УДК 519.63, 004.272.2

© *А. К. Новиков, И. М. Кузьмин, Н. С. Недожогин*

НЕКОТОРЫЕ ОСОБЕННОСТИ РЕАЛИЗАЦИИ КОНЕЧНО-ЭЛЕМЕНТНЫХ ТЕХНОЛОГИЙ ДЛЯ ГИБРИДНОЙ ВЫЧИСЛИТЕЛЬНОЙ АРХИТЕКТУРЫ[1]

Рассматриваются особенности параллельных конечно-элементных вычислений на гибридных архитектурах, связанные с заданием граничных условий первого рода (Дирихле). Сравниваются варианты задания граничных условий Дирихле при формировании и решении конечно-элементной системы уравнений, обеспечивающие сохранение симметрии матрицы коэффициентов.

Ключевые слова: параллельные вычисления, метод конечных элементов, граничные условия Дирихле, гибридные вычислительные архитектуры.

Введение

При конечно-элементном моделировании на кластерах гибридной (CPU+GPU) архитектуры вычисления тем или иным образом распределяются между центральными и графическими процессорами. Соответственно, конечно-элементные данные, в том числе матрицы и вектора, располагаются на различных уровнях подсистемы памяти кластера: кэш-память, оперативная память, память ускорителя, подсистемы памяти других вычислительных узлов. Подобное размещение данных вносит особенности в алгоритмы, применяемые при реализации вычислительных схем метода конечных элементов (МКЭ) на гибридных архитектурах.

В методе конечных элементов применяется ряд алгоритмов, обеспечивающих сохранение симметрии матрицы коэффициентов системы сеточных уравнений при задании граничных условий первого рода или Дирихле. Такие алгоритмы применяются как при формировании конечно-элементных систем [1–3], так и при их решении [4].

Отметим, что изменения, связанные с граничными условиями Дирихле, возможно внести на нескольких уровнях: в собранной системе линейных алгебраических уравнений [1]; в локальных (элементных) матрицах жесткости и векторах [2], а также без изменения матрицы коэффициентов и вектора правой части [3].

Задание условий первого рода в параллельных конечно-элементных вычислениях на гибридной архитектуре требует модификации существующих алгоритмов. Далее рассматриваются особенности реализации граничных условий Дирихле на этапах формирования и решения конечно-элементных систем при параллельных вычислениях на гибридной архитектуре.

§ 1. Условия Дирихле на этапе формировании конечно-элементной СЛАУ

Рассмотрим, каким образом на гибридных архитектурах задаются граничные условия первого рода в случае явной сборки конечно-элементной системы уравнений. До задания граничных условий Дирихле имеем систему (1.1) с вырожденной матрицей коэффициентов $\widetilde{K}$:

$$\widetilde{K}\widetilde{u} = \widetilde{f}; \quad \widetilde{K} = \widetilde{K}^T; \ (\widetilde{K}x, x) = 0 \,\forall x \neq 0; \ \widetilde{K} \in \mathbb{R}^{\widetilde{N}\times\widetilde{N}}, \ \widetilde{u} \in \mathbb{R}^{\widetilde{N}}, \ \widetilde{f} \in \mathbb{R}^{\widetilde{N}}. \tag{1.1}$$

Процесс сборки системы (1.1) запишем в следующем виде:

$$\widetilde{K} = \sum_{e=1}^{m} \widetilde{C}_e^T \widetilde{K}_e \widetilde{C}_e, \quad \widetilde{f} = \sum_{e=1}^{m} \widetilde{C}_e^T \widetilde{f}_e, \tag{1.2}$$

здесь $\widetilde{C}_e \in \mathbb{Z}^{N_e\times\widetilde{N}}$ — матрица связности, $\widetilde{K}_e \in \mathbb{R}^{\widetilde{N}_e\times N_e}$ — локальная матрица жесткости, $\widetilde{f}_e \in \mathbb{R}^{N_e}$ — локальный вектор, $\widetilde{N}_e$ — число степеней свободы в конечном элементе e; m — число конечных элементов, $\widetilde{N}$ — число всех степеней свободы до задания граничных условий первого рода.

[1] Работа поддержана РФФИ (гранты №№ 13–01–00101-а, 14–01–00055-а).

