УДК 519.6

А.Н. ХИМИЧ, А.В. ПОПОВ, А.В. ЧИСТЯКОВ, В.А. СИДОРУК

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ РЕШЕНИЯ ЧАСТИЧНОЙ ПРОБЛЕМЫ СОБСТВЕННЫХ ЗНАЧЕНИЙ ДЛЯ БЛОЧНО-ДИАГОНАЛЬНЫХ МАТРИЦ С ОКАЙМЛЕНИЕМ

Аннотация. Предложен гибридный алгоритм метода итераций на подпространстве решения частичной обобщенной проблемы собственных значений для симметричных положительно-определенных разреженных матриц блочно-диагональной структуры с окаймлением на гибридных компьютерах с графическими процессорами. Получены коэффициенты эффективности алгоритма. Проведена апробация разработанного алгоритма на тестовых и практических задачах.

Ключевые слова: алгебраическая проблема собственных значений, компьютер гибридной архитектуры, гибридный алгоритм, метод итераций на подпространстве, эффективность параллельных алгоритмов, мелкоплиточный алгоритм.

ВВЕДЕНИЕ

Математическое моделирование различных объектов в областях науки и техники, социально-экономических процессов, а также явлений природы во многих случаях сводится к решению алгебраической проблемы собственных значений (АПСЗ) [1]. При этом желание получать более достоверные результаты расчетных задач приводит к необходимости решения АПСЗ для матриц больших порядков. (В настоящее время распространены задачи с порядком матрицы в несколько десятков миллионов.) В большинстве случаев при дискретизации краевых задач методом конечных элементов или конечных разностей возникают симметричные положительно-определенные разреженные матрицы, структура которых определяется нумерацией неизвестных задач и часто может быть ленточной, профильной, блочно-диагональной с окаймлением и т.п. [2].

Для эффективного решения возникающих АПСЗ больших объемов необходимо применять современные суперкомпьютеры и создавать алгоритмы, учитывающие специфику разреженности матриц, а также архитектурные и технологические особенности компьютеров. В настоящее время такими компьютерами являются суперкомпьютеры различных параллельных архитектур. Среди них особенно перспективны гибридные архитектуры — многоядерные компьютеры MIMD-архитектуры с сопроцессорами-ускорителями (зачастую графическими) SIMD-архитектуры [3].

ОСОБЕННОСТИ ГИБРИДНЫХ КОМПЬЮТЕРВ И БИБЛИОТЕК ПРОГРАММ ДЛЯ НИХ

При создании эффективных гибридных алгоритмов для MIMD-компьютеров с графическими процессорами необходимо учитывать особенности их сложной архитектуры. Далее рассматривается гибридный компьютер, архитектура которого состоит из многоядерного MIMD-компьютера (CPU) с топологией коммуникаций между процессорами «гиперкуб» и графических SIMD-процессоров (GPU). Каждый из p вычислительных процессов на ядрах CPU с логическими номерами $i = 0, 1, 2, \ldots$ взаимодействует с одним процессором GPU, имеющим такой же номер. Эта архитектура часто обозначается $p\text{CPU} + p\text{GPU}$.

Вычислительные узлы CPU объединены некоторой коммуникационной средой, и доступ к данным, которые размещены в памяти других узлов, выполняет-

ся дольше. Узлы CPU и GPU имеют отдельную память, поэтому данные, с которыми приходится оперировать на графических процессорах, необходимо явным образом копировать из памяти CPU в память GPU и обратно. Следовательно, объем вычислений на GPU должен значительно превышать объемы требуемых данных, которые нужно пересылать между CPU и GPU, в противном случае вычисления на GPU не перекроют накладных расходов на пересылку [3].

Одновременным выполнением множества потоков на GPU обеспечивается массовый параллелизм вычислений с помощью технологии CUDA [3]. Например, каждому потоку ставится в соответствие один элемент матрицы или компонента вектора. Эффективность решения большинства задач линейной алгебры во многом зависит от эффективного выполнения матрично-матричных и матрично-векторных операций, а также решения систем линейных алгебраических уравнений (СЛАУ). Поэтому для выполнения этих операций целесообразно использовать GPU, их максимальная производительность достигается при выполнении однотипных математических операций над большими объемами данных в условиях полной загрузки каждого вычислительного устройства.

В последнее время большое внимание уделяется созданию прикладного программного обеспечения для решения задач линейной алгебры. Наиболее известные библиотеки алгоритмов и программ для решения задач линейной алгебры на компьютерах различных параллельных архитектур разрабатываются под руководством профессора Донгарры из Института компьютерных наук университета Теннесси, США. Так, на основе библиотеки матрично-векторных операций BLAS и библиотеки программ последовательных блочных алгоритмов прямых методов для решения задач линейной алгебры с плотными и ленточными матрицами LAPACK [4] разработано программное обеспечение для различных параллельных архитектур, в частности, для MIMD-компьютеров создана библиотека программ ScaLAPACK [4].

На основе библиотеки BLAS сотрудники фирмы NVIDIA создали библиотеку программ CUBLAS [5] для платформы NVIDIA®CUDA™, а для разреженных матриц — аналогичную библиотеку программ CUSPARSE [6]. Эти библиотеки широко используются при разработке программного обеспечения для параллельных компьютеров с графическими процессорами. Например, под руководством профессора Донгарры разработана высокопроизводительная библиотека программ MAGMA (Matrix Algebra for GPU and Multicore Architectures) [7], в которой представлены плиточные алгоритмы для решения задач линейной алгебры — аналоги блочных алгоритмов из LAPACK. Высокая производительность решения задач при использовании MAGMA обеспечивается за счет автоматического выбора оптимального размера блока матриц на GPU для максимального их заполнения, оригинальной передачи данных на GPU с перекрытиями вычислений на CPU.

На основании библиотек BLAS и LAPACK созданы для многоядерных компьютеров с общей памятью библиотеки программ Core Math Library AMD (оптимизирована под процессоры компании AMD) [8] и Intel Math Kernel Library (оптимизирована под процессоры компании Intel) [9].

