УДК 519.6:539.3

А.Ю. Баранов, А.В. Попов, Я.Е. Слободян, А.Н. Химич

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ПРОЧНОСТИ СТРОИТЕЛЬНЫХ КОНСТРУКЦИЙ НА ГИБРИДНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Введение

Средства численного моделирования различных физических процессов, использующие метод конечных элементов (МКЭ), сегодня применяются при проектировании и разработке в чрезвычайно широком диапазоне — начиная от проектирования корпуса мобильного телефона и заканчивая разработкой сейсмостойких объектов или ядерных реакторов. При этом большую часть времени моделирования занимает нахождение решения систем линейных алгебраических уравнений (СЛАУ), возникающих при дискретизации исходных систем дифференциальных уравнений. Следует заметить, что обычно моделирование предполагает многократное решение СЛАУ, поскольку интерес представляет динамика исследуемого процесса во времени, причем количество итераций может составлять сотни и тысячи. В случае нелинейных задач — это также многократное построение и решение СЛАУ в пределах одного шага по времени.

Таким образом, следует отметить чрезвычайное влияние скорости работы программы решения СЛАУ на общую скорость работы конечно-элементного решателя.

Еще одним важным фактором, определяющим применимость решателя СЛАУ для использования в современных задачах моделирования методом конечных элементов, является используемая схема размещения СЛАУ в оперативной памяти. Задачи проектирования сложных конструкций часто приводят к построению СЛАУ с миллионами (или десятками миллионов) неизвестных. Например, для системы порядка $n = 1217610$, построенной для численного анализа напряженно-деформируемого состояния многоэтажной конструкции, полуширина ленты матрицы после применения обратного алгоритма перенумерации Катхилла–Макки составляла $m = 3628$. Для ее хранения необходим объем оперативной памяти в 33 Гбайт (для 4417489080 элементов, с учетом симметричности матрицы, хранится только ее половина).

При решении СЛАУ большого порядка (10^5–10^8) с полушириной ленты матрицы $O(\sqrt{n})$ (т.е. 10^2–10^4) количество арифметических операций с плавающей запятой (при использовании метода Холецкого) составляет $O(10^9$–$10^{16})$. Общее количество хранимых элементов матрицы системы — 10^8–10^{12}. Такие требования к вычислениям намного опережают возможности традиционных параллельных компьютеров, даже несмотря на многоядерность процессоров.

Решение проблемы ускорения вычислений на многоядерных компьютерах может достигаться использованием сопроцессоров-ускорителей для выполнения больших объемов однородных арифметических операций. Раньше в качестве ускорителей использовались преимущественно графические процессоры (Graphical Processing Unit — GPU). В последнее время более высокая производительность чаще достигается при использовании процессоров Intel Xeon Phi в качестве таких сопроцессоров. Ускорение вычислений в ближайшей перспективе, на наш взгляд, возможно при использовании гибридных систем, которые совмещают многоядерные процессоры (Central Processing Unit — CPU) с сопроцессорами-ускорителями (см. рейтинг Top500 [1]), в частности с GPU.

В статье представлены результаты численного моделирования пространственных строительных конструкций с использованием в качестве процессорной компоненты программного обеспечения на основе нового гибридного алгоритма решения СЛАУ с ленточной симметричной положительно-определенной матрицей, совмещающего вычисления на многоядерных процессорах и графических ускорителях. Приведены многочисленные результаты с использованием для препроцессорной и постпроцессорной обработки гибридного варианта программного комплекса Лира–Сапр [2].

1. Моделирование напряженно-деформированного состояния строительных конструкций

Математические постановки задач. Математически задачи расчета прочности конструкций с использованием принципа возможных перемещений могут быть поставлены в виде следующих вариационных задач [3].

Необходимо найти вектор-функцию $u \in U_0$, удовлетворяющую для любой вектор-функции $v \in U_0$ (любого возможного перемещения) интегральному тождеству:

- для статической задачи

$$a(u, v) = l(f, v); \tag{1}$$

- для динамической задачи

$$a(u, v) + b(u'', v) + c(u', v) = l(f, v), \tag{2}$$

$$u(t_0) = u^{(0)}, \quad u'(t_0) = u^{(1)}; \tag{3}$$

- для задачи о собственных колебаниях

$$a(u, v) = \lambda b(u, v), \tag{4}$$

где U_0 — бесконечномерное функциональное пространство возможных перемещений; симметричные билинейные формы $a(u, v), b(u'', v), c(u', v)$ пропорциональны соответственно потенциальной, кинетической энергиям деформации и энергии торможения, а линейная форма $l(f, v)$ пропорциональна работе приложенных (внешних) усилий при нагружении; u' — первая производная вектор-функции u по времени; u'' — вторая.

Дискретизация задач методом конечных элементов. Теоретической основой Лира–Сапр является метод конечных элементов [4], реализованный в форме перемещений. Выбор именно этой формы объясняется простотой ее алгоритмизации и физической интерпретации, наличием единых методов построения матриц жесткости и векторов нагрузок для различных типов конечных элементов, возможностью учета произвольных граничных условий и сложной геометрии рассчитываемой конструкции.

Для получения с помощью МКЭ дискретной задачи область разбивается на конечные элементы, назначаются узлы и их степени свободы (перемещения и углы поворота узлов). Степеням свободы соответствуют базисные (координатные, аппроксимирующие) вектор-функции φ_i, отличные от нуля только на элементах, которые содержат соответствующий данной степени свободы узел. Кроме того, для степеней свободы и базисных функций справедливы следующие соотношения:

$$L_j(\varphi_i) = \delta_{ij}, \tag{5}$$

где δ_{ij} — символ Кронекера, а результатом операции $L_j(\varphi_i)$ является значение компоненты вектор-функции φ_i для степени свободы L_j.