На гибридной архитектуре процесс сборки системы уравнений (1.1) возможно осуществить, используя различные вычислительные ресурсы: центральные (CPU), графические (GPU) или одновременно центральные и графические процессоры (CPU+GPU). При выполнении сборки (1.2) на центральном процессоре матрицы $\widetilde{K}_e$, ранее вычисленные на GPU, необходимо переслать в оперативную память. После задания граничных условий первого рода получаем

$$Ku = f; \quad K = K^T; \ (Kx, x) > 0 \, \forall x \neq 0; \ K \in \mathbb{R}^{N \times N}, \ u \in \mathbb{R}^N, \ f \in \mathbb{R}^N, \tag{1.3}$$

где N — число неизвестных степеней свободы. Для параллельного решения полученной системы уравнений (1.3) ее необходимо разделить на блоки и разослать по вычислительным ресурсам кластера.

В случае сборки (1.2), выполняемой на нескольких GPU или на CPU+GPU, система уравнений (1.1) сразу формируется в распределенном виде. При этом в соответствующих областях памяти вычислительной системы формируются блоки строк матрицы K и блоки элементов вектора f. Тогда не требуется разделение и последующая рассылка системы (1.3), но для этого необходимо изменить существующие алгоритмы задания условий Дирихле.

В качестве более общего случая будем рассматривать алгоритмы [1] задания граничных условий первого рода на примере задания неоднородного условия Дирихле $u_i = \mathring{u}_i \neq 0$, где $i \in [1, \widetilde{N}]$ — номер неизвестного (степени свободы) в случае распределенной системы (1.1).

В первом алгоритме, использующем умножение на большое число α, полагается $k_{ii} \leftarrow \alpha k_{ii}$ и $f_i = \alpha k_{ii} \mathring{u}_i$, как правило, принимается $\alpha \geqslant 10^8$. В случае распределенной системы (1.1) алгоритм задания граничных условий выполняется на том вычислительном ресурсе, в области памяти которого оказался элемент $\widetilde{k}_{ii}$. Следует учесть, что в результате решения (1.3) при использовании первого алгоритма получаем $u_i \approx \mathring{u}_i$, так как в строке i матрицы K остаются недиагональные ненулевые элементы. Кроме того, выбор α ограничен сверху обусловленностью системы уравнений (1.3).

Рассмотрим второй алгоритм, в котором над элементами матрицы $\widetilde{K}$ и вектора $\widetilde{f}$ выполняются следующие операции: $\widetilde{k}_{ii} = 1$, $\widetilde{f}_i = \mathring{u}_i$, $\widetilde{f}_j \leftarrow \widetilde{f}_j - \widetilde{k}_{ji} \mathring{u}_i$, $\widetilde{k}_{ij} = 0$, $\widetilde{k}_{ji} = 0$, $j = \overline{1, N}$, здесь $N = \widetilde{N}$. Когда строка i и строки j распределенной матрицы $\widetilde{K}$, а также соответствующие элементы $\widetilde{f}$ оказываются на разных вычислительных узлах, необходимо или переслать значения $\mathring{u}_i$ и номера столбца i, или предварительно упорядочить строки внутри блока, исходя из связей степеней свободы, так, чтобы предотвратить возникновение такого случая. Выбор более эффективного способа реализации зависит от числа степеней свободы, в которых заданы условия Дирихле и архитектуры вычислительной системы.

Развитием этого алгоритма является исключение уравнений $u_i = \mathring{u}_i$ из системы (1.1). Уменьшение размерности ($N < \widetilde{N}$) системы уравнений (1.3) требует переупорядочить как уравнения, так и неизвестные, распределенные по областям памяти вычислительной системы. Для снижения затрат на переупорядочение степени свободы нумеруются так, чтобы заданные оказались в конце вектора $\widetilde{u}$ или его распределенных блоков. Запишем этот (третий) алгоритм в матричном виде [3]:

$$\widehat{K}\widehat{u} = \widehat{f}, \quad \widehat{K} = \widehat{C}^T \widetilde{K} \widehat{C}, \quad \widehat{f} = \widehat{C}^T (\widetilde{f} - \widetilde{K}\mathring{u}), \tag{1.4}$$

