ПРОГРАМНО-ТЕХНІЧНІ КОМПЛЕКСИ

А.Н. ХИМИЧ, А.В. ПОПОВ, А.В. ЧИСТЯКОВ

УДК 519.6

ГИБРИДНЫЕ АЛГОРИТМЫ РЕШЕНИЯ АЛГЕБРАИЧЕСКОЙ ПРОБЛЕМЫ СОБСТВЕННЫХ ЗНАЧЕНИЙ С РАЗРЕЖЕННЫМИ МАТРИЦАМИ

Аннотация. Предлагаются гибридные алгоритмы решения частичной обобщенной проблемы собственных значений для симметричных положительно-определенных разреженных матриц различной структуры на гибридных компьютерах с графическими процессорами. Получены коэффициенты эффективности алгоритмов, проведена апробация разработанных алгоритмов на тестовых и практических задачах.

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

ВВЕДЕНИЕ

Большое количество практических задач, в частности исследование устойчивости конструкций, расчет динамики напряженно-деформированного состояния объектов различной природы и другие, сводятся к решению частичной алгебраической проблемы собственных значений (АПСЗ) с разреженными симметричными положительно-определенными матрицами [1].

Разработка параллельных алгоритмов решения АПСЗ основана на известных, хорошо зарекомендовавших последовательных алгоритмах. В [2, 3] приведены явные и неявные одношаговые итерационные методы для решения частичной проблемы собственных значений, среди которых метод наискорейшего спуска, метод обратных итераций, степенной метод и попеременно-треугольный метод. В статье [4] рассматривается обобщенный метод сопряженных градиентов на основе метода симметричной верхней релаксации для решения систем линейных алгебраических уравнений. В работе [5] изложены теоретические аспекты метода итераций на подпространстве, метода Ланцоша для поиска нескольких минимальных собственных значений.

Требования к высокопроизводительным вычислениям намного опережают возможности традиционных параллельных компьютеров, даже несмотря на многоядерность процессоров. На сегодняшний день эта проблема решается использованием гибридных компьютеров, сочетающих MIMD- и SIMD-архитектуры, которые реализуют вычисления на компьютерах с многоядерными процессорами (CPU) и графическими ускорителями (GPU). Наиболее полно использовать потенциал таких компьютеров можно только при наличии алгоритмического и программного обеспечения, которые учитывают не только свойства задачи, но и особенности гибридной архитектуры.

В настоящее время наиболее известные алгоритмы и программы для решения задач линейной алгебры для различных типов параллельных компьютеров

разработаны под руководством профессора Д. Донгарра [6, 7]. Для решения АПСЗ с большими разреженными матрицами известна параллельная библиотека программ SLEPc (Scalable Library for Eigenvalue Problem Computations) [8]. Библиотека LIS (Library of Iterative Solvers) [9] позволяет решать задачи линейной алгебры, в частности системы линейных алгебраических уравнений и АПСЗ, с матрицами различных форматов на параллельных компьютерах с использованием арифметики расширенной разрядности.

В [10] рассматриваются некоторые особенности разработки параллельных алгоритмов метода итераций на подпространстве для компьютеров различных архитектур. В [11] формируются параллельные алгоритмы для попеременно-треугольного метода.

О НЕКОТОРЫХ ПРОБЛЕМАХ СОЗДАНИЯ ГИБРИДНЫХ АЛГОРИТМОВ

В настоящей статье предложены и исследованы гибридные алгоритмы для решения обобщенной проблемы собственных значений

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

где A, B — симметричные положительно-определенные разреженные матрицы порядка n, для компьютеров с многоядерными процессорами и графическими ускорителями. Рассматривается подход к созданию гибридных алгоритмов, базирующийся на структурной регуляризации разреженной матрицы общего вида за счет переупорядочения переменных, приведении ее к профильному, блочно-диагональному с окаймлением или ленточному виду с дальнейшим использованием гибридных алгоритмов на основе метода итераций на подпространстве и неявного метода сопряженных градиентов.

Основные проблемы создания эффективных гибридных алгоритмов связаны с согласованием распределения вычислений на CPU и GPU, а также с оптимизацией накладных расходов на коммуникационные взаимодействия между ними. Под эффективным алгоритмом решения задачи понимаем алгоритм, который обеспечивает достоверность решения при оптимальном использовании компьютерных ресурсов.

В приведенных ниже рассуждениях (если не оговорено иное) используется архитектура гибридного компьютера: каждый из p вычислительных процессов на CPU взаимодействует с одним (своим) GPU. Такую архитектуру часто обозначают $p\,\mathrm{CPU} + p\,\mathrm{GPU}$.

Эффективность методов решения большинства задач линейной алгебры, в том числе и АПСЗ, в большой степени зависит от эффективного выполнения матрично-матричных и матрично-векторных операций, а также решения систем линейных алгебраических уравнений (СЛАУ). Для выполнения этих операций целесообразно использовать GPU, максимальная производительность которых достигается при выполнении однородных математических операций над большими объемами данных в условиях полной загрузки каждого GPU.

Сформулируем такие требования к созданию эффективных гибридных алгоритмов решения АПСЗ:

- декомпозиция на подзадачи для выполнения на CPU и GPU;
- эффективный способ факторизации разреженных матриц;
- равномерная загрузка процессов CPU и синхронизация обменов между ними;
- решение подзадач, требующих наибольших затрат компьютерного времени на GPU;
- минимизация обменов между процессами CPU, а также между CPU и GPU.

При распределении матриц между процессами на CPU важно предусмотреть эффективную компьютерную конфигурацию (топологию) из процессов и их равномерную загрузку (балансировку нагрузки), чтобы избежать эффекта Гайдна [12], который снижает эффективность выполнения параллельного алгоритма. Как показали исследования, блочно-циклическая декомпозиция элементов матрицы между процессами CPU, а также топология связей «гиперкуб» между используе-

мыми вычислительными устройствами являются наиболее эффективными для рассматриваемых гибридных алгоритмов.

ГИБРИДНЫЙ АЛГОРИТМ МЕТОДА ИТЕРАЦИЙ НА ПОДПРОСТРАНСТВЕ

Метод итераций на подпространстве. Этот метод используется для нахождения r минимальных собственных значений и соответствующих им собственных векторов задачи (1) с ленточной симметричной матрицей. Данный метод является обобщением метода обратных итераций и заключается в построении для задачи (1) последовательности подпространств E_t $(t = 1, 2, \ldots)$, которая сходится к подпространству E_∞, содержащему искомые собственные векторы [5]. На t-й итерации вычисляется ортогональный базис подпространства E_t, и если достигнута требующаяся точность приближенного решения, определяются искомые собственные пары.

