $\widetilde{S}_{ii}$, недиагональные — матрицами малого ранга $\widetilde{S}_{ji} = U_{ji}V_{ji}^{\mathrm{T}}$.

3. Построение гауссова исключения (H-факторизация) диагонального блока производится с применением H-алгоритма быстрой факторизации (Fast Factorization, [7]): $\widetilde{S}_{ii} = \widetilde{L}_i\widetilde{L}_i^T$.

4. Факторизация недиагональных блоков основана на алгоритме (11): $L_{ji} = S_{ji}L_{ii}^{T^{-1}}$. Учитывая малоранговую аппроксимацию для S_{ji}, аппроксимация $\widetilde{L}_{ji} = U_{ji}\widehat{V}_{ji}^T$ вычисляется по формуле

$$\widehat{V}_{ji} = \widetilde{L}_i^{-1}V_{ji}, \quad j = i+1, \ldots, m. \tag{12}$$

После 4-го шага находится малоранговое/HSS-представление $\widetilde{L}_i$ для столбца L_i, при формировании которого экономия памяти достигается за счет ухода от явного представления блоков $\widetilde{L}_{ji}$ в плотном виде, а экономия по арифметическим вычислениям — за счет выполнения малоранговых/HSS-матричных операций умножения, факторизации и решения.

Для дальнейшей оптимизации алгоритма используется адаптивная малоранговая/HSS-аппроксимация дополнений Шура. Для выбранного столбца производится аппроксимация недиагональных блоков S_{ij} и для каждого из них оценивается эффективность его хранения в упакованном виде по критерию, описанному выше: $\dfrac{k(m+n)}{mn} < 1$, где $k = \mathrm{rank}_\varepsilon$. Предложенная оценка уже фактически является выполнением малоранговой нетрудозатратной аппроксимации, так как основана на использовании AKA-техники, что высокоэффективно по сравнению с SVD- и QR-алгоритмами [16]. Для диагональных блоков необходимость их представления в виде H-матриц зависит как от $\mathrm{rank}_\varepsilon$, так и от размера блока и оценивается экспериментально.

Как результат, столбец S_i имеет смешанную структуру и, в отличие от известных в литературе методов [26], состоит как из плотных, так и из аппроксимируемых блоков. В силу этого эффективно используется малоранговая аппроксимация для больших блоков, а малые блоки хранятся в плотном формате (рис. 11).

Результатом использования алгоритма на основе метода Гаусса, техники аппроксимации малого ранга и HSS-формата является $L_HL_H^T$-разложение, пример структуры которого показан на рис. 12.

3.2. Прямая и обратная подстановки. Решение системы $L_HL_H^Tx = b$ состоит из прямой (решение $L_Hy = b$) и обратной (решение системы $L_H^Tx = y$) подстановок. Для примера рассматривается процесс прямой и обратной подстановки для плотной нижнетреугольной блочной матрицы L размером $m \times m$. В конце этого подраздела показывается, как применить этот процесс к матрице L_H.

Прямая подстановка. Решение системы $Ly = b$ находится поочередным (по столбцам) занулением недиагональных элементов матрицы L путем применения к ней, а также к вектору правой части b, цепочки преобразований $\overrightarrow{P} = \overrightarrow{P_m}\ldots\overrightarrow{P_1}$. Полученная в результате система с единичной матрицей имеет решение, равное вектору правой части.

Для исключения первого столбца преобразование $\overrightarrow{P_1}$ имеет вид: $\overrightarrow{P_1} = \begin{pmatrix} L_{11}^{-1} & 0 \ldots 0 \\ -L_{21}L_{11}^{-1} & 1 \quad 0 \\ \vdots & \ddots \\ -L_{m1}L_{11}^{-1} & 0 \quad 1 \end{pmatrix}$.

Результатом поочередного применения $\overrightarrow{P_{i-1}}\ldots\overrightarrow{P_1}$ преобразований является система линейных уравнений $L^{(i)}x = b^{(i)}$ с матрицей $L^{(i)}$ и вектором правой части $b^{(i)} = \overrightarrow{P_{i-1}}\ldots\overrightarrow{P_1}b$. Исключение i-го столбца

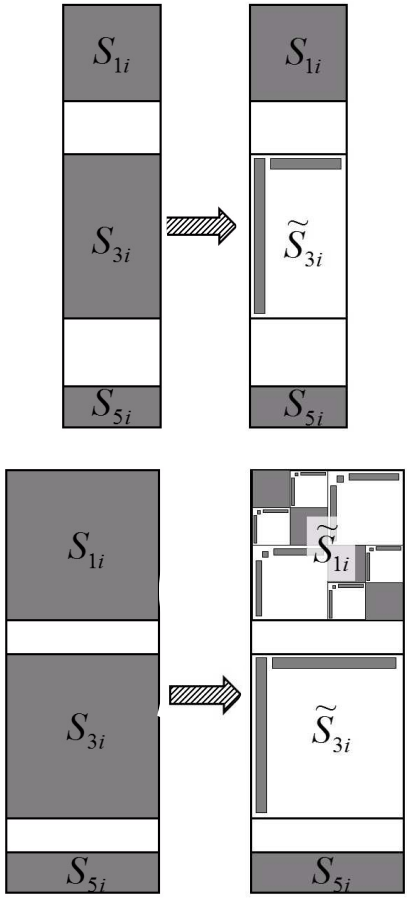

Рис. 11. Пример адаптивной малоранговой/HSS-аппроксимации для двух разных столбцов дополнения Шура