здесь $\widehat{K} \in \mathbb{R}^{\widehat{N} \times \widehat{N}}$; $\widehat{u} \in \mathbb{R}^{\widehat{N}}$ — вектор искомых степеней свободы; $\widehat{f} \in \mathbb{R}^{\widehat{N}}$ — новый вектор правой части; $\mathring{u} \in \mathbb{R}^{\widetilde{N}}$ — вектор заданных значений степеней свободы; $\widehat{C} \in \mathbb{Z}^{\widehat{N} \times \widetilde{N}}$ — матрица перестановок, извлекающая искомые степени свободы; $\widehat{N} = \widetilde{N} - N_{\mathring{u}}$, где $N_{\mathring{u}}$ — число степеней свободы с заданными условиями Дирихле.

Элементы матрицы $\widehat{C} = [\widehat{c}_{ij}]$ принимают значения $\widehat{c}_{ij} = 1$, если $\widehat{u}_i = u_j$, здесь $\widehat{u}_i \in \widehat{u}$, $u_j \in u$; $\widehat{c}_{ij} = 0$ — в другом случае. Каждый столбец матрицы $\widehat{C}$ содержит один элемент $\widehat{c}_{ij} = 1$. Вследствие разреженности матриц $\widehat{C}$ и $\widehat{C}^T$ теоретическая сложность вычисления $\widehat{K}$ в (1.4) составляет $O(\widehat{N} \cdot N + \widehat{N}^2)$. Векторы решения систем (1.3) и (1.4) связаны выражением $u = \widehat{C}\widehat{u} + \mathring{u}$.

Остановимся на особенностях хранения матриц в приведенных алгоритмах. Второй и третий алгоритмы не предполагают хранения матрицы $\widetilde{K}$ без нулевых элементов, потому что в этих случаях требуется удалить часть ненулевых элементов матрицы. Так, во втором алгоритме удаляются ненулевые элементы в строках $\widetilde{K}$, что требует существенных изменений в структуре

данных хранения матрицы, а в третьем алгоритме, кроме того, удаляются строки. Поэтому в этих алгоритмах в компактных форматах хранятся матрицы K, $\widehat{K}$, $\widehat{C}$ и $\widehat{C}^T$. Соответственно, алгоритм с умножением на большое число такого ограничения не имеет.

Применяя первый и второй из рассмотренных алгоритмов к локальным матрицам $\widetilde{K}_e \in \in \mathbb{R}^{N_e \times N_e}$ и векторам $\widetilde{f}_e \in \mathbb{R}^{N_e}$ отдельных конечных элементов $e = \overline{1,m}$, получим соответствующие поэлементные варианты алгоритмов. Здесь m — число конечных элементов, а N_e — число степеней свободы в конечном элементе e. Задание граничных условий Дирихле на уровне локальных матриц и векторов позволяет применять так называемые matrix-free схемы решения системы (1.3), в которых матрица K не формируется. Например, поэлементные схемы [6], в которых сборка (1.2) заменяется сборкой векторов на этапе решения системы.

§ 2. Учет граничных условий Дирихле в ходе решения системы уравнений

Как правило, существующие варианты задания граничных условий Дирихле в процессе решения конечно-элементной системы линейных алгебраических уравнений (СЛАУ) связаны с матрично-векторными произведениями в явных методах подпространств А.Н. Крылова [4]. В случае неявных методов граничные условия Дирихле должны быть заданы к моменту построения предобуславливателя, а значит до начала решения системы уравнений.

Рассмотрим вариант на основе третьего алгоритма из предыдущего параграфа. Правую часть (1.4) преобразуем до решения системы уравнений. Далее произведение $\widehat{K}\widehat{p}$, вычисляемое на каждой итерации метода решения СЛАУ, запишем в виде последовательности матрично-векторных произведений: $\widetilde{p} = \widehat{C}\widehat{p}$, $\widetilde{q} = \widetilde{K}\widetilde{p}$, $\widehat{q} = \widehat{C}^T\widetilde{q}$. Таким образом, матричные произведения из (1.4) заменяются на матрично-векторные, которые благодаря разреженности матриц $\widehat{C}$ и $\widehat{C}^T$ имеют сложность $O(N)$ и $O(\widehat{N})$ соответственно.