Также отметим параллельную библиотеку программ SLEPc (Scalable Library for Eigenvalue Problem Computations) [10], предназначенную для решения АПСЗ с большими разреженными матрицами, и особенно библиотеку программ LIS (Library of Iterative Solvers) [11], которая позволяет решать АПСЗ итерационными методами с матрицами различных форматов на параллельных компьютерах с использованием арифметики расширенной разрядности.

В Институте кибернетики им. В.М. Глушкова НАН Украины также проводятся исследования по созданию численного программного обеспечения для решения задач линейной алгебры для различных параллельных компьютерных архитектур и структур матриц [12–15]. При этом реализуются такие принципы разработки параллельных программ: исследование соответствия математических свойств задачи с приближенными данными выбранному параллельному алгоритму решения; построение эффективной компьютерной топологии из оптимального количества компьютерных устройств; эффективные способы распараллеливания данных; решение задачи с анализом достоверности компьютерных результатов [12]. В работах [16–19] описано применение созданного численного программного обеспечения для решения систем линейных алгебраических уравнений в математическом моделировании прикладных задач для электросварки и гражданского строительства на компьютерах различной параллельной архитектуры.

В настоящей работе предлагается гибридный алгоритм метода итераций на подпространстве решения частичной обобщенной проблемы собственных значений симметричных положительно-определенных разреженных матриц блочно-диагональной структуры с окаймлением.

ГИБРИДНЫЙ АЛГОРИТМ МЕТОДА ИТЕРАЦИЙ НА ПОДПРОСТРАНСТВЕ

Постановка задачи и метод решения. Рассмотрим решение частичной обобщенной проблемы собственных значений

$$Ax = \lambda Bx, \tag{1}$$

где A, B — симметричные положительно-определенные разреженные блочно-диагональные матрицы с окаймлением порядка n. Предположим, что исходную матрицу изначально можно представить в блочно-диагональном виде с окаймлением (в результате применения метода конечных разностей или конечных элементов) или получить после структурной регуляризации, использовав, например, метод параллельных сечений [12]. Отметим, что в этом случае реализуются преобразования подобия, поэтому собственные значения исходной и преобразованной матриц совпадают.

Для нахождения r минимальных собственных значений и соответствующих им собственных векторов задачи (1) применим метод итераций на подпространстве, являющийся обобщением метода обратных итераций и заключающийся в построении для задачи (1) последовательности подпространств E_t $(t = 1, 2, \dots)$, которая сходится к подпространству E_∞, содержащему искомые собственные векторы [12]. На t-й итерации вычисляется ортогональный базис подпространства E_t и, если достигнута требуемая точность приближенного решения, определяются искомые собственные пары. Таким образом, реализация метода итераций на подпространстве сводится к решению для $t = 1, 2, \dots$ следующих подзадач [14]:

— решение СЛАУ

$$AX_t = Y_{t-1}; \tag{2}$$

— вычисление прямоугольной матрицы

$$W_t = BX_t; \tag{3}$$

— нахождение проекций матриц A и B на подпространство E_t

$$A_t = X_t^{\mathrm{T}} Y_{t-1} \equiv X_t^{\mathrm{T}} AX_t,\ B_t = X_t^{\mathrm{T}} W_t \equiv X_t^{\mathrm{T}} BX_t; \tag{4}$$

— решение полной проблемы собственных значений для проекций

$$A_t Z_t = B_t Z_t \Lambda_t; \tag{5}$$

— вычисление приближения

$$Y_t = W_t Z_t. \tag{6}$$

Если после k-й итерации выполняются условия окончания итерационного процесса, например $\left| \dfrac{\lambda_i^{(c)} - \lambda_i^{(c-1)}}{\lambda_i^{(t)}} \right| \le \varepsilon$, то проводится дополнительная итерация и в качестве приближенных решений задачи (1) полагаются $\lambda_i^* = \lambda_i^{(c+1)}$ $(i = 1, 2, \ldots, r)$, а также первые r столбцов матрицы $X^* = X_{c+1} Z_{c+1}$ (имеется в виду, что собственные значения упорядочены по возрастанию $0 < \lambda_1 \le \lambda_2 \le \ldots \le \lambda_r \le \ldots$).

Итерационный процесс сходится линейно, причем скорость сходимости λ_i определяется отношением λ_q / λ_1, где q — размерность итерируемого подпространства E_t [12]. В последовательных реализациях алгоритма рекомендуется выбирать $q = \min(2r, r+8)$. Заметим, что решение СЛАУ (2) на каждой итерации выполняется с одной и той же матрицей A, полученной в результате треугольного разложения (например, LL^{T}) до начала итерационного процесса.

Основные этапы гибридного алгоритма метода итерации на подпространстве. На основе проведенного анализа архитектурных и технологических особенностей гибридного компьютера этапы решения частичной АПСЗ (1) гибридным алгоритмом метода итераций на подпространстве можно разделить на четыре подзадачи (различной вычислительной сложности).

1. Формирование распределенной между процессами матрицы Y_0 начальных итерируемых векторов таких, чтобы матрица B_1 из (4) была положительно-определенной. Эта операция выполняется на CPU, причем без обменов данными между процессами с использованием, например, предложенного в [12] параллельного алгоритма.

2. Проведение LL^{T}-факторизации разреженной симметричной положительно-определенной матрицы A (на CPU и GPU) с использованием, например, гибридного алгоритма для случая ленточной матрицы [13].