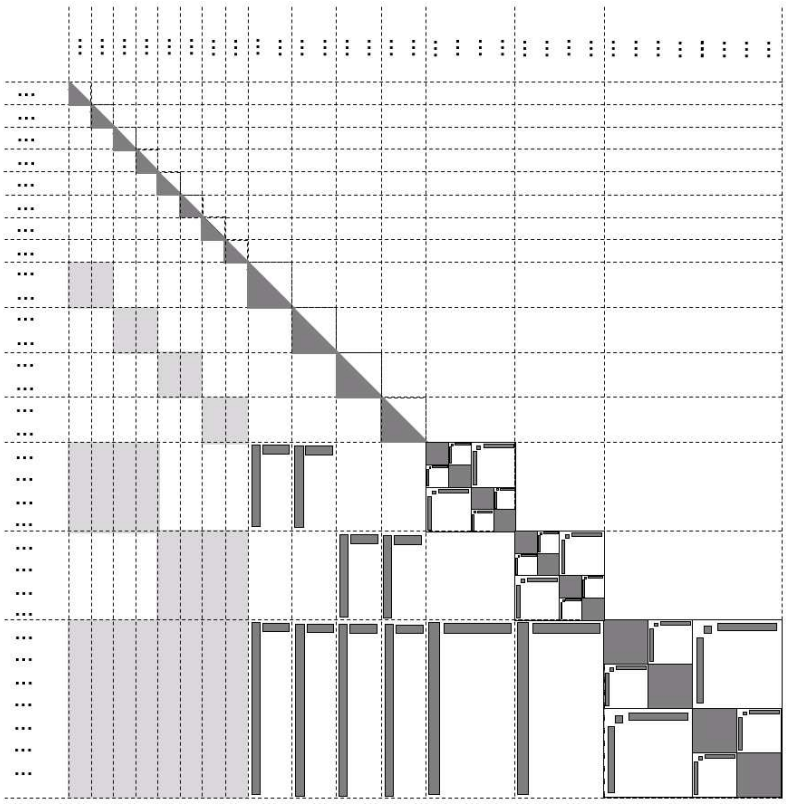

Рис. 12. Пример L_H матрицы

матрицы L показано на примере преобразования $\overrightarrow{P_i}$:

$$\overrightarrow{P_i} = \begin{pmatrix} 1 & 0 & & & & 0 \\ 0 & \ddots & & & & \\ & \ddots & 1 & & & \\ 0 & \dots & 0 & L_{ii}^{-1} & & \\ 0 & \dots & 0 & -\left(L_{i+1,i}L_{ii}^{-1}\right) & 1 & \\ \vdots & & \vdots & \vdots & \ddots & 0 \\ 0 & \dots & 0 & -(L_{mi}L_{ii}^{-1}) & 0 \dots & 1 \end{pmatrix}, \quad L^{(i)} = \begin{pmatrix} 1 & 0 & & & & 0 \\ 0 & \ddots & & & & \\ & \ddots & 1 & & & \\ 0 & \dots & 0 & L_{ii} & & \\ 0 & \dots & 0 & L_{i+1,i} & L_{i+1,i+1} & \\ \vdots & & \vdots & \vdots & \ddots & 0 \\ 0 & \dots & 0 & L_{mi} & \dots & \dots L_{mm} \end{pmatrix}. \tag{13}$$

Действие $\overrightarrow{P_i}$ на вектор $b^{(i)}$ записывается поэлементно: $b_i^{(i+1)} = L_{ii}^{-1}b_i^{(i)}$, $b_j^{(i+1)} = b_j^{(i)} - \left(L_{i+1,i}L_{ii}^{-1}\right)b_i^{(i)}$, $j = i+1, \dots, m$.

Решение системы $Ly = b$ является результатом действия на ее левую и правую части преобразования $\overrightarrow{P} = \overrightarrow{P_m} \dots \overrightarrow{P_1}$, т.е. $y = b^{(m)} = \overrightarrow{P}b$. Алгоритм действия $\overrightarrow{P}b$ записывается пошагово:

1. $i = 1, \dots, m$ (цикл по столбцам матрицы L).
   a) $b_i = L_{ii}^{-1}b_i$ (нормировка к единице диагонального элемента ii матрицы L);
   b) $j = i+1, \dots, m$ (цикл по строкам матрицы L, поочередное зануление недиагональных элементов в столбце i): $b_j = b_j - L_{ji}b_i$.
2. Искомое решение: $y = b$.