Таким образом, реализация метода итераций на подпространстве сводится к решению для $t = 1, 2, \ldots$ следующих подзадач:

- нахождение решения СЛАУ:

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

- вычисление прямоугольной матрицы:

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

- вычисление проекций матриц A и B на подпространство E_t:

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

- 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 c итераций 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) принимаются $\lambda_i^* = \lambda_i^{(c+1)} (i = 1, 2, \ldots, r)$ и первые r столбцов матрицы $X^* = X_{c+1} Z_{c+1}$ (имеется в виду, что собственные значения упорядочены по возрастанию: $0 < \lambda_1 \le \lambda_2 \le \ldots \le \lambda_r \le \ldots$).

Как известно [5], итерационный процесс сходится линейно, причем скорость сходимости λ_i определяется отношением λ_q / λ_1, где q — размерность итерируемого подпространства E_q. В последовательных реализациях алгоритма рекомендуется выбирать $q = \min(2r, r+8)$.

Поскольку на каждой итерации выполняется решение СЛАУ (2) с одной и той же матрицей A, то треугольное разложение, например LL^T, этой матрицы выполняется один раз до начала итерационного процесса.

Распределение данных между процессорными устройствами. Для решения задачи (1) матрица A разбивается на квадратные блоки $A_{I,J}$ порядка s. Элементы главной диагонали и нижнего или верхнего треугольника (в зависимости от используемых алгоритмов) ненулевых блоков этой разреженной симметричной матрицы распределяются между CPU в соответствии с одномерной блочно-циклической схемой. Согласно этой схеме блок $A_{I,J}$ хранится в CPU с логическим номером $(I + l) \bmod p$ (результат операции $k \bmod j$ — остаток от деления k на j, $-1 \le l \le p-2$ — сдвиг, обычно $l = -1$). Аналогично распределяются блоки нижней треугольной матрицы L или верхней треугольной матрицы L^T. Таким образом, имеем строки квадратных матричных блоков порядка s.

Такая же блочно-циклическая схема распределения используется для элементов матрицы B и прямоугольных матриц итерируемых векторов X_t, Y_t, W_t. При этом достаточно распределять и хранить только ненулевые элементы матрицы B в такой последовательности: поддиагональные, диагональные и наддиагональные. Это значительно упрощает алгоритм умножения такой матрицы на прямоугольную, несущественно увеличивая общий объем данных.

В памяти GPU хранятся копии тех блоков (в соответствии с распределением между CPU) матриц A, B, X_t, Y_t, W_t, которые используются для выполнения операций на GPU в соответствии с используемым вариантом гибридного алгоритма (выбор варианта определяется параметрами решаемой задачи и техническими особенностями GPU).

Распределение вычислений между процессорными устройствами. В соответствии с изложенным выше решение частичной АПСЗ (1) методом 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_t из (4) была положительно- определенной. Эта операция выполняется на CPU, причем без обменов данными между процессами, используя, например, предложенный в [10] параллельный алгоритм.

2. LL^T-факторизация разреженной симметричной положительно-определенной матрицы A (на CPU и GPU), используя, например, гибридный плиточный алгоритм [13] для случая ленточной матрицы.

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

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

• вычисления (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 выполняются или на CPU, или на CPU и GPU, или 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 больше одного GPU) проводится каждым процессом на CPU; такой подход существенно сокращает объем данных, которыми обмениваются CPU и GPU; на GPU вычисление произведений прямоугольных матриц 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; в таком случае отсутствует необходимость обменов 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^*), причем нет необходимости в обменах данными между процессорными устройствами.

4. Исследование достоверности результатов — вычисление оценок погрешности полученного решения (см., например, [10]). Для определения оценок относительной погрешности вычисленных приближений к собственным значениям выполняются операции, аналогичные операциям, выполняемым в итерационном процессе. Для реализации этих операций используются те же гибридные алгоритмы, что и в итерационном процессе.

Эффективность гибридного алгоритма. Исследование эффективности предложенного гибридного алгоритма базируется на методологиях из [10, 15]. Как оговорено выше, рассматривается компьютер гибридной архитектуры с p CPU и p GPU.

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

$$T_1 = O_1 t_C + O_{1G} t_G / n_o + O_{oG} t_{oG} + O_{cG} t_{cG},$$
$$T_p = O_p t_C + O_{pG} t_G / n_o + O_o t_o + O_{oG} t_{oG} + O_c t_c + O_{cG} t_{cG}, \tag{7}$$

где t_C, t_G — среднее время выполнения на CPU и GPU соответственно одной арифметической операции с плавающей запятой; n_o — количество таких операций, которые могут быть одновременно выполнены на GPU; t_o, t_{oG} — время, необходимое для обмена одним машинным словом между MPI-процессами на CPU или между CPU и GPU соответственно; O_o, O_{oG} — объем (количество машинных слов) обменов, выполняемых одним CPU и GPU соответственно; t_c, t_{cG} — время, необходимое для синхронизации двух MPI-процессов или MPI-процесса и его GPU соответственно; O_c, O_{cG} — количество таких синхронизаций, выполняемых одним MPI-процессом. При использовании гибридных архитектур в формулах (7) необходимо учитывать синхронность или асинхронность работы CPU и GPU. При асинхронной работе следует суммирование заменять определением максимального значения.

При исследовании и решении частичной АПСЗ (1) с симметричными матрицами 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. Число 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

$$S_p = \frac{T_1}{T_p} = \frac{T_p^{(F)}}{T_p} S_p^{(F)} + \frac{c_I T_p^{(1I)}}{T_p} S_p^{(It)}, \tag{8}$$

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 CPU + p GPU; $S_p^{(F)}$, $S_p^{(It)}$ — коэффициенты ускорения алгоритмов, используемых для решения этих подзадач.