3. Выполнение итерационного процесса (2)–(6), в котором для каждого $t = 1, 2, \ldots$ вычисления распределяются между процессорными устройствами следующим образом:

— для решения СЛАУ (2) используется полученное ранее LL^{T}-разложение матрицы A, т.е. решаются системы с треугольными матрицами $LV = Y_{t-1}$ и $L^{\mathrm{T}} X_t = V$;

— вычисление (3) прямоугольной матрицы $W_t = BX_t$ в зависимости от структуры (разреженная или диагональная) матрицы B, размерности итерируемого подпространства, количества используемых GPU и объема их памяти выполняется либо на CPU, либо на CPU и GPU, либо только на GPU;

— вычисления (4) произведений прямоугольных матриц для формирования проекций матриц A и B на подпространство выполняются на GPU, а сборка матриц проекций (при использовании более одного GPU) проводится каждым процессом на CPU. Такой подход существенно сокращает объем данных, которыми обмениваются CPU и GPU. Кроме того, вычисление произведений прямоугольных матриц на GPU можно выполнять асинхронно с другими вычислительными операциями или обменами;

— для решения полной обобщенной АПСЗ (5) с учетом сравнительно небольшого порядка матриц проекций используется метод Якоби. Задача решается каждым процессом на CPU, и в таком случае не требуются обмены данными между вычислительными устройствами;

— проверка условий окончания итерационного процесса выполняется каждым процессом на CPU;

— вычисление (6) новой матрицы итерируемых векторов Y_t (или матрицы приближенных собственных векторов X^*) выполняется на GPU в соответствии с распределением данных (вычисляется подматрица матрицы Y_t или X^*), при этом нет необходимости в обменах данными между процессорными устройствами.

4. Исследование достоверности результатов — вычисление оценок погрешности полученного решения (см., например, [12]). Для определения оценок относительной погрешности вычисленных приближений к собственным значениям выполняются операции, аналогичные выполняемым в итерационном процессе. Для реализации этих операций используются те же гибридные алгоритмы, что и в итерационном процессе.

Заметим, что для решения частичной АПСЗ нужно проводить отдельное исследование. Получив решение, следует убедиться, что найдены все необходимые собственные значения (и соответствующие собственные векторы). Например, при вычислении m минимальных собственных значений может возникнуть ситуация, когда вместо одного из них вычислено $(m+1)$-е собственное значение. В частности, подобные ситуации могут возникать, если начальное подпространство E_0 окажется B-ортогональным по крайней мере к одному из искомых собственных векторов. Установить, когда ввиду ортогональности к исходному подпространству не найдена одна (или более) из искомых собственных пар, можно, использовав свойство последовательности Штурма матрицы $A-\sigma B$ [20]. Поскольку матрица $A-\sigma B$ знаконеопределенная, для обеспечения устойчивости при LDL^{T}-разложении сдвиг σ выбирается бóльшим максимального из испытываемых собственных значений с учетом кратности или патологической близости в пределах точности вычислений, например $\sigma = 1.001\lambda_m$.

Как видно из описанного алгоритма, наибольшую вычислительную сложность имеет разложение блочно-диагональной матрицы с окаймлением. Рассмотрим подробнее представление матрицы и алгоритм LL^{T}-факторизации (гибридный алгоритм LDL^{T}-факторизации принципиально от него не отличается, различие заключается лишь в некоторых деталях).

Схема распределения матриц между процессорными устройствами. Блочно-диагональную матрицу с окаймлением можно представить в таком виде:

$$A = P^{\mathrm{T}}\widetilde{A}P = \begin{pmatrix} D_{11} & 0 & 0 & \dots & 0 & C_{1p} \\ 0 & D_{22} & 0 & \dots & 0 & C_{2p} \\ 0 & 0 & D_{33} & \dots & 0 & C_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & D_{p-1p-1} & C_{p-1p} \\ C_{p1} & C_{p2} & C_{p3} & \dots & C_{pp-1} & D_{pp} \end{pmatrix}, \qquad (7)$$

где P — матрица перестановок, а блоки D_{ii}, C_{ip} и C_{pi} сохраняют разреженную структуру, p — количество диагональных блоков, $p \geq 3$.

Благодаря такой структуре разреженной матрицы, можно эффективно проводить параллельную независимую обработку блоков D_i, C_i, $i=\overline{1, p-1}$, и при

этом изменять порядки и количество блоков в разбиении, что способствует гибкому управлению нагрузкой на вычислительные устройства. Заметим, что наибольшая эффективность распараллеливания достигается, если порядок последнего диагонального блока намного меньше, чем порядки других диагональных блоков. Кроме того, такая блочная структура позволяет работать с неразрывными массивами данных на GPU, что уменьшает количество индексных операций и проверок, которые на графическом ускорителе достаточно затратны.

Учитывая структуру матрицы, целесообразно использовать блочное распределение данных. Причем количество блоков следует выбирать в зависимости от количества процессоров, задействованных в расчетах.

Распараллеливание данных и вычислений на отдельные процессы CPU выполняется с помощью системы распараллеливания MPI [12].

Мелкоплиточный гибридный алгоритм LL^{T}-разложения блочно-диагональной матрицы с окаймлением. Рассмотрим мелкоплиточный гибридный алгоритм факторизации разреженной блочно-диагональной матрицы A с окаймлением, который используется в гибридном алгоритме метода итераций на подпространство. Данный алгоритм хорошо учитывает профильную структуру диагональных блоков и блоков окаймления.

Разобьем матрицу A на блоки размером $h\times h$. Далее для факторизации блочно-диагональной матрицы применим алгоритм, предложенный в [20] для плотных матриц.

Для факторизации матрицы на k-м шаге используется следующее соотношение:

$$A^k=\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}=\begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix}\begin{pmatrix} L_{11}^{\mathrm{T}} & L_{21}^{\mathrm{T}} \\ 0 & L_{22}^{\mathrm{T}} \end{pmatrix}, \tag{8}$$

где у блока A_{11} размер $h\times h$, у блока A_{21} размер $(n-kh)\times h$, у блока A_{22} размер $(n-kh)\times(n-kh)$, при этом блоки A_{11}, A_{21} и A_{22} соответствуют структуре блоков D_{ii}, C_{pi} и D_{pp}. Отсюда получаем алгоритм, с помощью которого проводится разложение на k-м шаге:

$$A_{11}=L_{11}*L_{11}^{\mathrm{T}}, \tag{9}$$

$$L_{21}=A_{21}*(L_{11}^{\mathrm{T}})^{-1}, \tag{10}$$

$$\widetilde{A}_{22}=A_{22}L_{21}*L_{21}^{\mathrm{T}}. \tag{11}$$

Отметим, что реализация (9)–(11) на каждом шаге модифицирует блоки D_{ii}, C_{pi}, $i=\overline{1,\,p-1}$, и D_{pp}. Аналогично операции (10) и (11) выполняются лишь с ненулевыми блоками подматрицы A_{21}. Поэтому в случае блочно-диагональной матрицы с окаймлением подматрицы D_{ii} и C_{pi}, а также подматрица $D_{pp,i}=L_{pi}(L_{pi})^{\mathrm{T}}$ модифицируются независимо (для каждого i). Это позволяет эффективно распараллеливать алгоритм.