Приближенные решения соответствующих задач ищутся в конечномерном подпространстве U_0^h пространства U_0. Вектор-функции из подпространства U_0^h являются кусочно-полиномиальными и могут быть представлены в виде удовлетворяющей главным (кинетическим) условиям линейной комбинации базисных вектор-функций

$$u_h(\chi) = \sum_{i=1}^{N} x_i \varphi_i(\chi), \tag{6}$$

где φ_i $(i = 1, 2, \ldots, N)$ — упомянутый выше кусочно-полиномиальный базис U_0^h. Тогда разрешающие дискретные задачи имеют вид:

- системы линейных алгебраических уравнений

$$Ax = b \tag{7}$$

для статической задачи (1);

- задачи с начальными условиями

$$Bx''(t) + Cx'(t) + Ax(t) = b(t), \;\; x(t_0) = x^{(0)}, \;\; x'(t_0) = x^{(1)} \tag{8}$$

для динамической задачи (2), (3);

- алгебраической проблемы собственных значений (АПСЗ)

$$Ax = \lambda^h Bx \tag{9}$$

для задачи о собственных колебаниях (4).

Разрешающие задачи (7)–(9) имеют такие особенности:

1) высокий порядок матриц разрешающих дискретных задач — от 100 000 до десятков миллионов;

2) элементы матриц и векторов разрешающих задач вычислены с погрешностями, которые вызваны погрешностями исходных данных, дискретизации и вычисления значений этих элементов в компьютере;

3) матрицы дискретных задач — жесткостей A, масс B, демпфирования C — являются

- симметричными;
- положительно-определенными или положительно-полуопределенными;
- разреженной структуры (ленточной, профильной, блочной и т.п.).

Матрица называется разреженной, если число ее элементов, отличных от 0, составляет kn и $k << n$, где n — порядок матрицы.

Важным преимуществом МКЭ является то, что элементы матриц и векторов правых частей задач (7)–(9) получаются суммированием соответствующих элементов матриц жесткости (масс, демпфирования) и векторов нагрузок, построенных для отдельных конечных элементов.

Такое свойство матриц и векторов дискретных задач МКЭ позволяет эффективно распараллелить процесс формирования этих матриц и векторов. Например, область может быть разбита на приблизительно равные подобласти (по числу используемых процессоров), и для конечных элементов, содержащихся в этих подобластях, независимо друг от друга формируются соответствующие матрицы и векторы с использованием обменов между процессорами, формируются глобальные матрицы и векторы соответствующей дискретной задачи (7)–(9). При этом глобальные матрицы и векторы могут быть или сформированы в файле (файлах) на внешнем носителе, или распределены между процессорами в соответствии с требованиями метода, который будет использоваться для решения дискретной задачи.

В целях минимизации количества арифметических операций при решении дискретных задач, а также необходимой оперативной и внешней памяти, учитывая разреженную структуру матриц задач (7)–(9), в большинстве случаев проводится переупорядочение неизвестных задач. В зависимости от критерия минимизации — ширины ленты матрицы, ее профиля, общего количества арифметических операций при треугольном разложении матрицы и т.д. — используются разные методы переупорядочения, например обратный алгоритм Катхилла–Макки, фактор деревьев, параллельных сечений, минимальной степени. Конкретные рекомендации выбора метода переупорядочения дать сложно, потому что эффективность того или другого алгоритма существенно зависит от исходной структуры конкретной матрицы.

Решение дискретных задач. Для решения дискретных задач (7)–(9) используются параллельные алгоритмы [5] традиционных методов решения соответствующих задач.

Решение систем линейных алгебраических уравнений. Для решения СЛАУ (7) — исходной или переупорядоченной — целесообразно использовать прямой метод Холецкого. При этом полученное треугольное разложение матрицы системы может быть сохранено в файле и использовано в последующих расчетах прочности конструкции для других нагрузок (т.е. правых частей), что существенно уменьшит общее время расчета прочности конструкции.

Решение задачи с начальными условиями. В полудискретном МКЭ приближенное решение ищется в виде (6), где коэффициенты x_i являются функциями времени *t*. В результате получаем систему обыкновенных дифференциальных уравнений второго порядка с начальными условиями (8), где $x(t)$, $x^{(0)}$, $x^{(1)}$ — векторы с элементами $x_i(t)$, $x_i^{(0)} = L_i(u^{(0)})$, $x_i^{(1)} = L_i(u^{(1)})$.

Для решения задачи с начальными условиями (8) используется метод разложения по формам собственных колебаний, т.е. по собственным векторам соответствующей задачи (9). Приближенное решение представляется в виде линейной комбинации нескольких собственных векторов, соответствующих наименьшим собственным значениям этой задачи. Если λ_k, z_k ($z_j^{\mathrm{T}} B z_k = \delta_{jk}$, $j, k = 1, 2, \ldots, n$) — решение алгебраической проблемы собственных значений (9), то, считая в (8) $x(t) = \sum_{k=1}^{n} y_k(t)\, z_k$, получим (при некоторых предположениях относительно матрицы демпфирования *C*) систему, которая распадается на независимые относительно $y_i(t)$ уравнения:

$$\begin{gathered} y_k''(t) + 2\xi_k \omega_k y_k'(t) + \omega_k^2 y_k(t) = P_k(t),\ t > t_0, \\ y_k(t_0) = y_k^{(0)}, \quad y_k'(t_0) = y_k^{(1)}, \end{gathered} \tag{10}$$

где $0 < \xi_k < 1$, $\omega_k^2 = \lambda_k$, $P_k(t) = z_k^{\mathrm{T}} b(t)$, $y_k^{(0)} = (x^{(0)})^{\mathrm{T}} B z_k$, $y_k^{(1)} = (x^{(1)})^{\mathrm{T}} B z_k$, $k = 1, 2, \ldots, n$. Анализ решений системы (10) свидетельствует о том, что существенный вклад в решение $x(t)$ задачи (8) дают лишь 10–20 слагаемых, соответствующих минимальным собственным значениям.