Далее, введем для архитектуры p CPU + p GPU 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 (используя вычисленное ранее LL^T-разложение матрицы системы); $T_p^{(Fo)}$, $T_p^{(Io)}(k)$ — соответственно времена выполнения остальных операций при решении подзадач с фиксированным числом арифметических операций и выполнении одной итерации. Тогда $T_p^{(F)}$ и $T_p^{(1I)}$ можно представить в следующем виде:

$$T_p^{(F)} = T_p^{(LLT)} + T_p^{(SS)}(q) + T_p^{(Fo)}, \quad T_p^{(1I)} = T_p^{(SS)}(q) + T_p^{(Io)}(q). \tag{9}$$

Заметим, что здесь и далее формулы для всех $T_1^{(*)}$ можно получить, подставив в формулы для $T_p^{(*)}$ значение $p=1$, при этом для всех таких времен справедливы формулы (7). Тогда коэффициент ускорения гибридного алгоритма метода итераций на подпространстве для архитектуры $p\,\mathrm{CPU}+p\,\mathrm{GPU}$, базируясь на (8), можно вычислить по формуле

$$S_p=\frac{T_p^{(F)}}{T_p}\left(\alpha_p^{(LLT)}S_p^{(LLT)}+\alpha_p^{(SS)}(q)S_p^{(SS)}(q)+\alpha_p^{(Fo)}S_p^{(Fo)}\right)+$$
$$+\frac{c_I T_p^{(1I)}}{T_p}\left(\beta_p^{(SS)}(q)S_p^{(SS)}(q)+\beta_p^{(Io)}(q)S_p^{(Io)}(q)\right), \tag{10}$$

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 в (9) для времен $T_p^{(*)}$.

Далее рассмотрим случай, когда матрицы задачи (1) являются ленточными. Введем обозначения: m_A, m_B — полуширина ленты матриц A и B соответственно (будем считать, что $n=O(10^k m_A)$, $k>1$, $m_A\ge m_B$ и $m_A>>q$); η_B — среднее количество ненулевых элементов в одной строке матрицы B (как правило, $\eta_B<<m_B$). При $p>1$ обмены между MPI-процессами (CPU) — это операции мультирассылки и мультисборки, для реализации которых предполагается использование алгоритма дерева. Также введем времена для обмена массивом из d двойных слов между парой CPU: $t_C^{(E)}(d)=t_s+dt_o$, а для архитектуры CPU + GPU имеем $t_G^{(E)}(d)=t_{sG}+dt_{oG}$.

Для случая использования гибридных алгоритмов решения СЛАУ с симметричной ленточной матрицей (см. [13, 14]) можно сформулировать лемму (пренебрегая величинами самых малых порядков).

Лемма 1. При условии $n>>q>r$ и $m_A>sp$ для времен, затрачиваемых гибридным алгоритмом на LL^T-разложение ленточной симметричной матрицы и на решение двух СЛАУ с нижней и верхней треугольными ленточными матрицами на архитектуре $p\,\mathrm{CPU}+p\,\mathrm{GPU}$, справедливы оценки

$$T_p^{(LLT)}\approx\frac{n}{p}\left(\frac{s^2}{3}pt_C+\max\left\{m_A^2\,\frac{t_G}{n_o},\,t_C^{(E)}(sm_A)\frac{p}{s}\log_2 p\right\}+\right.$$
$$\left.+2psm_A\,\frac{t_G}{n_o}+t_G^{(E)}(sm_A)\frac{p}{s}\right), \tag{11}$$

$$T_p^{(SS)}(k)=$$
$$=\frac{n}{p}\left(kt_C\,p\log_2 p+(4m_A+2ps)k\,\frac{t_G}{n_o}+(t_C^{(E)}(sk)\log_2 p+2t_G^{(E)}(sk))\frac{2p}{s}\right). \tag{12}$$

Используя (11), (12) и учитывая возможность одновременного выполнения ряда операций вычислений и обменов на CPU и GPU или разными потоками на GPU, а также пренебрегая слагаемыми меньшего порядка, получаем следующие оценки коэффициентов ускорения гибридных алгоритмов LL^T-разложения ленточной симметричной матрицы и решения СЛАУ вида (2) на архитектуре $p\,\mathrm{CPU}+p\,\mathrm{GPU}$:

$$S_p^{(LLT)}\approx p\left(1-\frac{2s(p-1)}{m_A+2ps}\right),\ t_G^{(m)}(s)\ge t_C^{(E)}(sm_A),$$

$$S_p^{(LLT)} \approx p\left(\frac{2ps}{m_A+2s}+\frac{p\log_2 p}{m_A s(m_A+2s)}\frac{n_o t_C^{(E)}(sm_A)}{t_G}\right)^{-1}, \tag{13}$$

$$t_G^{(m)}(s) < t_C^{(E)}(sm_A),\ t_G^{(m)}(s)=\frac{m_A^2 s}{p\log_2 p}\frac{t_G}{n_o};$$

$$S_p^{(SS)}(k) \approx p\left(1+\frac{(p-1)s}{2m_A+s}+\frac{p\log_2 p}{ks(2m_A+s)}\frac{n_o t_C^{(E)}(ks)}{t_G}\right)^{-1}.$$

При получении оценок $T_p^{(Fo)}$ и $T_p^{(Io)}(k)$ необходимо учитывать, что в общем случае наибольшее количество арифметических операций выполняется при вычислении произведения разреженной матрицы B на $(n\times k)$-матрицу и при вычислении матриц или векторов, элементами которых являются скалярные произведения n-мерных векторов. Количество остальных арифметических операций, выполняемых одним из p GPU, намного меньше и им можно пренебречь. Тогда справедливо следующее утверждение.

Лемма 2. При условиях $n>>q>r$, $m_A \geq m_B$ и $1\leq \eta_B << m_B$ для времен $T_p^{(Fo)}$ и $T_p^{(Io)}(k)$ справедливы оценки

$$T_p^{(Fo)} = \frac{n}{p}\left(\frac{n+2(n+2)p\log_2 p}{n} qt_C + (2\eta_B+6)q\frac{t_G}{n_o}\right)+$$
$$+(2t_C^{(E)}(sq)\log_2 p + 2t_G^{(E)}(sq))\frac{n}{s}, \tag{14}$$

$$T_p^{(Io)}(k) = \frac{n}{p}\left(\frac{(n+2k)\log_2 p+O(k^2)}{n} pkt_C + (2\eta_B+4k)k\frac{t_G}{n_o}\right)+ \tag{15}$$
$$+(t_C^{(E)}(sk)\log_2 p + 2t_G^{(E)}(sk))\frac{n}{s}.$$

Из (14) видно, что время $T_p^{(Fo)}$ существенно меньше значений $T_p^{(LLT)}$ и $T_p^{(SS)}(q)$ из (9), поэтому им можно пренебречь, положив в (10) $\alpha_p^{(Fo)} \cong 0$.