Для реализации алгоритма на гибридном компьютере используем такую схему распределения: в GPU(i) ($i=\overline{1,\,p-1}$) содержатся одинакового размера блоки D_{ii}, C_{pi} и $A_{pp}^{(i)}$, а в GPU(p) — блок D_{pp}.

На рис. 1 представлено блочное распределение данных на k-м шаге факторизации блочно-диагональной матрицы с окаймлением с учетом предложенной декомпозиции.

Параллелизация вычислений треугольной факторизации состоит в том, что разложение блоков A_{11}, а также модификация A_{21} и A_{22} могут происходить независимо в каждом CPU(i) и GPU(i) для $i = \overline{1,\ p-1}$.

На каждом шаге во всех парах CPU(i) и GPU(i) для $i = \overline{1,\ p-1}$ выполняются такие вычисления:

— в CPU(i) факторизуется A_{11} из D_{ii}: $A_{11} = L_{11} * L_{11}^{\mathrm{T}}$;

— в GPU(i) модифицируется столбец блоков L_{21}: $L_{21} = A_{21}(L_{11}^{\mathrm{T}})^{-1}$;

— в GPU(p) модифицируются блоки матрицы A_{22} из $A_{pp}^{(i)}$: $\widetilde{A}_{22} = A_{22} - L_{21}L_{21}^{\mathrm{T}}$;

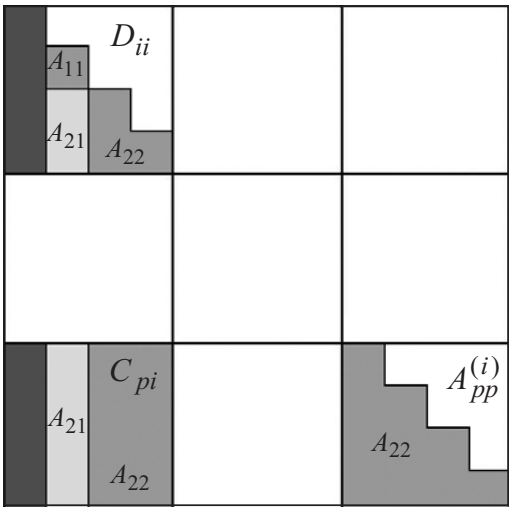

Рис. 1. Декомпозиция данных на GPU(i)

— в CPU(p) с использованием мультисборки модифицируется блок D_{pp}: $D_{pp}^{*} = D_{pp} - \sum_{i=1}^{p-1} A_{pp}^{(i)}$.

После этого факторизируется блок D_{pp}^{*} и процесс факторизации матрицы A заканчивается.

Эффективность гибридного алгоритма. Исследование эффективности предложенного гибридного алгоритма базируется на методологиях из [10, 15]. Как отмечалось ранее, рассматривается компьютер гибридной архитектуры pCPU + pGPU.

Эффективность параллельных алгоритмов характеризуется коэффициентами ускорения $S_p = T_1 / T_p$ и эффективности $E_p = S_p / p$. Здесь (для гибридной архитектуры) T_1 — время решения задачи на архитектуре 1CPU + 1GPU, T_p — время решения этой же задачи на архитектуре pCPU + pGPU. Времена T_1 и T_p можно вычислить по формулам:

$$T_1 = Ot_g,$$
$$T_p = Ot_g + M_1 t_{opg} + M_2 t_{opp} + Q_1 t_{cpg} + Q_2 t_{cpp}, \tag{12}$$

где O — количество операций в алгоритме; t_g — среднее время выполнения одной арифметической операции на GPU; t_{opg} — время обмена информацией между CPU и GPU; t_{opp} — время обмена одним машинным словом между двумя процессами CPU; t_{cpg} — время для установки связи между CPU и GPU; t_{cpp} — время установки связи между двумя процессами CPU; M_i, Q_i — соответственно количество обменов и синхронизаций между CPU и GPU. При использовании гибридной архитектуры в формуле (12) необходимо учитывать синхронность или асинхронность работы CPU и GPU. В случае асинхронной работы следует суммирование в (12) заменять определением максимального значения, а в случае гибридной архитектуры целесообразно оценивать только ускорение, полученное за счет использования GPU.

Поскольку вычисления происходят в основном на GPU, который реализует архитектуру SIMT (Single Instruction, Multiple Threads), при дальнейших расчетах следует учитывать, что одна инструкция выполняется для группы данных, т.е.

одновременно происходит n_0 операций с плавающей точкой. Величина n_0 зависит от аппаратных характеристик конкретного графического ускорителя.

При исследовании и решении частичной АПСЗ (1) для симметричных матриц часть этапов (формирование матрицы начальных итерируемых векторов, LL^{T}-разложение матрицы A, вычисление оценок приближенного решения) имеет фиксированное количество арифметических операций, а для итерационного процесса (2)–(6) количество арифметических операций пропорционально количеству выполненных итераций. Количество итераций, необходимое для нахождения r минимальных собственных значений, как правило, определяется величиной $O(r)$, причем оно тем меньше, чем больше размерность q итерируемого подпространства.

Таким образом, $T_p = T_p^{(F)} + c_I T_p^{(1I)}$, а коэффициент ускорения предлагаемого гибридного алгоритма можно представить в виде

$$S_p = \frac{T_1}{T_p} = \frac{T_p^{(F)}}{T_p} S_p^{(F)} + \frac{c_I T_p^{(1I)}}{T_p} S_p^{(It)}, \tag{13}$$

где c_I — количество итераций, $T_p^{(F)}$, $T_p^{(1I)}$ — соответственно время решения подзадач с фиксированным числом арифметических операций и время выполнения одной итерации на $p\mathrm{CPU} + p\mathrm{GPU}$, а $S_p^{(F)}$, $S_p^{(1I)}$ — коэффициенты ускорения алгоритмов решения этих подзадач.