Обратная подстановка. Идея обратной подстановки $L^T x = y$ схожа с идеей прямой подстановки: поочередное зануление недиагональных элементов используемой матрицы. Главное отличие в том, что решается система с верхнетреугольной (транспонированной) матрицей L^T, и зануление элементов происходит по строкам: начинается с последней строки m и заканчивается первой. Для исключения последней строки преобразование $\overleftarrow{P_m}$ принимает диагональный вид: $\overleftarrow{P_m} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & 1 & 0 \\ 0 & 0 & 0 & L_{mm}^{T^{-1}} \end{pmatrix}$.

Элементы i-й строки матрицы $L^{T^{(i)}}$ зануляются преобразованием $\overleftarrow{P_i}$:

$$\overleftarrow{P_i} = \begin{pmatrix} 1 & 0 & \dots & \dots & & \dots & \dots & 0 \\ 0 & \ddots & \ddots & & & & & \vdots \\ \vdots & \ddots & 1 & 0 & & \dots & \dots & 0 \\ \vdots & & \ddots & L_{ii}^{T^{-1}} & -\left(L_{ii}^{T^{-1}} L_{i+1,i}\right) & \dots & -\left(L_{ii}^{T^{-1}} L_{mi}\right) \\ \vdots & & & \ddots & 1 & 0 & 0 \\ \vdots & & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & \dots & \dots & 0 & 1 \end{pmatrix}, \quad L^{(i)} = \begin{pmatrix} L_{11}^T & \dots & \dots & \dots & \dots & L_{m1}^T \\ 0 & \ddots & & & & \vdots \\ \vdots & \ddots & L_{ii}^T & L_{i+1,i}^T & \dots & L_{mi}^T \\ \vdots & & \ddots & 1 & 0 & 0 \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & \dots & 0 & 1 \end{pmatrix} \tag{14}$$

Действие преобразования $\overleftarrow{P_i}$ на правую часть $y^{(i)}$ изменяет только элемент $y_i^{(i-1)}$, который и определяет i-ю компоненту вектора решения исходной системы $L^T x = y$: $y_i^{(i-1)} = L_{ii}^{T^{-1}} \left[y_i^{(i)} - \sum\limits_{j=m}^{i+1} L_{ij}^T y_i^{(i)} \right]$.

И наконец, результатом преобразования $\overleftarrow{P} = \overleftarrow{P_1} \dots \overleftarrow{P_m}$ является решение системы $L^T x = y$, где $x = y^{(1)}$.

Таким образом, алгоритм обратной подстановки состоит из двух шагов:

1. $i = m, \dots, 1$ (цикл по строкам матрицы L^T):

а) $j = m, \dots, i+1$ (цикл по столбцам матрицы L^T, поочередное зануление недиагональных элементов в строке i): $y_i = y_i - L_{ji}^T y_j$;

б) $y_i = \left(L_{ii}^T\right)^{-1} y_i$ (нормировка к единице диагонального элемента ii матрицы L^T).

2. Искомое решение: $x = y$.

Основные арифметические операции в алгоритмах прямой и обратной подстановок — это умножение прямоугольной плотной матрицы L_{ij} на вектор и обращение нижнетреугольной матрицы L_{ii}. Матрица L_H, в отличие от L, содержит плотные блоки, записанные в малоранговой структуре, а также диагональные HSS-блоки. Для такой матрицы L_H описанные выше алгоритмы модифицируются путем использования малоранговой и HSS-арифметики при умножении недиагональных блоков и обращении диагональных.

3.3. Итерационное уточнение. Получаемое на основе метода Гаусса решение с аппроксимацией матрицами малого ранга не является точным решением системы (1) и содержит как погрешности машинного округления при выполнении операций над числами с плавающей точкой, так и ошибки малоранговой аппроксимации L-множителей. Однако погрешности машинного округления легко уменьшаются увеличением количества разрядов для хранения чисел с плавающей точкой. Наиболее распространенными способами хранения являются *double precision* и *quad precision*, соответствующие 8 и 16 байтам на одно число. В программной реализации алгоритма используется *double precision* представление как самый распространенный способ хранения чисел, высокая производительность арифметических операций над которыми обусловлена аппаратной поддержкой современными процессорами, а также графическими сопроцессорами.

Ошибка решения может быть также уменьшена путем увеличения точности малоранговой аппроксимации. Однако при этом уменьшается степень сжатия, увеличиваются затраты по памяти и одновременно затраты на получение LL^T-разложения, обратной и прямой подстановки.

Универсальным способом уменьшения ошибки решения (одновременно машинной погрешности и ошибки аппроксимации) является алгоритм итерационного уточнения, который заключается в следующем.