Влияние $T_p^{(Io)}(q)$ зависит от соотношения m_A, η_B и q: если q существенно больше η_B, то в (15) можно пренебречь слагаемым, содержащим η_B; если $m_A >> \eta_B$ и $m_A >> q$, то в (9) можно пренебречь $T_p^{(Io)}(q)$. Из (15) также следует оценка

$$S_p^{(Io)}(k) \approx p\left(1+\frac{p\log_2 p}{2k(\eta_B+2k)}\frac{n_o}{t_G}(kt_C + t_C^{(E)}(sk))\right)^{-1}. \tag{16}$$

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

Теорема 1. Для гибридного алгоритма метода итераций на подпространстве при условиях $n>>q, m_A \geq m_B, m_A >> q, m_A > sp$ и $\eta_B << m_B$ имеет место оценка коэффициентов ускорения

$$S_p = \frac{\gamma_p}{\gamma_p + c_I}(\alpha_p^{(LLT)} S_p^{(LLT)} + \alpha_p^{(SS)}(q) S_p^{(SS)}(q)) +$$
$$+\frac{c_I}{\gamma_p + c_I}(\beta_p^{(SS)}(q) S_p^{(SS)}(q) + \beta_p^{(Io)}(q) S_p^{(Io)}(q)), \tag{17}$$

где $\gamma_p = (T_p^{(LLT)} + T_p^{(SS)}(q))/T_p^{(1I)}$; $T_p^{(1I)}$ представлено в (9); для вычисления коэффициентов $\alpha_p^{(*)} = T_p^{(*)}/T_p^{(F)}$ и $\beta_p^{(*)} = T_p^{(*)}/T_p^{(1I)}$ используются форму-

лы (11), (12), (14), (15), причем в полученных выражениях малые по отношению к другим слагаемые отбрасываются; коэффициенты $S_p^{(*)}$ ускорения представлены формулами (13) и (16).

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

Анализ полученных оценок (11)–(17) свидетельствует, что эффективность предложенного алгоритма в значительной мере определяется эффективностью алгоритма LL^T-разложения, например гибридного плиточного алгоритма [16], а также эффективностью алгоритмов решения СЛАУ с нижней и верхней разреженными треугольными матрицами. Следует заметить, если время, затраченное на пересылку данных между вычислительными устройствами, будет меньше времени вычислений, то эти алгоритмы ввиду использования асинхронности будут иметь коэффициенты эффективности, близкие к единице.

На эффективность гибридного алгоритма решения частичной обобщенной АПСЗ методом итераций на подпространстве существенное влияние оказывает количество c_I выполненных итераций для достижения необходимой точности. Обычно при постоянном r значение c_I тем меньше, чем больше размерность итерируемого подпространства q. Как следует из оценок (12) и (15), объем вычислений пропорционален q. Однако в ряде случаев при относительно малых значениях q возникает недостаточная загруженность GPU (т.е. фактически уменьшается значение n_o) при увеличении количества обменов, что приводит к снижению реальной эффективности алгоритма. В таких случаях увеличение размерности итерируемого подпространства (до момента достижения максимального значения n_o) позволяет сократить время решения задачи за счет сокращения количества итераций.

Экспериментальное исследование эффективности гибридного алгоритма. Экспериментальное исследование проводилось на ленточных матрицах различных порядков, были также использованы матрицы из коллекции Флоридского университета [17] (табл. 1).

На рис. 1, *а* даны графики зависимости ускорения от количества GPU. Проведенные эксперименты свидетельствуют об обеспечении хорошей масштабируемости, т.е. время решения задачи пропорционально уменьшается с ростом количества вычислительных устройств. Также обеспечивается более высокая эффективность для матриц с большой полушириной ленты (эффективность алгоритма улучшается с увеличением полуширины ленты исходных матриц). На рис. 1, *б* приведена диаграмма с указанием ускорения алгоритма итераций при разном количестве и типах плиток. Использование плиток оптимального размера в LL^T-разложении и в итерационном процессе позволяет получить хорошую сбалансированность загрузки используемых вычислительных устройств, существенно сократить время решения и повысить эффективность вычислений. В зависимости от типа вычислительного устройства и его характеристик рекомендуемый порядок плиток составляет 512 для CPU и 192÷256 для GPU. Разработанный гибридный алгоритм позволяет эффективно использовать современные вычислительные устройства различного типа и адаптироваться к структуре матрицы задачи.

Таблица 1. Тестовые ленточные матрицы из коллекции Флоридского университета

Название задачи	Проблемная область	Порядок матрицы (полуширина ленты)
G2_circuit	Structural problem	150 102 (65 956)
Bone010	Model reduction problem	986 703 (26 572)
Emila_923	Structural problem	923 136 (34 175)