Далее для архитектуры $p\mathrm{CPU} + p\mathrm{GPU}$ используем обозначения: $T_p^{(LLT)}$ — время для LL^{T}-разложения матрицы A; $T_p^{(SS)}(k)$ — время для решения СЛАУ вида (2) с k правыми частями с применением вычисленных ранее LL^{T}-разложений матрицы системы; $T_p^{(Fo)}$, $T_p^{(Io)}(k)$ — соответственно времена выполнения остальных операций при решении подзадач с фиксированным числом арифметических операций и при выполнении одной итерации. Тогда $T_p^{(F)}$ и $T_p^{(1I)}$ можно представить в виде:

$$T_p^{(F)} = T_p^{(LLT)} + T_p^{(SS)}(q) + T_p^{(Fo)}, \;\; T_p^{(1I)} = T_p^{(SS)}(q) + T_p^{(Io)}(q). \tag{14}$$

Заметим, что здесь и далее формулы для всех $T_1^{(*)}$ можно получить, подставив в формулы для $T_p^{(*)}$ значение $p = 1$, и что для всех этих времен справедлива формула (13). Тогда коэффициент ускорения гибридного алгоритма метода итераций на подпространстве для архитектуры $p\mathrm{CPU} + p\mathrm{GPU}$ можно вычислить по формуле

$$S_p = \frac{T_p^{(F)}}{T_p}(\alpha_p^{(LLT)} S_p^{(LLT)} + \alpha_p^{(SS)}(q) S_p^{(SS)}(q) + \alpha_p^{(Fo)} S_p^{(Fo)}) + \\ + \frac{c_I T_p^{(1I)}}{T_p}(\beta_p^{(SS)}(q) S_p^{(SS)}(q) + \beta_p^{(Io)}(q) S_p^{(Io)}(q)), \tag{15}$$

где $\alpha_p^{(*)} = T_p^{(*)} / T_p^{(F)}$, $\beta_p^{(*)} = T_p^{(*)} / T_p^{(1I)}$, $S_p^{(*)}$ — коэффициент ускорения гибридного алгоритма решения соответствующей подзадачи (выполнения операций), а верхние индексы и параметры совпадают с соответствующими индексами и параметрами для времен $T_p^{(*)}$.

Далее рассмотрим случай, когда матрицы задачи (1) являются блочно-диагональными с окаймлением. Для случая использования гибридных алгоритмов факторизации (см. [13]) можно сформулировать такие леммы (пренебрегая величинами малых порядков). Для простоты изложения будем считать, что порядок всех диагональных блоков $z_i \approx z = (n-\upsilon)/(p-1)$, где υ — порядок последнего диагонального блока. Тогда справедливы следующие леммы.

Лемма 1. Количество операций, которые выполняются на одном GPU для нахождения факторизации блочно-диагональной матрицы с окаймлением, оценивается величиной $O_p \dfrac{z^3}{3} + \upsilon z^2 = \dfrac{z^2}{3}(z+3\upsilon)$, а затрачиваемое время

$$T_p \approx \frac{z^3 + 3z\upsilon}{3}\frac{t_g}{n_0} + \frac{\upsilon^2 \log_2 p}{2} t_{opp} + (2zh \log_2 p + p\upsilon^2 + 2\upsilon h) t_{opg}.$$

Лемма 2. Ускорение мелкоплиточного гибридного алгоритма LL^{T}-разложения разреженной блочно-диагональной матрицы A с окаймлением составляет

$$S_p^{LLT} \approx (\log_2 p)\left(1 + \frac{3\upsilon^2 \log_2 p}{2z^2(z+3\upsilon)}\left(\tau_{opp} + \left(\frac{2p}{(\log_2 p)} + \frac{4zh}{\upsilon^2} + \frac{4h}{(\log_2 p)\upsilon}\right)\tau_{opg}\right)\right)^{-1}, \quad (16)$$

где h — размер плитки при использовании гибридного алгоритма мелкоплиточного разложения, $\tau_{opp} = \dfrac{n_0 t_{opp}}{t_g}$, $\tau_{opg} = \dfrac{n_0 t_{opg}}{t_g}$.

При получении оценки $T_p^{(Io)}(k)$ необходимо учитывать, что в общем случае наибольшее количество арифметических операций выполняется при вычислении произведения разреженной матрицы B и $(n \times k)$-матрицы, а также при вычислении матриц или векторов, элементами которых являются скалярные произведения n-мерных векторов, при η_B — среднем количестве ненулевых элементов в одной строке матрицы B. Количество остальных арифметических операций, выполняемых одним из p GPU, значительно меньше и им можно пренебречь. Тогда справедлива такая лемма.

Лемма 3. Количество операций при выполнении итерационного процесса для блочно-диагональных матриц с окаймлением оценивается величиной $T_p^{(Io)}(k)$, а время выполнения остальных операций — величиной $T_p^{(Fo)}$:

$$T_p^{(Io)}(k) = \frac{n}{p}\left(\frac{(n+2k)\log_2 p + O(k^2)}{n} pkt_C + (2\eta_B + 4k)k\frac{t_G}{n_0}\right) + \\ + (t_C^{(E)}(zk)\log_2 p + 2t_G^{(E)}(zk))\frac{n}{z}, \quad (17)$$

$$T_p^{(Fo)} = \frac{n}{p}\left(\frac{n + 2(n+2)p\log_2 p}{n} qt_C + (2\eta_B + 6)q\frac{t_G}{n_0}\right) + \\ + (2t_C^{(E)}(zq)\log_2 p + 2t_G^{(E)}(zq))\frac{n}{z}, \quad (18)$$

где t_C — среднее время выполнения одной арифметической операции на CPU, $t_C^{(E)}(d) = t_{cpp} + dt_{opp}$ и $t_G^{(E)}(d) = t_{cpg} + dt_{opg}$ — времена обмена d двойных слов между двумя CPU или CPU и соответствующего GPU.