Приближенное решение x_0 можно представить как точное решение вспомогательной системы

$$\widetilde{A}x = b, \tag{15}$$

где $\widetilde{A}$ — некая аппроксимация матрицы A. Ошибка решения записывается в виде $\delta x_0 = x - x_0$, а невязка как $r_0 = b - Ax_0$. Таким образом, ошибка решения выражается через невязку по следующей формуле:

$$A\delta x_0 = r_0. \tag{16}$$

Для нахождения значения ошибки δx_0 решается система (16) с некоторой погрешностью δx_1. В результате находится точное решение, представимое в виде $x = x_1 + \delta x_1$, где $x_1 = x_0 + \delta x_0$. Если δx_1 достаточно мало, то x_1 используется в качестве точного решения. Иначе заново вычисляется невязка $r_1 = b - Ax_1$ и находится ошибка δx_1 как решение системы $A\delta x_1 = r_1$.

Вычислительный процесс продолжается до тех пор, пока относительная ошибка $\delta x_i/x_0$ не уменьшится до необходимой величины. На практике в качестве критерия остановки используется относительная невязка r_i/b. Формально алгоритм итерационного уточнения записывается в следующем виде:

$x_0 = \widetilde{A}^{-1}b$
$r_0 = b - Ax_0$
while $||r_i||/||b|| < \varepsilon$ **do**
    $r_i = b - Ax_i$
    $\delta x_i = \tilde{A}^{-1}r_i$
    $x_{i+1} = x_i + \delta x_i$

Следует отметить тот факт, что предложенный алгоритм сходится не всегда, а при достаточно точной аппроксимации $\tilde{A}x = b$ исходной задачи $Ax = b$.

4. Численные эксперименты. Численные эксперименты проводились для проверки работоспособности предложенного алгоритма, а также для проверки его производительности и эффективности использования памяти. Расчеты производились на системе: Intel Core i7-3770K, 3.50GHz, 4 вычислительных ядра, RAM 32GB.

Для оценки эффективности алгоритма производилось сравнение по скорости и по памяти с прямым методом решения СЛАУ, реализованным в компоненте PARDISO из высокопроизводительной библиотеки Intel MKL. Кроме того, исследовалась возможность решать максимально большие задачи, которые не могут быть решены прямыми методами по причине большого требования по памяти.

4.1. Зависимость невязки и скорости сходимости решения СЛАУ от точности low-rank-аппроксимации. Первая серия экспериментов проводилась для анализа зависимости точности полученного решения от качества малоранговой аппроксимации. Рассматривалась СЛАУ, полученная в результате аппроксимации уравнения Лапласа в кубе фиксированного размера с нулевыми условиями Дирихле, правая часть уравнения равна константе во всей области. Расчетная сетка — параллелепипедальная, равномерная по всем осям координат с фиксированным числом узлов $n = 140$ по каждой оси. Число расчетных узлов сетки задавалось максимально возможным для решения исходной задачи прямым методом на системе с RAM=32GB без использования жесткого диска (*swap*), а именно $N = 140^3 \sim 2.7 \times 10^9$. Так как аналитическое решение задачи не известно, то в тестах исследуется невязка $r = f - Ax$ численного решения x.

В первом тесте исследуется зависимость значения невязки решения от точности аппроксимации (рис. 13). Для удобства данные по двум осям представлены в логарифмической шкале.

На графике показана прямая зависимость невязки от качества аппроксимации. Следует также отметить, что использование Intel MKL PARDISO дает решение указанной задачи с точностью $\sim 3.2 \times 10^{-12}$ (норма C) и $\sim 3.0 \times 10^{-14}$ (норма L_1). Указанные значения отмечены на графике соответственно горизонтальной сплошной и пунктирной линиями.

Для проверки эффективности итерационного уточнения исследовалась зависимость скорости сходимости $||r_{n+1}||/||r_n||$ от качества аппроксимации (рис. 14), где r_n и r_{n+1} — значения невязки на n-м и $n+1$-м шаге итерационного уточнения.

Начиная с $\varepsilon = 10^{-3}$ процесс довольно быстро сходится. К примеру, для достижения относительной невязки, равной 10^{-12}, необходимо от девяти итераций ($\varepsilon = 10^{-3}$) до одной ($\varepsilon = 10^{-15}$). При этом временны́е затраты при проведении итераций незначительны по сравнению с затратами на факторизацию исходной матрицы.

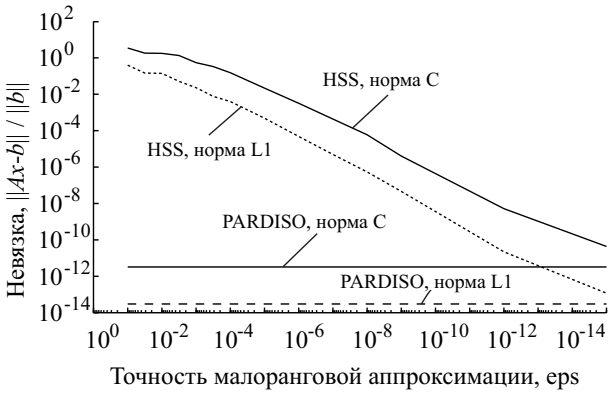

Рис. 13. Зависимость невязки решения в норме C и L_1 от точности малоранговой аппроксимации