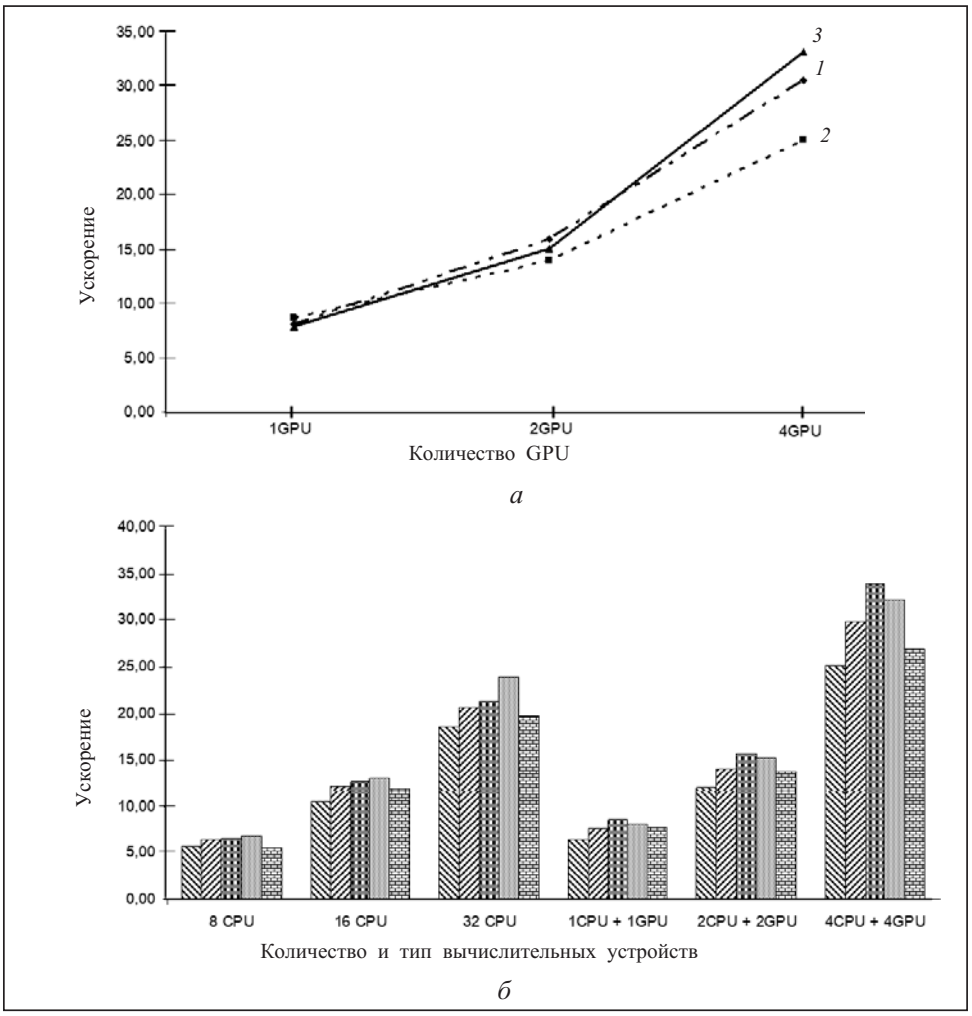

Рис. 1. а — графики ускорения гибридного алгоритма итераций на подпространстве при использовании разного числа GPU для задач G2_circuit (*1*), Bone010 (*2*), Emila_923 (*3*); *б* — диаграмма с указанием ускорения алгоритма итераций на подпространстве при разных количестве и типах вычислительных устройств для блоков — размер 64, — размер 128, — размер 192, — размер 256, — размер 512

ГИБРИДНЫЙ АЛГОРИТМ МЕТОДА СОПРЯЖЕННЫХ ГРАДИЕНТОВ

Во многих практических расчетах возникает обобщенная АПСЗ (1) с диагональной положительно-определенной матрицей B. Эта задача легко сводится к стандартной АПСЗ — задаче на собственные значения симметричной положительно-определенной матрицы $C = B^{-1/2} A B^{-1/2}$: $Cy = \lambda y$, где $y = B^{1/2} x$. При этом разреженная структура исходной матрицы остается неизменной. Далее рассмотрим итерационный метод решения такой задачи.

Обобщенный метод сопряженных градиентов для решения частичной проблемы собственных значений

$$Ax = \lambda x \tag{18}$$

представлен в [4] следующими формулами:

$$x^{k+1} = \begin{cases} x^k + a_k p^k, \ k \neq N-1, \ 2N-1, \ldots, \\ \dfrac{x^k + a_k p^k}{\|x^k - a_k p^k\|_2}, \ k = N-1, \ 2N-1, \ldots, \end{cases} \tag{19}$$

$$p^k = Z(x^k) f(x^k) - \beta_k p^{k-1};\ f(x) = [A - \mu(x) I] x / \|x\|_2 , \qquad (20)$$

$$\beta_k = \begin{cases} 0, \quad k = 0,\ 2N,\ \ldots, \\ \dfrac{(Z(x^k) f(x^k), [A - \mu(x^k) I] p^{k-1})}{(p^{k-1},\ [A - \mu(x^k) I] p^{k-1})},\ k = 1, 2, 3, \ldots, \end{cases} \qquad \mu(x) = \frac{(Ax, x)}{(x, x)} .$$

Здесь параметр α_k может быть выбран как точка локального минимума функционала $\mu(x^k - \alpha p^k)$, N — момент восстановления, матрица $Z(x^k)$ выбирается из условий ускорения сходимости итерационного процесса и особенностей архитектуры параллельного компьютера. Например, $Z(x)$ можно выбрать на основе метода верхней симметричной релаксации для решения систем линейных алгебраических уравнений. В этом случае если исходную матрицу представить в виде

$$A = I - L - L^T$$

через единичную, нижнюю и верхнюю треугольные матрицы, то

$$Z(x) = \omega(2 - \omega)(I - \omega \hat{L}^T(x))^{-1} (I - \omega \hat{L}(x))^{-1} , \qquad (21)$$

где $\hat{L}(x) = \dfrac{1}{1 - \mu(x)} L$, ω — параметр релаксации. В [4] установлена сходимость итерационного процесса и обоснован выбор параметра ω.

Рассмотрим гибридный алгоритм метода сопряженных градиентов для разреженных матриц. Идеологической предпосылкой использования гибридных вычислений при обработке разреженных матриц произвольной структуры является предварительное приведение структуры исходной матрицы к разреженному блочно-диагональному виду с окаймлением с помощью метода параллельных сечений:

$$\tilde{A} = P^T A P = \begin{pmatrix} D_1 & 0 & 0 & \ldots & 0 & C_1 \\ 0 & D_2 & 0 & \ldots & 0 & C_2 \\ 0 & 0 & D_3 & \ldots & 0 & C_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & D_{p-1} & C_{p-1} \\ C_1 & C_2 & C_3 & \ldots & C_{p-1} & D_p \end{pmatrix},$$

где P — матрица перестановок, а блоки D_i и C_i сохраняют разреженность, p — количество диагональных блоков в матрице. Естественным для метода параллельных сечений является то, что порядок последнего диагонального блока намного меньше, чем порядки других диагональных блоков.

Таким образом, исходная задача (18) сводится к эквивалентной задаче: $\tilde{A} y = \lambda y$.

Для хранения данных с учетом структуры полученной матрицы $\tilde{A}$ целесообразно использовать блочное распределение, причем количество блоков следует выбирать с учетом количества процессоров, которые задействованы в расчетах. Реализуется такое распределение блоков (подматриц) между процессами на CPU: процессы с номерами $0 \le i < p - 1$ сохраняют блоки D_{i+1} и C_{i+1}, а процесс с номером $p - 1$ сохраняет блок D_p. Здесь p — общее количество процессов.