Влияние $T_p^{(Io)}(q)$ зависит от соотношения η_B и q, если q существенно больше η_B, то в (17) можно пренебречь слагаемым, содержащим η_B, если $z >> \eta_B$ и

$z >> q$, то в (14) можно пренебречь $T_p^{(Io)}(q)$. Из (17) также следует оценка

$$S_p^{(Io)}(k) \approx p\left(1+\frac{p\log_2 p}{2k(\eta_B+2k)}\frac{n_0}{t_G}(kt_C+t_C^{(E)}(zk))\right)^{-1}. \qquad (19)$$

Теперь можно оценить ускорение и эффективность предлагаемого гибридного алгоритма в целом.

Из анализа полученных оценок (14)–(19) следует, что эффективность предложенного алгоритма в значительной мере определяется эффективностью алгоритма LL^{T}-разложения, например, гибридного плиточного алгоритма [16]. Заметим, если накладные расходы на пересылку данных между вычислительными устройствами меньше времени вычислений, то эти алгоритмы за счет использования асинхронности будут иметь коэффициенты эффективности, близкие к единице.

Теорема 1. Для гибридного алгоритма метода итерации на подпространстве с использованием гибридного мелкоплиточного алгоритма LL^{T}-разложения при условии $t_G^{(m)} \geq t_C^{(E)}$ коэффициент ускорения составляет

$$S_p \approx \frac{1}{T_p^{(LLT)}+c_I T_p^{(1I)}}\left(\frac{T_p^{(LLT)}T_1^{(LLT)}+c_I T_p^{(F)}T_1^{(1I)}}{T_p^{(F)}}\right),$$

где

$$T_p^{(LLT)} = \frac{z^3+3z\upsilon}{3}\frac{t_g}{n_0}+\frac{\upsilon^2\log_2 p}{2}t_{opp}+(2zh\log_2 p+p\upsilon^2+2\upsilon h)t_{opg},$$

$$T_p^{(Fo)} = \frac{n}{p}\left(\frac{n+2(n+2)p\log_2 p}{n}qt_C+(2\eta_B+6)q\frac{t_G}{n_0}\right)+$$

$$+(2t_C^{(E)}(zq)\log_2 p+2t_G^{(E)}(zq))\frac{n}{z},$$

$$T_p^{(Io)}(k) = \frac{n}{p}\left(\frac{(n+2k)\log_2 p+O(k^2)}{n}pkt_C+(2\eta_B+4k)k\frac{t_G}{n_0}\right)+$$

$$+(t_C^{(E)}(zk)\log_2 p+2t_G^{(E)}(zk))\frac{n}{z}.$$

Эффективность алгоритма оценивается величиной $E_p = S_p / p$.

Как уже упоминалось, на эффективность гибридного алгоритма решения частичной обобщенной АПСЗ методом итераций на подпространстве существенное влияние имеет количество c_I выполненных итераций для достижения требуемой точности. Обычно при постоянном r количество выполняемых итераций c_I тем меньше, чем больше размерность итерируемого подпространства q. Как следует из представленных оценок, объем вычислений пропорционален q. Однако в ряде случаев при относительно малых значениях q наблюдается недостаточная загруженность GPU (т.е. фактически уменьшается значение n_0) при увеличении количества обменов, что приводит к снижению реальной эффективности алгоритма. В таких случаях увеличение размерности итерируемого подпространства (пока не будет достигнуто максимального значения n_0) позволяет сократить время решения задачи за счет сокращения количества итераций [14].

ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ГИБРИДНОГО АЛГОРИТМА

Апробация разработанного гибридного алгоритма метода итераций на подпространстве проводилась на матрицах различных порядков (в том числе на матрицах из коллекции Флоридского университета [21] (табл. 1)), на компьютере гибрид-

Таблица 1. Тестовые ленточные матрицы из коллекции Флоридского университета

Название задачи	Проблемная область	Порядок матрицы (количество ненулевых элементов)
G2_circuit	Structural problem	150102 (726674)
Bone010	Model reduction problem	986703 (47851783)
Emila_923	Structural problem	923136 (40373538)

ной архитектуры СКИТ-4 [22] с использованием процессорных узлов CPU и графических процессоров с такими техническими характеристиками: CPU — Intel(R) Xeon(R) CPU E5-26700; тактовая частота — 2.60 GHz; скорость — 8 GT/s; кеш-память — 20 MB; в узле 2 CPU по 8 ядер + Hyperthreading = 32 ядра; максимальный объем памяти — 384 GB; GPU — Nvidia Tesla M2050; память — 3 GB; пиковая производительность — 515 Gflops.

Программы, реализующие алгоритмы, написаны на языке программирования C++ с использованием на CPU системы MPI и библиотеки программ Intel MKL [9], а для распараллеливания на GPU — с применением технологии CUDA, библиотеки программ cuBLAS и cuSPARSE [7, 8].

На рис. 2 представлен график ускорения решения АПСЗ гибридным алгоритмом метода итераций на подпространстве для различных матриц из коллекции Флоридского университета на различных гибридных архитектурах. Проведенные эксперименты свидетельствуют, что обеспечивается хорошая масштабируемость алгоритма, т.е. полученное ускорение решения задачи пропорционально увеличивается с ростом количества вычислительных устройств.

На рис. 3 приведен график зависимости ускорения алгоритма от количества блоков, полученных при приведении матриц к блочно-диагональному виду с окаймлением для матрицы Bone010 (см. табл. 1). Использование оптимального количества блоков при выполнении этой операции в LL^{T}-разложении и в итерационном процессе позволяет хорошо сбалансировать загрузку используемых вычислительных устройств (минимизация эффекта Гайдна), повысить эффективность вычислений и, как следствие, существенно сократить время решения задач. В зависимости от типа вычислительного устройства и его характеристик рекомендуемый размер плиток составляет от 192 до 1024 для одного вычислительного процесса. Разработанный гибридный алгоритм позволяет эффективно использовать современные вычислительные устройства различного типа и адаптироваться к используемой структуре матрицы. Из рис. 2, 3 видно, что результаты практических экспериментов соответствуют теоретическим исследованиям алгоритма.