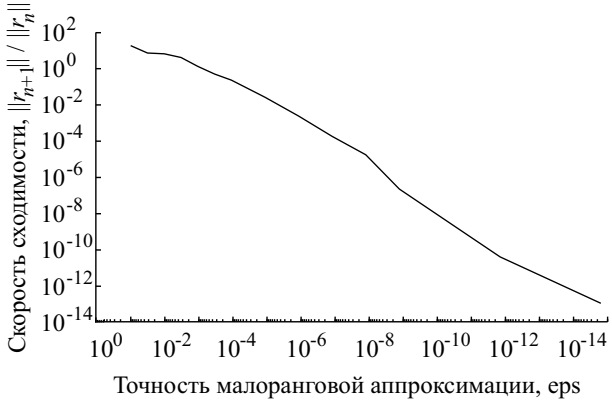

Рис. 14. Зависимость скорости сходимости от точности малоранговой аппроксимации

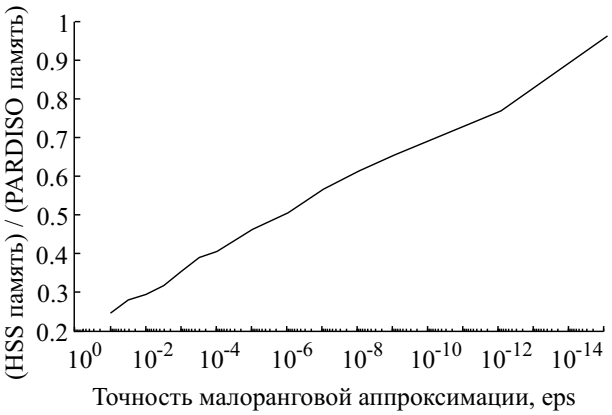

Рис. 15. Коэффициент экономии памяти HSS по сравнению с Intel MKL PARDISO

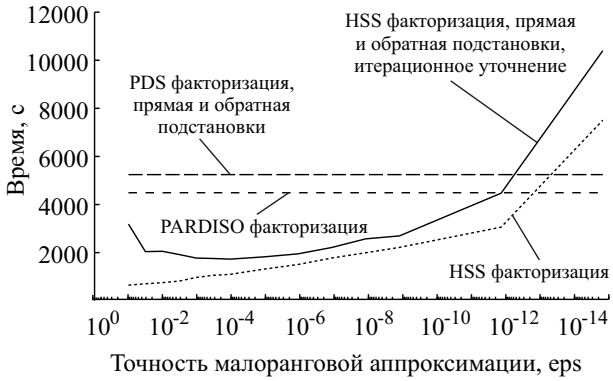

Рис. 16. Временны́е затраты на факторизацию и общее время решения СЛАУ

4.2. Сравнение эффективности HSS-алгоритма с методом Гаусса. Для сравнения эффективности разработанного алгоритма по сравнению с PARDISO проводились тесты на производительность и использование памяти на примере задачи из предыдущего теста. На графике (рис. 15) показана зависимость коэффициента эффективности использования памяти по сравнению с PARDISO для различной степени сжатия. Указанный коэффициент — это отношение размера памяти, необходимой для HSS, к памяти, используемой PARDISO.

На следующем графике (рис. 16) показана зависимость временны́х характеристик HSS-алгоритма от точности аппроксимации ε.

Отметим, что при использовании PARDISO этап факторизации занимает 4491 секунд, а решение системы с 10 правыми частями — 75 секунд.

При уменьшении точности аппроксимации уменьшается время, затраченное на факторизацию, но в то же время уменьшается скорость сходимости итерационного уточнения, что приводит к увеличению числа итераций. Отсюда вывод: для практических задач необходимо правильно подбирать значение ε. В данном случае, при оптимальном $\varepsilon = 10^{-3}$, алгоритм HSS в 3 раза быстрее и почти во столько же раз экономнее по памяти, чем PARDISO.

4.3. Эффективность HSS-алгоритма при решении задач большого размера. Чтобы показать эффективность использования памяти в HSS-алгоритме, проводились вычисления на размерах бóльших, чем возможные с использованием PARDISO. Постановка задачи та же, что и в предыдущих тестах. Отличие заключается в варьировании числа узлов n расчетной сетки по каждой оси.

В табл. 2 показаны размер памяти, необходимой для хранения матрицы L, полученной с помощью прямого метода, реализованного в PARDISO, а также полученной с использованием предложенного алгоритма.

Эффективность использования памяти предложенного алгоритма достигает более чем трехкратного превышения по сравнению с PARDISO. Из таблицы видно, что на больших размерах эффективность

Таблица 2
Размер памяти, необходимой для хранения матрицы L, ГБ

$n =$	150	160	170	180	190
PARDISO	35.0	47.2	60.7	77.1	97.6
HSS	12.6	15.6	19.6	23.6	27.5

больше, чем на малых, что дает большой потенциал для адаптации HSS-алгоритма под распределенные вычислительные системы, а именно при разработке MPI-версии.