Параллельная реализация гибридного алгоритма метода сопряженных градиентов на основе метода симметричной верхней релаксации для разреженной блочно-диагональной матрицы с окаймлением будет определяться в соответствии с (19)–(21) в основном решением двух систем:

$$(I - \omega \tilde{\hat{L}}(x)) y = z(x),\ (I - \omega \tilde{\hat{L}}^T(x)) w = y(x),$$

где $z(x) = \omega(2 - \omega) f(x)$; при этом $w^k = p^k - \beta_k p^{k-1}$.

Разреженная блочно-треугольная матрица с окаймлением имеет вид

$$\widetilde{\widetilde{L}} = \begin{pmatrix} \widetilde{L}_1 \\ \widetilde{L}_2 \\ \widetilde{L}_3 \\ \vdots \\ \widetilde{L}_{p-1} \\ \widetilde{L}_p \end{pmatrix} = \begin{pmatrix} \widetilde{D}_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & \widetilde{D}_2 & 0 & \dots & 0 & 0 \\ 0 & 0 & \widetilde{D}_3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \widetilde{D}_{p-1} & 0 \\ C_1 & C_2 & C_3 & \dots & C_{p-1} & \widetilde{D}_p \end{pmatrix}.$$

Алгоритм решения системы $(I - \omega\widetilde{L}(x))y = z$ с нижней треугольной матрицей сводится к одновременному и независимому решению треугольных систем каждым процессом (кроме последнего): $(I - \omega\widetilde{D}_q(x))y_q = z_q$; q $(1 \le q < p)$ — номер процесса (CPU), и последующим вычислением $\widetilde{y}_q = C_q(x)y_q$.

Далее все процессы пересылают $\widetilde{y}_q$ последнему (p-му) процессу, который вычисляет вектор y_p, решая систему $(I - \omega\widetilde{D}_p(x))y_p = z_p - \sum_{q-1}^{p-1}\widetilde{y}_q$.

Аналогично решается система $(I - \omega\widetilde{L}^T)w = y$. При этом p-й процесс решает систему $(I - \omega\widetilde{D}_p^T(x))w_p = y_p$, а затем рассылает соответствующие компоненты w_p другим процессам, которые независимо решают системы $(I - \omega\widetilde{D}_q^T(x))w_q = y_q - C_q^T(x)w_p$.

В гибридном алгоритме метода сопряженных градиентов наиболее сложными арифметическими операциями по времени выполнения являются умножение разреженной матрицы на плотный вектор, а также решение двух систем с блочно-треугольными разреженными матрицами.

Для определения эффективности гибридного алгоритма, если не оговорено иное, воспользуемся обозначениями, представленными ранее, и дополнительно введем следующие обозначения: $nz(D_i)$ — количество ненулевых элементов в i-м блоке треугольной матрицы, n_i — порядок соответствующего блока.

Для нахождения ускорения используем формулу $S_p = T_1 / T_p$. При этом учтем, что большинство операций выполняется на GPU и временем синхронизации и пересылками данных между CPU и GPU можно пренебречь. При решении треугольной системы для каждого диагонального блока, кроме последнего, выполняется $2nz(\widetilde{D}_i) + 2nz(C_{pi}) + 3n_i$ арифметических операций, для умножения разреженной матрицы на плотный вектор выполняется $2nz(\widetilde{D}_i) + 2nz(C_{pi}) + n_i$ операций. Следовательно, общее количество арифметических операций, необходимое для реализации одной итерации гибридного алгоритма на одном GPU, можно оценить величиной $\sum_{i=1}^{p-2} O_{kGi}$, где $O_{kGi} = 10nz(\widetilde{D}_i) + 10nz(C_{pi}) + 9n_i$, а затраченное время на одну итерацию составляет $T_1 = \sum_{i=1}^{p-1} O_{kGi}\frac{t_G}{n_o}$; O_k и O_{kG} — количество арифметических операций, выполняемых одним из k CPU и k GPU соответственно.

Время, затраченное на выполнение одной итерации гибридного алгоритма на архитектуре $p\,\mathrm{CPU} + p\,\mathrm{GPU}$, определяется величиной $\max\limits_{1 \le i \le p\text{-}1} O_{kGi}\frac{t_G}{n_o}$. При решении треугольных систем объем передаваемых данных между CPU равен $n_p \log_2 p$, а при умножении разреженной матрицы на плотный вектор составляет $3(n_p \log_2 p)/2$.

Таким образом, для оценки коэффициента ускорения гибридного алгоритма имеет место следующая теорема.

Теорема 2. Если $n_p << \min\limits_{1\le i\le p-1} n_i$, а для операций мультисборки и мультирассылки используется алгоритм дерева, то коэффициент ускорения гибридного алгоритма метода сопряженных градиентов решения задачи на собственные значения блочно-диагональной матрицы с окаймлением (19) составляет

$$S_p \approx \frac{\sum\limits_{i=1}^{p-1} O_{kGi}\,\dfrac{t_G}{n_o}}{\max\limits_{1\le i\le p\text{-}1} O_{kGi}\,\dfrac{t_G}{n_o} + t_{cG}\,n_p \log_2 p + t_{cG}\,3\,(n_p \log_2 p)/2},$$

где $O_{kGi} = 10nz(\widetilde{D}_i) + 10nz(C_{pi}) + 9n_i$.