Решение алгебраической проблемы собственных значений. Для решения частичной обобщенной АПСЗ (9) используется метод итераций на подпространстве [6], позволяющий вычислить заданное количество минимальных собственных значений задачи и соответствующих им собственных векторов. Этот метод

заключается в построении последовательности подпространств E_t $(t = 1, 2, \ldots)$, сходящейся к подпространству E_∞, которое содержит искомые собственные векторы. Предполагается, что матрицы задачи (9) симметричные и одна из них (обычно матрица A) положительно-определенная.

На t-й итерации необходимо выполнить следующие операции: (i) вычисление решения СЛАУ $AX_t = Y_{t-1}$; (ii) вычисление проекции $A_t = X_t^{\mathrm{T}} Y_{t-1}$ матрицы A на подпространство E_t; (iii) вычисление прямоугольной матрицы $W_t = BX_t$; (iv) вычисление проекции $B_t = X_t^{\mathrm{T}} W_t$ матрицы B на подпространство E_t; (v) решение полной АПСЗ для проекций $A_t Z_t = B_t Z_t \Lambda_t$; (vi) вычисление очередного приближения $Y_t = W_t Z_t$; (vii) проверка условий окончания итерационного процесса. Если эти условия выполняются после t итераций, то приближенным решением задачи (9) считаем $\lambda_i^* = \lambda_i^{(t+1)}$ $(i = 1, 2, \ldots, r,$ в предположении, что собственные значения упорядочены по возрастанию), $X^* = X_{t+1} Z_{t+1}$. Матричная система $AX_t = Y_{t-1}$ решается методом Холецкого. Причем разложение матрицы выполняется один раз до начала итерационного процесса, а на каждой итерации решаются две системы с нижней и верхней треугольными матрицами.

Для сходимости итерационного процесса важно, чтобы матрицы проекций A_t и B_t были положительно определены. Однако при математическом моделировании прочности конструкций матрица массы может быть положительно полуопределена. Поэтому необходимо так выбирать начальное подпространство E_0, чтобы матрица B_1 была положительно определена.

Решение нелинейных задач выполняется итерационно — на каждой итерации формируется новая СЛАУ с разреженной симметричной матрицей жесткости. Эта система решается методом Холецкого. После этого проверяются условия окончания итерационного процесса, и, если заданная точность не достигнута, на основе полученного приближения к решению модифицируются данные для формирования новой системы.

Таким образом, решение нелинейных задач, а также задач (8) и (9) базируется на решении СЛАУ вида (7) с разреженной симметричной матрицей, основой решения которой служит разложение этой матрицы в произведение нижней и верхней треугольных матриц. Следовательно, основные параметры (схемы распределения между процессорными устройствами, хранения и обработки данных) и эффективность параллельных алгоритмов решения задач (7)–(9) в первую очередь определяются качеством параллельных алгоритмов разложения методом Холецкого разреженных симметричных матриц (жесткости) различных структур.

В зависимости от структуры и заполненности матрицы, а также архитектуры компьютера используются

— блочные параллельные алгоритмы для исследования и решения СЛАУ с узкими ленточными матрицами и блочно-диагональными матрицами с окаймлением [5];

— одномерные блочные циклические параллельные алгоритмы для исследования и решения СЛАУ с ленточными, профильными или «небоскребными» (у которых не принимают участие в вычислениях внутренние элементы профиля матрицы, остающиеся нулевыми в процессе разложения матрицы) матрицами [5, 7–9].

Далее подробно рассмотрим блочные циклические алгоритмы решения на компьютерах гибридной архитектуры СЛАУ (7) с ленточной симметричной матрицей.

2. Гибридный алгоритм решения СЛАУ с симметричной положительно-определенной ленточной матрицей

Рассмотрим решение СЛАУ (в общем случае матричной) вида (7)

$$AX = B \tag{11}$$

порядка n с ленточной (ширина ленты $2m+1$) симметричной положительно-определенной матрицей и q правыми частями.

Как известно, наиболее эффективным прямым методом решения задачи (11) является метод Холецкого. В этом случае задача (11) разбивается на три подзадачи:

- факторизация (LL^T- или LDL^T -разложение) исходной матрицы системы

$$A = LL^T \text{ или } A = LDL^T; \tag{12}$$

- решение СЛАУ с нижней треугольной матрицей

$$LY = B; \tag{13}$$

- решение СЛАУ с верхней треугольной матрицей

$$L^T X = Y \text{ или } DL^T X = Y. \tag{14}$$

Схемы распределения и хранения данных. Проблему эффективного распараллеливания алгоритмов, в основе которых лежит факторизация исходной матрицы, удалось решить с использованием циклических схем распределения и хранения элементов матриц и векторов. Эти схемы были предложены независимо друг от друга и практически одновременно несколькими группами исследователей [10, 11]. При использовании строчно-циклической схемы элементы матрицы распределяются циклически между p процессорами (потоками, процессами), участвующими в вычислениях: если в $(i-1)$-м процессоре располагаются элементы r-й строки матрицы, то в следующем ($i-$м) — элементы $(r+1)$-й строки.

Однако более эффективными являются блочно-циклические схемы распределения и хранения элементов матриц [5, 7, 8]. Например, при использовании одномерной блочно-циклической схемы каждому CPU распределяются не отдельные строки, а полосы из s подряд идущих строк, а операции с отдельными элементами матрицы заменяются операциями с квадратными блоками порядка s или с прямоугольными блоками из s строк. С одной стороны, такой подход позволяет уменьшить коммуникационные потери, а с другой, — обеспечивает значительно больший объем однородных вычислений, что приобретает особое значение при использовании GPU.