Заключение. Разработанный, экспериментально обоснованный и протестированный алгоритм решения СЛАУ основан на технике вложенных сечений и аппроксимации матрицами малого ранга в методе Гаусса при построении множителей L и U. Данный алгоритм описан на примере симметричных положительно определенных матриц (нахождение L) и в дальнейшем может быть распространен на симметричные неопределенные, а также несимметричные СЛАУ.

Техника вложенных сечений позволяет уменьшить заполненность ненулевыми элементами матрицы L на порядок, а использование малоранговой аппроксимации позволяет дополнительно уменьшить количество используемой памяти для хранения L. Это основано на малоранговой аппроксимации недиагональных блоков указанных матриц, а также представления диагональных в иерархическом формате HSS (Hierarchically Semiseparable Structure). В процессе гауссова исключения матрицы A арифметические операции выполняются как в точной, так и в малоранговой, а также HSS-арифметике. Тем самым на порядок уменьшается время на формирование матрицы L.

Высокая эффективность предложенного алгоритма основана на применении оригинальных подходов: при построении малоранговой/HSS-аппроксимации плотных блоков из L используется техника адаптивной крестовой аппроксимации (АКА); малоранговая/HSS-аппроксимация используется адаптивно, т.е. основывается на предварительном анализе эффективности указанной аппроксимации для плотных блоков матрицы L; используется эффективный по памяти алгоритм $L_H L_H^T$-разложения диагональных H-блоков в гауссовом исключении. Кроме того, для повышения точности полученного решения используется техника итерационного уточнения. При программной реализации использовались компоненты BLAS и LAPACK из высокопроизводительной библиотеки Intel MKL, что в совокупности с предложенными подходами дает высокое качество и эффективность программного продукта, а это подтверждается тестами на синтетических данных.

Проверка работоспособности демонстрирует прямую зависимость точности получаемого решения от качества малоранговой/HSS-аппроксимации. Показана эффективность итерационного уточнения, позволяющего при доврольно низком качестве малоранговой аппроксимации L-множителя получить решение с точностью прямых методов всего за несколько итераций.

Сравнение программной реализации алгоритма с высокопроизводительной библиотекой Intel MKL (компонента PARDISO) показало более чем трехкратное превосходство как по памяти, так и по производительности гауссова исключения. Кроме того, продемонстрирована возможность решения трехмерных задач с числом неизвестных до 7×10^9 на машине с оперативной памятью 32 ГБ, в то время как PARDISO позволяет решать задачи не более чем с 3×10^9 неизвестными.

В заключение отметим, что разработанный алгоритм имеет большой потенциал для реализации на распределенных вычислительных системах, так как его эффективность по сравнению с прямыми методами гораздо больше на задачах больших размерностей, чем на малых.

Данная работа выполнена при поддержке гранта CRDF (US Civilian Research and Development Foundation; код RUE1-30034-NO-13) и грантов РФФИ (коды 14–01–31340, 14–05–31222, 14–05-93090, 14–05-00049).

СПИСОК ЛИТЕРАТУРЫ

1. *Бахвалов Н.С.* О сходимости одного релаксационного метода при естественных ограничениях на эллиптический оператор // Ж. вычисл. матем. и матем. физ. 1966. **6**, № 5. 861–885.
2. *Джордж А., Лю Дж.* Численное решение больших разреженных систем уравнений. М.: Мир, 1984.
3. *Ортега Дж.* Введение в параллельные и векторные методы решения линейных систем. М.: Мир, 1991.