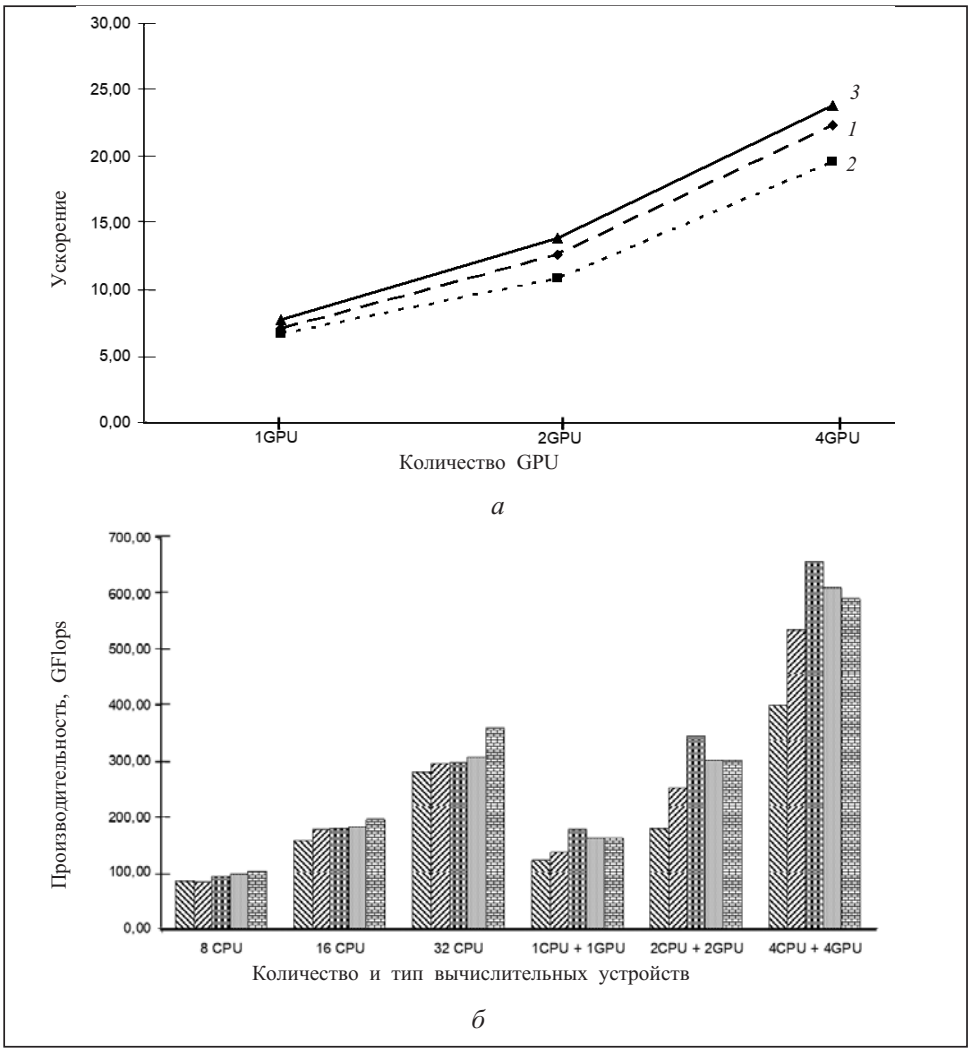

Рис. 2. а — графики ускорения гибридного алгоритма сопряженных градиентов при использовании разного числа GPU для задач G2_circui (*1*), Bone010 (*2*), Emila_923 (*3*); *б* — диаграмма с указанием производительности гибридного алгоритма сопряженных градиентов при разных количестве и типах вычислительных устройств для блоков — размер 64, — размер 128, — размер 192, — размер 256, — размер 512

Как и для алгоритма метода итерации на подпространстве, исследование эффективности проводилось на разреженных матрицах из [17].

На рис. 2, *а* приведены графики ускорения, полученные при решении задачи для различных порядков матрицы при использовании различного количества процессов CPU и GPU. Данные результаты свидетельствуют, что разработанный алгоритм хорошо масштабируется, обеспечивая равномерную загрузку вычислительных устройств вне зависимости от их типа, подтверждая теоретические результаты.

На рис. 2, *б* представлена диаграмма с указанием производительности гибридного алгоритма. Данные результаты позволяют экспериментально доказать, что используемые подходы обеспечивают высокую эффективность распараллеливания вычислительного процесса при использовании нескольких GPU.

ЧИСЛЕННОЕ РЕШЕНИЕ ОДНОЙ ЗАДАЧИ УСТОЙЧИВОСТИ

Апробация разработанных гибридных алгоритмов проводилась на различных практических задачах, в частности решалась задача устойчивости слоистого двухкомпонентного композитного материала регулярной структуры [18] при равномерном одноосном сжатии армирующих слоев поверхностной нагрузкой постоянной интенсивности.

Для исследования устойчивости композитных структур применяется статический метод трехмерной линеаризированной теории устойчивости в рамках второго варианта теории малых докритических деформаций. Докритическое состояние определяется решением плоской задачи линейной теории упругости кусочно-однородных тел для различных граничных условий, которые соответствуют трехслойному элементу периодичности в случае материала и трехслойному элементу структуры с боковыми сторонами, свободными от напряжений, в случае композитного образца.

При использовании статического метода Эйлера задача устойчивости сводится к обобщенной задаче на собственные значения, в которой минимальное собственное значение λ определяет критическую загрузку, а соответствующая собственная функция $u = (u_1, u_2)$ определяет форму потери устойчивости.

Задача решалась с различными исходными данными на персональном суперкомпьютере гибридной архитектуры Инпарком_pg [19] (один вычислительный узел, два четырехядерных процессора Xeon 5606, два GPU Tesla K40). Программы, реализующие гибридные алгоритмы, написаны на языке программирования C++ с использованием на CPU системы MPI и библиотеки программ Intel MKL [20], а для распараллеливания на GPU использовали технологию CUDA, библиотеки программ cuBLAS и cuSPARSE [21]. Ниже приведен листинг протокола решения задачи: порядок матриц A и B: 12282, полуширина ленты матрицы A: 6212, полуширина ленты матрицы B: 71.