Пусть гибридный алгоритм решения задачи (11) реализуется на p графических процессорах и таком же количестве CPU, виртуальные связи между которыми обеспечивают возможность передачи данных из одного вычислительного устройства в любое другое. Вычислительные устройства имеют логические номера $0, 1, \ldots, p-1$. При этом GPU с логическим номером i является сопроцессором-ускорителем CPU с таким же логическим номером.

Для простоты изложения предположим, что порядок ленточной симметричной матрицы системы и полуширина ее ленты кратны порядку матричных блоков s, т.е. $n = Ns$ и $m = Ms$. Параметр s должен выбираться таким образом, чтобы блок полностью помещался в кеш-памяти как CPU, так и GPU. Таким образом,

ленточная симметричная матрица разбивается на N^2 квадратных блоков размера $s \times s$, причем каждая строка содержит не более $2M+1$ ненулевых блоков. Поскольку матрица системы симметричная, для решения достаточно хранить только ее диагональные элементы и элементы ее нижнего или верхнего треугольника (в зависимости от варианта алгоритма), находящиеся в ненулевых блоках. Аналогичное представление используется и для нижней треугольной ленточной матрицы разложения L (рис. 1) или верхней треугольной матрицы L^{T}.

$$\begin{pmatrix} L_{1,1} & 0 & 0 \cdots & 0 & 0 \\ L_{2,1} & L_{2,2} & 0 \cdots & 0 & 0 \\ L_{3,1} & L_{3,2} & \ddots & & \\ \vdots & \vdots & & \vdots & \vdots \\ L_{M+1,1} & L_{M+1,2} & & & \\ 0 & L_{M+2,2} & & & \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \cdots & L_{N-1,N-1} & 0 \\ 0 & 0 & \cdots & L_{N,N-1} & L_{N,N} \end{pmatrix}$$

Рис. 1

Элементы исходной матрицы распределяются между CPU согласно следующей схеме: блок $A_{I,J}$ хранится в CPU с логическим номером $(I-1) \bmod p$. Таким же образом распределяются блоки матрицы L. Прямоугольные матрицы B и X делятся на прямоугольные блоки размера $s \times q$. Эти блоки распределяются между CPU согласно той же блочно-циклической схеме, что и блоки матриц A и L.

Гибридный алгоритм LL^{T}-разложения ленточной симметричной матрицы. Блочный алгоритм факторизации состоит из N шагов. Будем называть CPU и GPU ведущими на данном шаге, если блок $L_{I,I}$ или $A_{I,I}$ хранится в их памяти. Для обозначения блоков матриц A и L, находящихся в памяти GPU, используется верхний индекс d.

На I-м шаге вычисляются ненулевые блоки I-го столбца матрицы L и модифицируется квадратная подматрица порядка Ms или меньше, в верхнем левом углу которой расположен блок $A_{I+1,I+1}$. Таким образом, на I-м шаге алгоритма факторизации обрабатываются только те блоки, которые находятся в I-м столбце блоков и в M столбцах блоков, следующих за ним [6]. Из разбиения матрицы (рис. 1) видно, что в каждом таком столбце имеется не больше $M+1$ ненулевых блоков. Поэтому с учетом симметрии на каждом шаге алгоритма достаточно хранить в памяти GPU не более $(M+1)(M+2)/2$ блоков. Предполагается, что блоки подматрицы матрицы A порядка $(M+1)s$, расположенной в верхнем левом углу матрицы, скопированы в память соответствующих GPU согласно правилу: блок $A_{I,j}$ располагается в памяти GPU с номером $(2I-J-1) \bmod p$.

Таким образом, алгоритм LL^{T}-разложения ленточной симметричной матрицы состоит из следующих действий (решений подзадач) для каждого $I = 1, 2, \ldots, N$.

(i) LL^{T}-разложение блока $A_{I,I}$ на ведущем CPU с использованием последовательного алгоритма

$$A_{I,I} = L_{I,I} L_{I,I}^{\mathrm{T}}. \tag{15}$$

(ii) Рассылка блока $L_{I,I}$ из ведущего CPU всем GPU, участвующим в вычислениях.

(iii) Вычисление соответствующим GPU ненулевых блоков I-го столбца $L^d_{J,I}$, где $J = I+1,\ldots,\min\{I+M, N\}$, путем решения соответствующих матричных СЛАУ

$$L_{I,I}L^{\mathrm{T}}_{J,I} = A^{\mathrm{T}}_{J,I}. \tag{16}$$

(iv) Асинхронная $2s$-ранговая модификация на GPU при условии $(K-J) \bmod p = 0$

$$A^{(I)}_{J,K} = A^{(I-1)}_{J,K} - L_{J,I}L^{\mathrm{T}}_{K,I}, \tag{17}$$

где $J = I+1,\ldots,\min\{I+M, N\}$, $K = J,\ldots,\min\{I+M, N\}$.

(v) Рассылка каждым CPU вычисленных при выполнении операции (iii) блоков $L^d_{J,I}$ всем остальным вычислительным устройствам.

(vi) $2s$-ранговая модификация на соответствующем GPU (при условии $(K-J) \bmod p > 0$) по формуле (17) блоков $A^d_{J,K}$, где $J = I+1,\ldots,\min\{I+M, N\}$, $K = J,\ldots,\min\{I+M, N\}$.

(vii) Асинхронное копирование всех блоков матрицы, которые находятся в столбце $I+M+2$, в память соответствующих GPU, если $I+M+2 < p$.