4. *Федоренко Р.П.* Релаксационный метод решения разностных эллиптических уравнений // Ж. вычисл. матем. и матем. физ. 1961. **1**, № 5. 922–927.
5. *Bebendorf M.* Approximation of boundary element matrices // Numer. Mathem. 2000. **86**, N 4. 565–589.
6. *Chandrasekaran S., Dewilde P., Gu M., Somasunderam N.* On the numerical rank of the off-diagonal blocks of Schur complements of discretized elliptic PDEs // Siam Journal on Matrix Analysis and Applications. 2010. **31**, N 5. 2261–2290.
7. *Chandrasekaran S., Gu M., Li X.S., Xia J.* Some fast algorithms for hierarchically semiseparable matrices. Report LBNL-62897. Berkeley: Lawrence Berkeley Nat. Lab., 2007.
8. *Chandrasekaran S., Dewilde P., Gu M., Pals T.* A fast ULV decomposition solver for hierarchically semiseparable representations // SIAM J. Matrix Anal. Appl. 2006. **28**, N 3. 603–622.
9. *Ernst O.G., Gander M.J.* Why it is difficult to solve Helmholtz problems with classical iterative methods // Numerical Analysis of Multiscale Problems. Heidelberg: Springer, 2012. 325–363.
10. *George A.* Nested dissection of a regular finite element mesh // SIAM Journal on Numerical Analysis 1973. **10**, N 2. 345–363.
11. *Lipton R., Rose D., Tarjan R.* Generalized nested dissection // SIAM J. Numer. Anal. 1979. **16**, N 2. 346–358.
12. *Hackbusch W.* A sparse matrix arithmetic based on H-matrices. Part I: Introduction to H-matrices // Computing. 1999. **62**, N 2. 89–108.
13. *Bebendorf M., Hackbusch W.* Existence of H-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients // Numer. Mathem. 2003. **95**, N 1. 1–28.
14. *Karypis G., Kumar V.* METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 4.0. Minneapolis: University of Minnesota, 1998.
15. *Le Borne S., Grasedyck L., Kriemann R.* Domain-decomposition based H-LU preconditioners // Lecture Notes in Computational Science and Engineering. Vol. 55. Heidelberg: Springer, 2007. 667–674.
16. *Rjasanow S.* Adaptive cross approximation of dense matrices // Proc. Int. Association for Boundary Element Methods. Austin: Univ. Texas at Austin, 2002. 1–12.
17. *Saad Y.* Iterative methods for sparse linear systems. Philadelphia: SIAM, 2003.
18. *Saad Y., Vorst H.* Iterative solution of linear systems in the 20th century // Journal of Computational and Applied Mathematics. 2000. **123**, N 1. 1–33.
19. *Goreinov S.A., Tyrtyshnikov E.E., Zamarashkin N.L.* A theory of pseudo-skeleton approximations // Linear Algebra Appl. 1997. **261**, N 1–3. 1–21.
20. *Tyrtyshnikov E.E.* Mosaic-skeleton approximations // Calcolo. 1996. **33**, N 1. 47–57.
21. *Tyrtyshnikov E.E.* Incomplete cross approximation in the mosaic-skeleton method // Computing. 2000. **64**, N 4. 367–380.
22. *Wang S., de Hoop M.V., Xia J., Li X.S.* Massively parallel structured multifrontal solver for time-harmonic elastic waves in 3-D anisotropic media // Geophys. J. Int. 2012. **191**, N 1, 346–366.
23. *Wang S., Li X.S., Xia J., de Hoop M.V., Situ Y.* Efficient scalable algorithms for hierarchically semiseparable matrices // SIAM J. Sci. Comput. 2013. **35**, N 6. 519–544.
24. *Xia J.* A robust inner-outer hierarchically semi-separable preconditioner // Numer. Linear Algebra Appl. 2012. **19**, N 6. 992–1016.
25. *Xia J.* Efficient structured multifrontal factorization for general large sparse matrices // SIAM J. Scientific Computing. 2013. **35**, N 2. 832–860.
26. *Xia J.* Robust and efficient multifrontal solver for large discretized PDEs // High-Performance Scientific Computing. London: Springer, 2012. 199–217.
27. *Xia J., Chandrasekaran S., Gu M., Li X.S.* Superfast multifrontal method for large structured linear systems of equations // SIAM J. Matrix Anal. Appl. 2009. **31**, N 3. 1382–1411.

Поступила в редакцию
22.05.2014

Application of the Low-Rank Approximation Technique in the Gauss Elimination Method for Sparse Linear Systems

S. A. Solovyev [1]

[1] *Trofimuk Institute of Petroleum Geology and Geophysics, Siberian Branch of Russian Academy of Sciences; Koptyug prospekt 3, Novosibirsk, 630090, Russia; Ph.D., Scientist, e-mail: 511ssa@mail.ru*

Received May 22, 2014

Abstract: A fast direct algorithm for 3D discretized linear systems using the Gauss elimination method together with the nested dissection ordering approach and low-rank approximations is proposed. This algorithm

is described for symmetric positive definite matrices and can easily be extended to the case of nonsymmetric systems. In order to store the factor L in the LU-decomposition of the original matrix, the large-block representation as well as HSS format (Hierarchically Semiseparable Structure) are used. The construction of a low-rank approximation is based on using the adaptive cross approximation (ACA) approach, which is more efficient compared to the SVD and QR methods. In order to enhance the efficiency of the corresponding solver, a number of Intel MKL BLAS and LAPACK subroutines are used. This solver was implemented for shared memory computing systems. The functional testing shows a high quality of low-rank/HSS approximation. The performance testing demonstrates up to 3 times performance gain in comparison with the Intel MKL PARDISO direct solver. The proposed solver allows one to significantly decrease the memory and time consumption while using the Gauss elimination method.

Keywords: three-dimensional problems of mathematical physics, algorithms for sparse linear systems, Gauss elimination method, low-rank approximation, HSS matrix representation, iterative refinement.

References

1. N. S. Bakhvalov, "On the Convergence of a Relaxation Method with Natural Constraints on the Elliptic Operator," Zh. Vychisl. Mat. Mat. Fiz. **6** (5), 861–885 (1966) [USSR Comput. Math. Math. Phys. **6** (5), 101–135 (1966)].