Апробация разработанного гибридного алгоритма метода итераций на подпространстве также проводилась на практических задачах. Например, решалась задача устойчивости слоистого двухкомпонентного композитного материала регулярной структуры [23, 24] при равномерном одноосном сжатии армирующих слоев поверхностной нагрузкой по-

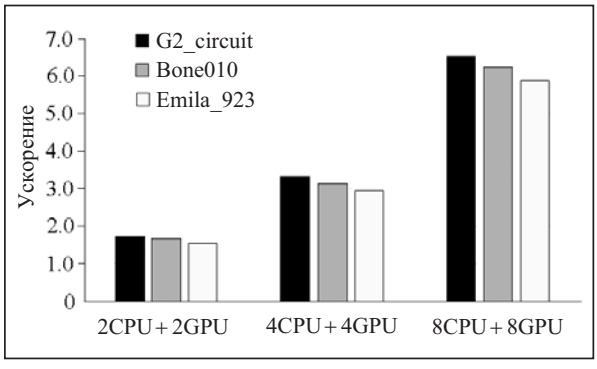

Рис. 2. График ускорения решения различных задач гибридным алгоритмом метода итераций на подпространстве

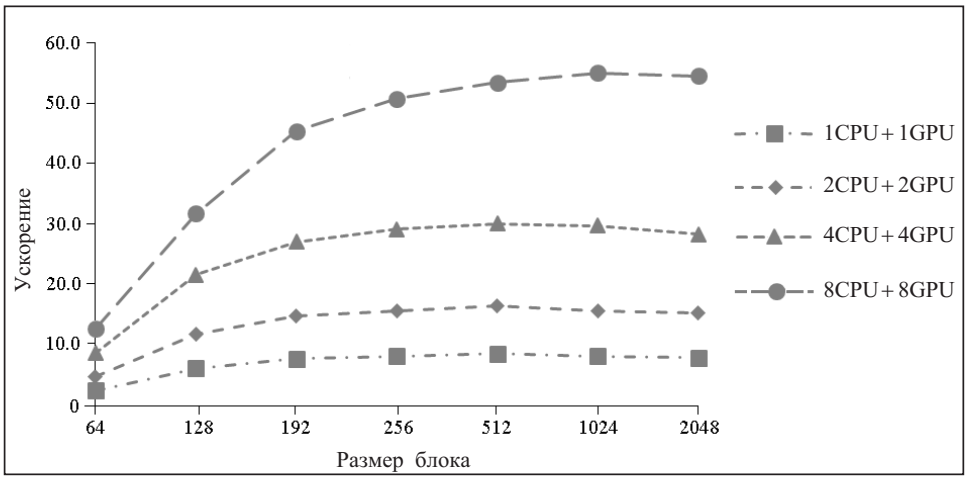

Рис. 3. График зависимости ускорения от размера блока при факторизации для алгоритма метода итераций на подпространстве

стоянной интенсивности, которая сводилась к обобщенной задаче на собственные значения. Вычисленное гибридным алгоритмом метода итераций на подпространстве минимальное собственное значение определило критическую нагрузку, а соответствующая собственная функция — форму потери устойчивости.

Задача решалась с различными исходными данными. Далее приведен листинг протокола решения задачи с такими исходными данными: порядок матриц A и B — 12282, полуширина ленты матрицы A — 6212, полуширина ленты матрицы B — 71.

Фрагмент протокола решения задачи

```
    Problem solving: total time = 7.84e+00
    Results: SOLUTION WAS CALCULATED by 20 iterations

FIRST 3 EIGENVALUES
    Eigenvalues (calculated) Estimates of Errors
        1.248665556779830e-01        1.893e-06
        1.248734577580778e-01        8.894e-07
        1.248765574182433e-01        2.837e-06
```

В протоколе представлены три вычисленные минимальные собственные значения. Решение этой задачи с помощью пакета MATLAB [25] получено за 340 с. Время решения задачи гибридным алгоритмом метода итераций на подпространстве с использованием мелкоплиточного алгоритма факторизации блочно-диагональной матрицы с окаймлением на СКИТ-4 составляет 7.84 с, что приблизительно в 43 раза быстрее.

ЗАКЛЮЧЕНИЕ

В работе предложен и исследован новый параллельный алгоритм метода итераций на подпространстве для решения частичной обобщенной проблемы собственных значений положительно-определенных блочно-диагональных матриц с окаймлением на гибридных компьютерах. Исследованы коэффициенты ускорения и эффективности созданного алгоритма. Проведено экспериментальное исследование на тестовых и прикладных задачах. Разработанный гибридный алгоритм учитывает структуру матрицы, оптимизирует коммуникации между процессорными элементами и использование памяти, достигая тем самым высокой эффективности.

В настоящее время постоянно растут объемы задач, которые приходится решать на современных суперкомпьютерах. Поэтому актуальна проблема эффективного использования компьютерных ресурсов для таких задач. Дальнейшие исследования авторов будут направлены на создание новых параллельных алгоритмов решения задач линейной алгебры для различных разреженных структур матриц, а также поиска эффективных способов их переупорядочения и хранения.