(viii) Синхронизация вычисления блока $A^d_{I+1,I+1}$ на CPU с логическим номером $I \bmod p$.

(ix) Копирование блока $A^d_{I+1,I+1}$ в память соответствующего CPU.

Заметим, что условие $(K-J) \bmod p = 0$ при выполнении операции (iv) алгоритма означает, что все блоки, необходимые для вычисления по формуле (16), находятся в одном и том же GPU. А при выполнении операции (vi) алгоритма модифицируются только те блоки, у которых разность индексов не делится на p. Это означает, что необходимые блоки были получены при рассылке (v) от других устройств.

Следует также отметить, что аналогичный алгоритм может быть использован для LDL^{T}-разложения ленточной симметричной матрицы. В этом случае вычисления проводятся по формулам $A_{I,I} = G_{I,I}L^{\mathrm{T}}_{I,I}$, $L_{I,I}G^{\mathrm{T}}_{J,I} = A^{\mathrm{T}}_{J,I}$ и $A^{(I)}_{J,K} = A^{(I-1)}_{J,K} - G_{J,I}L^{\mathrm{T}}_{K,I}$, где $G_{J,I} = L_{J,I}D_I$, позволяющим не увеличивать количество арифметических операций, но требующим дополнительный объем памяти для временного хранения блоков $G_{J,I}$.

Решение СЛАУ с нижней треугольной ленточной матрицей. Вычислительная схема алгоритма решения задачи (13) $LY = B$ во многом аналогична вычислительной схеме алгоритма факторизации. Этот алгоритм также состоит из N шагов. На каждом шаге $I = 1, 2,\ldots, N$ выполняются следующие действия (решаются подзадачи), в результате которых вычисляется один блок решения.

(i) Решение СЛАУ $L_{I,I}Y_I = B^{(I-1)}_I$ с нижней треугольной матрицей $L_{I,I}$ на ведущем GPU.

(ii) Рассылка из ведущего GPU вычисленного блока решения Y_I всем остальным GPU.

(iii) Модификация (s-ранговая) блоков правой части B_J ($J = I+1,\ldots$ $\ldots, \min\{I+M, N\}$) по формуле $B_J^{(I)} = B_J^{(I-1)} - L_{J,I}Y_I$ на GPU в соответствии с их распределением.

Решение СЛАУ с верхней треугольной ленточной матрицей. Вычислительная схема алгоритма решения задачи (14) $L^{\mathrm{T}}X = Y$ существенно отличается от предыдущей, поскольку распределенными между процессорными устройствами являются столбцы блоков матрицы системы. Этот алгоритм также состоит из N шагов. На I-шаге ($I = 1, 2, \ldots, N$) выполняются следующие действия (решаются подзадачи), в результате которых вычисляется один блок решения системы (здесь $K = N - I + 1$).

(i) Вычисление при условиях $I > 1$ и $(K-J) \bmod p = 0$ каждым GPU матричных произведений $L_{J,K}^{\mathrm{T}} X_J$ ($J = K+1, \ldots, \min\{K+M, N\}$) в соответствии с распределением блоков X_J.

(ii) Вычисление (при $I > p$) каждым GPU сумм вычисленных ранее матричных произведений.

(iii) Мультисборка ведущим CPU (при $I > p$) сумм матричных произведений и вычисление $Y_K^{(I-1)} = Y_K - \sum_{J=1}^{M_K} L_{K+J,K}^{\mathrm{T}} X_{K+J}$, $M_K = \min\{M, M-K\}$.

(iv) Решение СЛАУ $L_{K,K}^{\mathrm{T}} X_K = Y_K^{(I-1)}$ с верхней треугольной матрицей $L_{K,K}^{\mathrm{T}}$ на ведущем GPU ($(Y_I^{(0)} \equiv Y_I$).

Эффективность алгоритма факторизации. Количество арифметических операций, выполняемых при факторизации (12) матрицы системы (11), в m/q раз больше количества арифметических операций, выполняемых при решении треугольной системы (13) или (14), поэтому эффективность рассматриваемого гибридного алгоритма решения СЛАУ с ленточной симметричной матрицей определяется эффективностью алгоритма разложения матрицы системы.

Обозначим время выполнения одной арифметической операции на CPU как t_C, а на GPU — t_G. Количество арифметических операций, которое GPU может выполнять одновременно, обозначим n_o. Поскольку GPU реализует архитектуру SIMD, среднее время, затраченное одним GPU на выполнение Q однородных арифметических операций, можно оценить величиной Qt_G/n_o.

Количество арифметических операций, выполняемых при решении k-й подзадачи алгоритма, обозначим O_k. Тогда справедливы следующие оценки: $O_1 \approx s^3/3$, $O_3 \approx ms^2$, $O_4 \approx (m+sp)ms/p$, $O_7 \approx m^2 s(p-1)/p$. Остальные подзадачи связаны с обменами данными между процессорными устройствами. Время пересылки одного блока между двумя GPU обозначим t_B. Если для рассылки данных от одного GPU всем остальным используется алгоритм «дерева» [12], то общее время мультирассылки можно оценить величиной $t_B \log_2 p$. Обозначим суммарное время копирования (ii) и (ix) блоков между CPU и GPU как t_I. Время на выполнение копирования (vii) можно не учитывать, так как это копирование проводится на фоне вычислений.