2. A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems* (Prentice-Hall, Englewood Cliffs, 1981; Mir, Moscow, 1984).

3. J. M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems* (Plenum, New York, 1988; Mir, Moscow, 1991).

4. R. P. Fedorenko, "A Relaxation Method for Solving Elliptic Difference Equations," Zh. Vychisl. Mat. Mat. Fiz. **1** (5), 922–927 (1961) [USSR Comput. Math. Math. Phys. **1** (4), 1092–1096 (1962)].

5. M. Bebendorf, "Approximation of Boundary Element Matrices," Numer. Math. **86** (4), 565–589 (2000).

6. S. Chandrasekaran, P. Dewilde, M. Gu, and N. Somasunderam, "On the Numerical Rank of the Off-Diagonal Blocks of Schur Complements of Discretized Elliptic PDEs," SIAM J. Matrix Anal. Appl. **31** (5), 2261–2290 (2010).

7. S. Chandrasekaran, M. Gu, X. S. Li, and J. Xia, *Some Fast Algorithms for Hierarchically Semiseparable Matrices*, Report LBNL-62897 (Lawrence Berkeley Nat. Lab., Berkeley, 2007).

8. S. Chandrasekaran, P. Dewilde, M. Gu, and T. Pals, "A Fast ULV Decomposition Solver for Hierarchically Semiseparable Representations," SIAM J. Matrix Anal. Appl. **28** (3), 603–622 (2006).

9. O. G. Ernst and M. J. Gander, "Why it is Difficult to Solve Helmholtz Problems with Classical Iterative Methods," in *Numerical Analysis of Multiscale Problems* (Springer, Heidelberg, 2012), pp. 325–363.

10. A. George, "Nested Dissection of a Regular Finite Element Mesh," SIAM J. Numer. Anal. **10** (2), 345–363 (1973).

11. R. Lipton, D. Rose, and R. Tarjan, "Generalized Nested Dissection," SIAM J. Numer. Anal. **16** (2), 346–358 (1979).

12. W. Hackbusch, "A Sparse Matrix Arithmetic Based on H-Matrices. Part I: Introduction to H-Matrices," Computing **62** (2), 89–108 (1999).

13. M. Bebendorf and W. Hackbusch, "Existence of H-Matrix Approximants to the Inverse FE-Matrix of Elliptic Operators with L^∞-Coefficients," Numer. Math. **95** (1), 1–28 (2003).

14. G. Karypis and V. Kumar, *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 4.0* (Univ. of Minnesota, Minneapolis, 1998).

15. S. Le Borne, L. Grasedyck, and R. Kriemann, "Domain-Decomposition Based H-LU Preconditioners," in *Lecture Notes in Computational Science and Engineering* (Springer, Heidelberg, 2007), Vol. 55, pp. 667–674.

16. S. Rjasanow, "Adaptive Cross Approximation of Dense Matrices," in *Proc. Int. Association for Boundary Element Methods, May 28-30, 2002* (Univ. Texas at Austin, Austin, 2002), pp. 1–12.

17. Y. Saad, *Iterative Methods for Sparse Linear Systems* (SIAM, Philadelphia, 2003).

18. Y. Saad and H. Vorst, "Iterative Solution of Linear Systems in the 20th Century," J. Comput. Appl. Math. **123** (1), 1–33 (2000).

19. S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, "A Theory of Pseudo-Skeleton Approximations," Linear Algebra Appl. **261**, Nos. 1–3, 1–21 (1997).

20. E. E. Tyrtyshnikov, "Mosaic-Skeleton Approximations," Calcolo **33** (1), 47–57 (1996).

21. E. E. Tyrtyshnikov, "Incomplete Cross Approximation in the Mosaic-Skeleton Method," Computing **64** (4), 367–380 (2000).

22. S. Wang, M. V. de Hoop, J. Xia, and X. S. Li, "Massively Parallel Structured Multifrontal Solver for Time-Harmonic Elastic Waves in 3-D Anisotropic Media," Geophys. J. Int. **191** (1), 346–366 (2012).

23. S. Wang, X. S. Li, J. Xia, et al., "Efficient Scalable Algorithms for Hierarchically Semiseparable Matrices," SIAM J. Sci. Comput. **35** (6), 519–544 (2013).

24. J. Xia, "A Robust Inner-Outer Hierarchically Semi-Separable Preconditioner," Numer. Linear Algebra Appl. **19** (6), 992–1016 (2012).

25. J. Xia, "Efficient Structured Multifrontal Factorization for General Large Sparse Matrices," SIAM J. Sci. Comput. **35** (2), 832–860 (2013).

26. J. Xia, "Robust and Efficient Multifrontal Solver for Large Discretized PDEs," in *High-Performance Scientific Computing* (Springer, London, 2012), pp. 199–217.

27. J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, "Superfast Multifrontal Method for Large Structured Linear Systems of Equations," SIAM J. Matrix Anal. Appl. **31** (3), 1382–1411 (2009).