Фрагмент протокола решения задачи на Инпарком_pg

```
Problem  solving:  total  time  =  10.0033e+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 вычисленных минимальных собственных значения. Решение этой задачи с помощью пакета MATLAB [22] было получено за 340 с. Время решения задачи гибридным алгоритмом метода итераций на подпространстве на Инпарком_pg составляет 10,0033 с, что приблизительно в 30 раз быстрее, чем с использованием MATLAB.

Проводился сравнительный анализ производительности разработанных гибридных алгоритмов с соответствующими параллельными алгоритмами для

MIMD-компьютеров. Для параллельного алгоритма неявного метода сопряженных градиентов наибольшее ускорение для MIMD-компьютера при использовании восьми процессов получено от пяти до шести раз по сравнению с последовательной версией программы (1 CPU). Для гибридного алгоритма при использовании одного GPU получено ускорение от 6,5 до 7,5 раз, а при использовании двух GPU — от 9 до 11 раз по сравнению с последовательной версией программы. Результаты решения параллельным и гибридным алгоритмами метода итерации на подпространстве показали, что при использовании восьми процессов на CPU получено ускорение в четыре раза по сравнению с последовательной версией программы, а при использовании GPU получено ускорение в шесть и девять раз для одного и двух GPU соответственно.

ЗАКЛЮЧЕНИЕ

Дальнейшее развитие изложенных схем гибридных алгоритмов (разбиение на блоки; распределение между процессорными устройствами; обработка блоков, соответствующих ненулевым блокам) лежит в плоскости расширения типов обрабатываемых структур с разреженными матрицами, таких как, например, профильные.

Проведенная апробация предложенных гибридных алгоритмов на решении практических задач свидетельствует о перспективности этого направления в развитии алгоритмического и программного обеспечения для математического моделирования задач устойчивости материалов и конструкций.

В дальнейшем целесообразно модифицировать предложенный гибридный алгоритм метода итераций на подпространстве, заменив LL^T-разложение разреженной симметричной матрицы ее LDL^T-разложением. Это позволит проводить более глубокое апостериорное исследование результатов решения (например, [6]), а также находить собственные значения (и соответствующие им собственные векторы) в середине спектра.

Исследования разработанных гибридных алгоритмов в дальнейшем будут направлены на их реализацию для графических ускорителей архитектуры Maxwell и Pascal, позволяющих реализовывать динамический параллелизм и непосредственный обмен данными между графическими ускорителями без участия центрального процессора. Кроме того, предполагается оптимизация этих алгоритмов для новых процессоров Intel Xeon Phi второго поколения. Также предполагается адаптация разработанных алгоритмов для использования арифметики повышенной разрядности при решении плохо обусловленных задач.

СПИСОК ЛИТЕРАТУРЫ

1. Писанецки С. Технология разреженных матриц. Москва: Мир, 1988. 410 с.
2. Приказчиков В.Г. Прототипы итерационных процессов в задаче на собственные значения. *Дифференциальные уравнения.* 1980. Т. 16, № 9. С. 1688–1697.
3. Хіміч О.М., Чистяков О.В. Паралельні однокрокові ітераційні методи розв'язання алгебраїчної проблеми власних значень для розріджених матриць. *Комп'ютерна математика.* 2014. № 2. С. 81–88.
4. Савинов Г.В. Исследование сходимости одного обобщенного метода сопряженных градиентов для определения экстремальных собственных значений матрицы. *Записки научного семинара ЛОМИ*. 1981. № 111. С. 145–150.
5. Парлет Б. Симметричная проблема собственных значений. Москва: Мир, 1983. 382 с.
6. NetLib. 2015. URL: http://www.netlib.org/.
7. MAGMA. 2015. URL: http://icl.cs.utk.edu/magma/.
8. SLEPc. 2015. URL: http://slepc.upv.es/.
9. LIS. 2015. URL:http://www.ssisc.org/lis/.
10. 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.
11. Чистяков О.В. Про особливості розробки програмного забезпечення для розв'язання задач на власні значення з розрідженими матрицями на гібридних комп'ютерах. *Комп'ютерна математика.* 2015. № 1. С. 75–84.

12. Михалевич В.С., Молчанов И.Н., Сергиенко И.В. и др. Численные методы для многопроцессорного вычислительного комплекса ЕС. Под ред. И.Н. Молчанова. Москва: ВВИА им. Н.Е. Жуковского, 1986. 401 с.
13. Баранов А.Ю., Попов А.В., Слободян Я.Е., 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.
14. Хіміч О.М., Сидорук В.А. Гібридний алгоритм розв'язування лінійних систем з розрідженими матрицями на основі блочного LLT методу. *Комп'ютерна математика.* 2015. № 1. С. 67–74.
15. Попов О.В. Дослідження ефективності паралельних алгоритмів для комп'ютерів гібридної архітектури. Праці міжнародної наукової школи-семінару «Питання оптимізації обчислень (ПООХLII)», 2015. С. 16.
16. Хіміч О.М., Попов О.В., Баранов А.Ю., Чистяков О.В. Гібридний алгоритм розв'язування задач на власні значення для стрічкових матриць. *Теорія оптимальних рішень*. 2016. С. 86–94.
17. The SuiteSparse Matrix Collection. 2015. URL: https://cise.ufl.edu/research/sparse/matrices/.
18. Декрет В.А., Зеленский В.С., Быстров В.М. Численное исследование устойчивости слоистого композитного материала при одноосном сжатии наполнителя. Киев: Наук. думка, 2008. 248 с.
19. Хіміч О.М., Молчанов І.М. Попов О.В. та ін. Інтелектуальний персональний суперкомп'ютер для розв'язування науково-технічних задач. *Наука і інновації.* 2016. Т. 12, № 4. С. 17–31.
20. Intel Maths Kernel Library. 2015. URL: https://software.intel.com/en-us/mkl.
21. CUDA library. 2015. URL: https://nvidia.com/.
22. MATLAB. 2017. URL https://mathworks.com/.

Надійшла до редакції 13.07.2017

О.М. Хіміч, О.В. Попов, О.В. Чистяков
ГІБРИДНІ АЛГОРИТМИ РОЗВ'ЯЗУВАННЯ АЛГЕБРАЇЧНОЇ ПРОБЛЕМИ ВЛАСНИХ ЗНАЧЕНЬ З РОЗРІДЖЕНИМИ МАТРИЦЯМИ

Анотація. Запропоновано гібридні алгоритми розв'язання часткової узагальненої проблеми власних значень для симетричних додатно-означених розріджених матриць різної структури на гібридних комп'ютерах з графічними процесорами, наведено коефіцієнти ефективності алгоритмів, проведено апробацію розроблених алгоритмів на тестових та практичних задачах.

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

A.N. Khimich, A.V. Popov, O.V. Chistyakov
HYBRID ALGORITHMS FOR SOLVING THE ALGEBRAIC EIGENVALUE PROBLEM WITH SPARSE MATRICES

Abstract. Hybrid algorithms for solving a partial generalized eigenvalue problem for symmetric positive-definite sparse matrices of different structures on hybrid computers with graphic processors are proposed, coefficients for the efficiency of the algorithms are obtained, and approbation of the developed algorithms for test and practical problems is carried out.

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

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