Таким образом время выполнения алгоритма на 1 CPU и 1 GPU оценивается величиной

$$T_1 = N(O_1 t_C + t_I + ms(m+2s)t_G/n_o), \qquad (18)$$

а время выполнения алгоритма на p CPU и p GPU величиной

$$T_p = \frac{N}{p}(p(O_1 t_C + t_I) + ms(m + s - m/p)t_G/n_o + \\ + \max\{Mt_B p\log_2 p;\ O_4 t_G/n_o\}). \qquad (19)$$

В большинстве практических задач временем обмена t_I и факторизации диагонального блока $O_1 t_C$ можно пренебречь, так как эти времена, как правило, намного меньше (при естественном условии $M > p$) остальных слагаемых в формулах (18) и (19). При таком предположении оценки ускорения и эффективности рассматриваемого гибридного алгоритма имеют следующий вид ($C_B = Mp\log_2 p$):

$$S_p = \frac{T_1}{T_p} \cong \begin{cases} p\left(1 - \dfrac{1 + M/p}{M+2} + \dfrac{p\log_2 p}{s^3(M+2)}\dfrac{n_o t_B}{t_G}\right)^{-1}, & O_4 t_G/n_o < C_B t_B; \\ p, & O_4 t_G/n_o \ge C_B t_B; \end{cases} \qquad (20)$$

$$E_p = \frac{S_p}{p}.$$

Оценки (20) свидетельствуют, что эффективность алгоритма не зависит от порядка матрицы, а при определенных условиях определяется количеством ненулевых блоков в строке блоков и их размером. На эффективность алгоритма также влияет отношение времени пересылки одного блока между двумя GPU t_B и значения t_G/n_o. Также следует отметить, что хотя время факторизации диагональных блоков существенно не влияет на эффективность распараллеливания, его сокращение позволяет уменьшить общее время разложения матрицы системы (учитывая, что эти факторизации выполняются в последовательном режиме). Например, можно перенести эти вычисления на GPU.

Экспериментальное исследование гибридных алгоритмов. Описанный гибридный алгоритм решения СЛАУ с ленточной симметричной матрицей реализован в программной среде MPI на компьютерах с многоядерными и графическими процессорами — кластерах семейства СКИТ Института кибернетики им. В.М. Глушкова НАН Украины [13] и семейства Инпарком (совместная разработка Института кибернетики и ГНПП «Электронмаш» [14]). Для организации вычислений на GPU использована технология CUDA. При реализации гибридного алгоритма для матрично-векторных и матрично-матричных операций использованы функции библиотек Intel MKL (для вычислений на CPU) и CUBLAS (для GPU).

Для лучшего использования вычислительных ресурсов гибридного компьютера необходимо асинхронное выполнение некоторых операций, при этом для вычислений на GPU применяются дополнительные потоки CUDA. Это позволяет выполнять одновременно операцию синхронизации (viii), разложение на CPU диагонального блока (i) и 2s-ранговую модификацию по формуле (17) на GPU (iv).

Исследования производительности предложенного гибридного алгоритма LL^{T}-разложения ленточной симметричной матрицы в зависимости от величины выбранного блока и количества GPU проводились для двух матриц СЛАУ со следующими параметрами: первая — порядок $n = 1151112$, полуширина ленты $m = 5357$, вторая — $n = 1438890$, $m = 6693$. Результаты (производительность в Gflops — 10^9 арифметических операций с плавающей запятой в секунду) пред-

ставлены в виде диаграмм (А — для первой матрицы, Б — второй) на рис. 2. Эти исследования позволяют экспериментально определить оптимальный размер блока. Как видно из приведенных диаграмм, для матриц с указанными выше параметрами оптимальным является размер блока $s = 192$, так как увеличение размера дает или незначительный рост, или уменьшение производительности, в то же время увеличение размера блока с 96 до 192 позволило увеличить производительность в 1,5–2 раза.

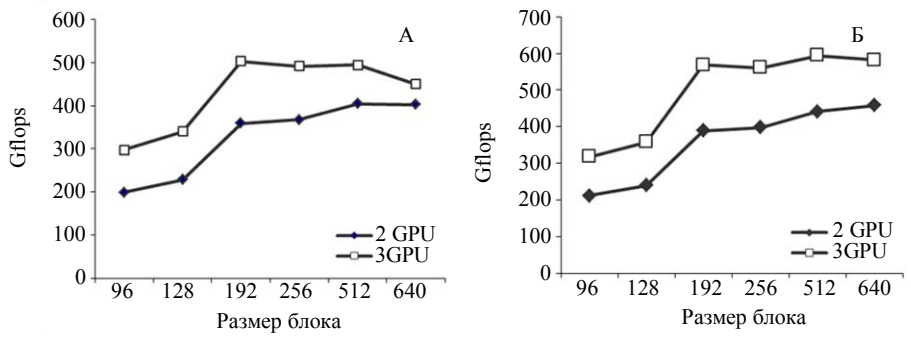

Рис. 2

Также проведено сравнение предложенного гибридного алгоритма и параллельных алгоритмов для многоядерных компьютеров, в частности алгоритма, программно реализованного в библиотеке Intel MKL. На рис. 3 представлены результаты этого сравнения — ускорение гибридного алгоритма по отношению к используемому библиотекой Intel MKL в зависимости от количества используемых ядер CPU. Результаты получены при решении СЛАУ порядка 1151112 (полуширина ленты матрицы — 5357) и свидетельствуют о высокой эффективности предложенного алгоритма.

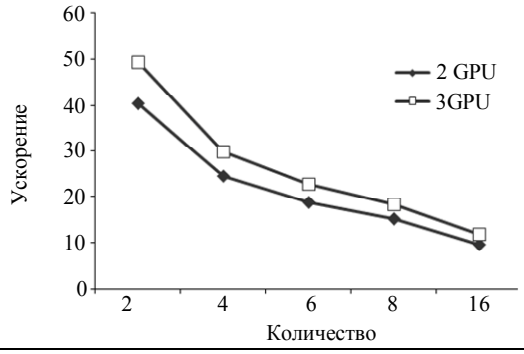

Рис. 3

В заключение этого раздела следует отметить, что общая схема предложенных алгоритмов (разбиение на блоки, распределение между процессорными устройствами, обработка только блоков, соответствующих ненулевым блокам матрицы разложения) может быть успешно применена и для решения СЛАУ с другими структурами разреженных матриц — профильной, блочно-диагональной с окаймлением и т.п.