СПИСОК ЛИТЕРАТУРЫ

1. Писанецки С. Технология разреженных матриц. Москва: Мир, 1988. 410 с.
2. Парлет Б. Симметричная проблема собственных значений. Москва: Мир, 1983. 382 с.
3. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. Москва: Пресс, 2010. 232 с.
4. NetLib. 2015. URL: http://www.netlib.org/.
5. cuBLAS. URL: https://developer.nvidia.com/cublas.
6. cuSparse Library. URL: http://docs.nvidia.com/cuda/cuSPARSE/.
7. MAGMA. URL: http://icl.cs.utk.edu/magma/.
8. AMD. URL: http://www.amd.com/en-gb.
9. Math Kernel Library. URL: https://software.intel.com./en-us/mkl.
10. SLEPc. 2015. URL: http://slepc.upv.es/.
11. LIS. 2015. URL: http://www.ssisc.org/lis/.
12. Химич А.Н., Молчанов И.Н., Попов А.В., Чистякова Т.В., Яковлев М.Ф. Параллельные алгоритмы решения задач вычислительной математики. Киев: Наук. думка, 2008. 247 с.
13. Хіміч О.М., Сидорук В.А. Гібридний алгоритм розв'язування лінійних систем з розрідженими матрицями на основі блочного LL^{T}-методу. *Комп'ютерна математика*. 2015. № 1 С. 67–74.
14. Химич А.Н., Попов А.В., Чистяков А.В. Гибридные алгоритмы решения алгебраической проблемы собственных значений с разреженными матрицами. *Кибернетика и системный анализ.* 2017. Т. 53, № 6. С. 132–146.
15. Хіміч О.М., Сидорук В.А. Дрібноплитковий гібридний алгоритм факторизації розрідженої матриці. Матеріали Всеукраїнської науково-практичної конференції за міжнародною участю «Інформатика та системні науки (ІСН-2016)» (Полтава, 19–21 березня 2016 р.). С. 326–328.
16. Великоиваненко Е.А., Миленин А.С., Попов А.В., Сидорук В.А., Химич А.Н. Методы и технологии высокопроизводительных вычислений для математического моделирования напряженно-деформированного состояния конструкций с учетом вязкого разрушения. *Международный научно-технический журнал «Проблемы управления и информатики»*. 2014. № 6. С. 42–52.
17. Sergienko I.V., Deineka V.S. Solving combined inverse problems for multicomponent parabolic distributed systems. *Cybernetics and Systems Analysis*. 2007. Vol. 43, N 5. P. 655–674.
18. Velikoivanenko E.A., Milenin A.S., Popov A.V., Sidoruk V.A., Khimich A.N. Methods of numerical forecasting of serviceability of welded structures on computers of hybrid architecture. *Cybernetics and Systems Analysis.* 2019. Vol. 53, N 1. P. 117–127.
19. Баранов А.Ю., Слободян Я.Е., Попов А.В., Химич А.Н. Математическое моделирование прочности строительных конструкций на гибридных вычислительных системах. *Международный научно-технический журнал «Проблемы управления и информатики»*. 2017. № 4. С. 68–81.
20. Попов О.В. Комп'ютерне дослідження достовірності розв'язків узагальненої алгебраїчної проблеми власних значень. *Компьютерная математика*. 2012. №. 1. С. 52–59.
21. The SuiteSparse Matrix Collection. URL: https://cise.ufl.edu/research/sparse/matrices/.
22. Суперкомпьютерный комплекс СКИТ. URL: http://icybcluster.org.ua.

23. Химич А.Н., Декрет В.А., Попов А.В., Чистяков А.В. Численное исследование устойчивости композитных материалов на компьютерах гибридной архитектуры. *Международный научно-технический журнал «Проблемы управления и информатики»*. 2018. № 4. С. 73–88.

24. Быстров В.М., Декрет В.А., Зеленский В.С. Численное исследование устойчивости слоистого композитного материала при сжатии поверхностной нагрузкой. *Проблеми обчислювальної механіки і міцності конструкцій*. 2018. Вип. 28. С. 23–33.

25. MATLAB for deep learning. URL: https://mathworks.com/.

Надійшла до редакції 15.07.2020

О.М. Хіміч, О.В. Попов, О.В. Чистяков, В.А. Сидорук

ПАРАЛЕЛЬНИЙ АЛГОРИТМ РОЗВ'ЯЗУВАННЯ ЧАСТКОВОЇ ПРОБЛЕМИ ВЛАСНИХ ЗНАЧЕНЬ ДЛЯ БЛОЧНО-ДІАГОНАЛЬНИХ МАТРИЦЬ З ОБРАМЛЕННЯМ

Анотація. Запропоновано гібридний алгоритм методу ітерацій на підпросторі розв'язання часткової узагальненої проблеми власних значень для симетричних додатно означених розріджених матриць блочно-діагональної структури з обрамленням на гібридних комп'ютерах з графічними процесорами. Наведено коефіцієнти ефективності алгоритму. Виконано апробацію розробленого алгоритму на тестових та практичних задачах.

Ключові слова: алгебраїчна проблема власних значень, комп'ютер гібридної архітектури, гібридний алгоритм, метод ітерацій на підпросторі, ефективність паралельних алгоритмів, дрібноплитковий алгоритм.

O.M. Khimich, O.V. Popov, O.V. Chistyakov, V.A. Sidoruk

A PARALLEL ALGORITHM FOR SOLVING THE PARTIAL EIGENVALUE PROBLEM FOR BLOCK-DIAGONAL BORDERED MATRICES

Abstract. A hybrid algorithm of the iteration method for the subspace of solution of a partial generalized eigenvalue problem for symmetric positive definite sparse matrices of block-diagonal structure with bordering on hybrid computers with graphic processors is proposed, efficiency coefficients of the algorithm are obtained, and the algorithm is tested against test and practical problems.

Keywords: algebraic eigenvalue problem, computer of hybrid architecture, hybrid algorithm, subspace iteration method, efficiency of parallel algorithm.

Химич Александр Николаевич,
чл.-кор. НАН Украины, доктор физ.-мат. наук, профессор, заместитель директора Института кибернетики им. В.М. Глушкова НАН Украины, Киев, e-mail: khimich505@gmail.com.

Попов Александр Владимирович,
кандидат физ.-мат. наук, старший научный сотрудник Института кибернетики им. В.М. Глушкова НАН Украины, Киев, e-mail: alex50popov@gmail.com.

Чистяков Алексей Валерьевич,
научный сотрудник Института кибернетики им. В.М. Глушкова НАН Украины, Киев, e-mail: alexej.chystyakov@gmail.com.

Сидорук Владимир Антонович,
кандидат физ.-мат. наук, старший научный сотрудник Института кибернетики им. В.М. Глушкова НАН Украины, Киев, e-mail: wolodymyr.sydoruk@gmail.com.