3. Численное моделирование

Для исследования возможностей предложенного параллельного алгоритма решения СЛАУ решен ряд сложных практических задач проектирования — численные исследования ряда объектов, в том числе 27-этажного здания (рис. 4).

В таблице приведены результаты исследований ускорений, получаемых при решении на компьютере гибридной архитектуры СЛАУ и возникающих при ста-

тическом анализе прочности различных строительных объектов и отдельных конструкций. При дискретизации элементов конструкции использованы трехмерные конечные элементы (3D КЭ).

Таблица

Объект	Порядок матрицы жесткости	Полуширина ленты матрицы	Время решения СЛАУ	
			Лира–Сапр* (мин.)	Инпарком_G** (мин.)
Здание, 12 эт.	189956	22 585	1	0,274
Фундамент	283031	19 530	2	0,384
Балка, 3D КЭ	300000	335	1	0,048
Здание, 27 эт.	661590	34 242	2	0,948
Плита, 3D КЭ	666000	1 004	1	0,138
Плита, 3D КЭ	1000332	1 004	2	0,207
Балка, 3D КЭ	1200000	1 265	16	0,371

Примечание:
*Лира–Сапр [2] — многопоточная версия 2015 г., GPU не использовался.
**Инпарком_G [14] — использован 1 GPU, гибридный алгоритм.

На рис. 4 представлена 3D-модель МКЭ 27-этажного здания. Рассмотрим общую характеристику и входные данные численной модели пространственных несущих конструкций этого здания:

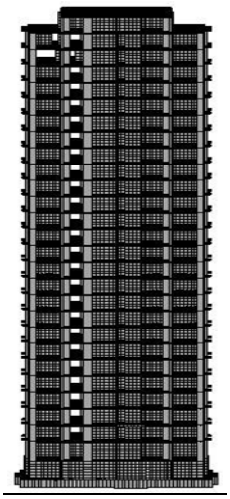

Рис. 4

• сваи основания моделируются одноузловыми конечными элементами (КЭ): $Rz = 50990$ т/м, $Rx = 50990$ т/м, $Ry = 50990$ т/м;

• фундаментная плита на отметке $h = -4$ м моделируется КЭ оболочки; модуль деформации бетона $E = 3{,}059 \times \times 10^6$ т/м^2; коэффициент Пуассона $\nu = 0{,}25$; толщина плиты $H = 150$ см;

• колонны моделируются стержневыми КЭ; $E = 3{,}059 \times \times 10^6$ т/м^2; $\nu = 0{,}25$; ширина и высота сечения колонн ($b \times h$) 25×90 см, 25×120 см.

• перекрытия моделируются КЭ оболочки; $E = 3{,}059 \times \times 10^6$ т/м^2; $\nu = 0{,}25$; толщина плит перекрытия $H = 18$ см.

Рассмотрено шесть вариантов нагружений:

(1) собственный вес 27-этажного здания;

(2) полы, перегородки, кровля;

(3) полезные, снег;

(4) трапециевидная нагрузка от грунта на фундаментно-подвальную часть здания;

(5) ветер по направлению X;

(6) ветер по направлению Y.

Таким образом, разрешающая СЛАУ (11) имеет шесть правых частей, т.е. $q = 6$.

Некоторые результаты численного анализа прочности высотного здания представлены детальными картинами распределения перемещений по Y, Z (рис. 5) и усилий N (рис. 6) для рассматриваемых вариантов нагружений.

Полученные результаты численных исследований поведения строительных конструкций при статических нагружениях показали многократное сокращение времени решения СЛАУ с ленточными симметричными положительно-определенными матрицами на многопроцессорных (многоядерных) компьютерах с графическими ускорителями при использовании предложенных гибридных алгоритмов.

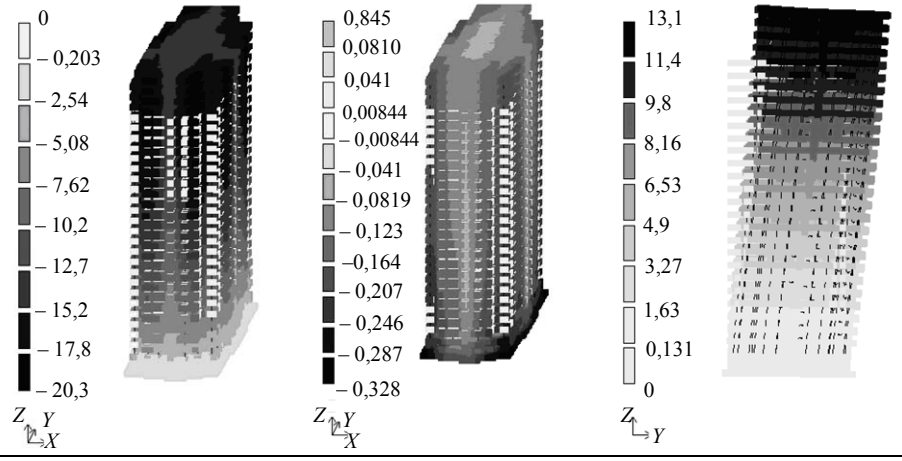

Рис. 5

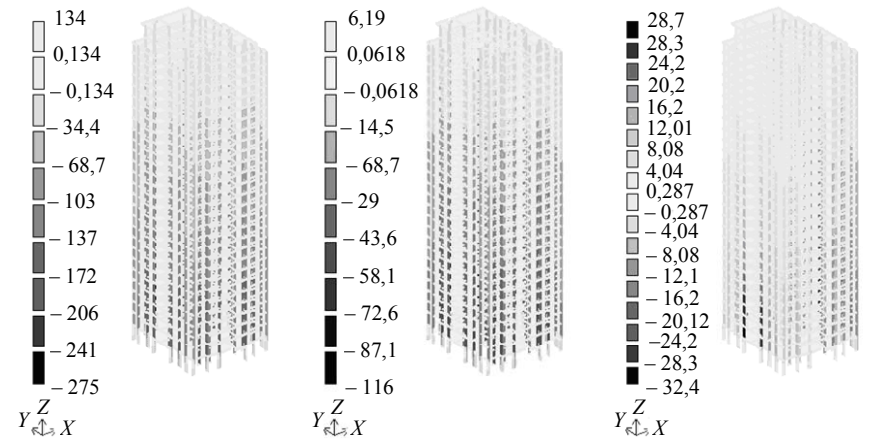

Рис. 6

Заключение

Общая схема параллельных алгоритмов (15)–(17), а также ее реализация (разбиение на блоки, распределение между процессорными устройствами, обработка блоков, соответствующих ненулевым блокам матрицы разложения) могут быть успешно применены и для решения СЛАУ с другими структурами разреженных матриц — профильной, блочно-диагональной с окаймлением и т.п. (см., например, [15]).

Оптимизация вычислений для компьютеров гибридной архитектуры лежит, очевидно, в такой модификации алгоритмов из [8], которая позволяет более полно учитывать разреженную структуру матрицы жесткости и ее LL^{T}-разложения.

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

Особенно существенный эффект ускорения можно ожидать при решении практических задач проектирования ответственных особо сложных пространственных конструкций при статических и динамических нагружениях, в том числе при исследовании жизненного цикла высотных сооружений с учетом нелинейных свойств материалов конструкций и грунтов оснований с использованием вышеизложенных прямых методов решения СЛАУ (с количеством неизвестных 10^6–10^9).

Использование приведенных высокопроизводительных параллельных алгоритмов позволяет обеспечить высокое качество проектирования и безопасность возведения, эксплуатации и реконструкции особо сложных строительных объектов.

А.Ю. Баранов, О.В. Попов, Я.О. Слободян, О.М. Хіміч

МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ МІЦНОСТІ БУДІВЕЛЬНИХ КОНСТРУКЦІЙ НА ГІБРИДНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ

Розглядаються алгоритми розв'язування на комп'ютерах гібридної архітектури (на багатоядерних процесорах і графічних прискорювачах) систем лінійних алгебраїчних рівнянь зі стрічковими симетричними додатно-визначеними матрицями, які виникають при чисельному моделюванні з використанням методу скінченних елементів, просторових будівельних конструкцій. Наведено результати апробації запропонованих гібридних алгоритмів на розв'язуванні задач статичного аналізу міцності декількох будівельних об'єктів і окремих конструкцій.

A.Yu. Baranov, A.V. Popov, Ya.E. Slobodyan, A.N. Khimich

MATHEMATICAL MODELING OF STRENGTH OF BUILDING STRUCTURES FOR HYBRID COMPUTING SYSTEMS

Algorithms for solving on computers of hybrid architecture (with multi-core processors and graphics accelerators) the systems of linear algebraic equations with band symmetrical positive definite matrices, arising in the numerical modeling of 3D building constructions using the finite element method is considered. Some results of the proposed hybrid algorithms testing for solving the problems of static strength analysis of construction sites and individual constructions are presented.

1. *https*://www.top500.org/lists/2016/11/
2. *http*://www.liraland.ua/company/person.php
3. *Тимошенко С.П., Гудьер Дж.* Теория упругости. — М. : Наука, 1975. — 575 с.
4. *Стренг Г., Фикс Дж.* Теория метода конечных элементов. — М. : Мир, 1973. — 349 с.
5. *Параллельные* алгоритмы решения задач вычислительной математики / А.Н. Химич, И.Н. Молчанов, А.В. Попов, Т.В. Чистякова, М.Ф. Яковлев. — Киев : Наук. думка, 2008. — 248 с.
6. *Парлетт Б.* Симметричная проблема собственных значений. — М. : Мир, 1983. — 318 с.
7. *Попов А.В., Химич А.Н.* Параллельный алгоритм решения систем линейных алгебраических уравнений с ленточной симметричной матрицей // Компьютерная математика. — 2005. — № 2. — С. 52–59.
8. *Химич А.Н., Попов А.В., Полянко В.В.* Алгоритмы параллельных вычислений для задач линейной алгебры с матрицами нерегулярной структуры // Кибернетика и системный анализ. — 2011. — № 6. — С. 159 –174.
9. *Хіміч О.М., Баранов А.Ю.* Гібридний алгоритм розв'язування лінійних систем із стрічковими матрицями прямими методами // Компьютерная математика. — 2013. — № 2. — С. 80–87.
10. *Численные* методы для многопроцессорного вычислительного комплекса ЕС / Михалевич В.С., Молчанов И.Н., Сергиенко И.В. и др. / Под ред. И.Н. Молчанова. — М. : ВВИА им. проф. Н.Е. Жуковского, 1986. — 401 с.
11. *Ортега Дж.* Введение в параллельные и векторные методы решения линейных систем. — М. : Мир, 1991. — 368 с.
12. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. — СПб. : БХВ-Петербург, 2002. — 608 с.
13. *http*://icybcluster.org.ua
14. *http*://www.inparcom.com
15. *Хіміч О.М., Сидорук В.А.* Плитковий гібридний алгоритм факторизації розріджених блочно-діагональних матриць з обрамленням // Компьютерная математика. — 2016. — № 1. — С. 72−79.

Получено 29.12.2016