На примере метода сопряженных градиентов (МСГ) рассмотрим более известный вариант задания граничных условий Дирихле [4]. Если задано значение $u_i = \mathring{u}_i = 0$ степени свободы i, то полагается, что компоненты i векторов p и q равны нулю на каждой итерации МСГ, здесь $q = \widetilde{K}p$, $p = \{p_i\}$, $q = \{q_i\}$, $p, q \in \mathbb{R}^N$. В случае неоднородных граничных условий Дирихле столбец i матрицы $\widetilde{K}$ умножается на $\mathring{u}_i$ и вычитается из вектора правой части $\widetilde{f}$. Так поступаем с каждым заданным значением степени свободы. Далее, на каждой итерации метода сопряженных градиентов полагаем $p_i = 0$ и $q_i = 0$. Этот подход также применим к поэлементным схемам решения конечно-элементных систем.

Следует отметить, что даже для диагонального предобуславливания требуется сборка диагональной части матрицы K с заданными граничными условиями Дирихле. Поэтому во втором алгоритме не будем заменять значения $\widetilde{k}_{ii}$ на 1, а в правой части зададим $\widetilde{f}_i = \mathring{u}_i\widetilde{k}_{ii}$. В таком варианте $\operatorname{diag}(K) = \operatorname{diag}(\widetilde{K})$, что позволяет построить и достаточно эффективно применить на гибридной архитектуре диагональный предобуславливатель.

§ 3. Особенности программной реализации

В зависимости от выбранной вычислительной схемы отображение конечно-элементных вычислений по процессорам в том или ином виде наследует разделение расчетной сетки [2]. Для поэлементных схем результатом разделения являются подмножества из m_i, $i = 1, \ldots, n_p$, конечных элементов, пересекающихся по общим узлам сетки, здесь n_p — число параллельных процессов, равное числу центральных процессоров. Это разделение наследуют подмножества матриц конечных элементов $\{\widetilde{K}_e\}_{e=1}^{m_i}$, вычисление которых распределяется между CPU и GPU. Если численное интегрирование локальных матриц осуществляется на GPU, в том числе на CPU и GPU [5], то матрицы $\widetilde{K}_e$ размещаются в памяти ускорителей (все или часть матриц) в массиве для хранения соответствующего подмножества матриц.

Рассмотрим отображение вычислений по параллельным процессам гибридной вычислительной системы. На кластерах гибридной архитектуры реализуется гибридная модель параллелизма (обмен сообщениями + общая память + параллелизм данных), которая, как правило, реализуется при помощи нескольких технологий параллельного программирования: MPI, OpenMP, CUDA. В рамках этой модели каждому подмножеству конечных элементов ставим в соответствие один из n_p процессов MPI, в котором создается несколько нитей OpenMP (равное числу

ядер CPU). Мастер-нить OpenMP обеспечивает MPI-коммуникации между вычислительными узлами кластера. Другие нити обеспечивают параллелизм при передаче данных между ядрами CPU и несколькими GPU внутри узла, запуск kernel-функций CUDA и вычисления на ядрах центрального процессора.

Кратко остановимся на реализации поэлементного варианта второго алгоритма. На GPU условия Дирихле задаются при помощи kernel-функции CUDA, содержащей программный код вычислений для отдельного конечного элемента. При запуске kernel-функции каждой нити CUDA с номером e ставится в соответствие матрица $\widetilde{K}_e$ и вектор $\widetilde{f}^e$ конечного элемента e, находящегося в памяти ускорителя. Результатом выполнение kernel-функций является множество матриц $\{K_e\}_{e=1}^{m_i}$ и вектор f, составленный из векторов f_e с заданными условиями Дирихле.

Относительно схем с блочным разделением выше было отмечено, что вычисления матрицы $\widehat{K}$ и вектора $\widehat{f}$ в (1.4) состоят из двух произведений матриц, двух матрично-векторных произведений и разности векторов, все перечисленные операции возможно эффективно выполнить на GPU. Вместе с тем при реализации третьего алгоритма на этапе формирования системы в памяти графического ускорителя необходимо кроме блоков матрицы $\widetilde{K}$ хранить также соответствующие блоки матриц $\widehat{K}$, $\widehat{C}$ и $\widehat{C}^T$, а при решении прикладных задач возможны случаи, когда $N_{\mathring{u}} \ll \widetilde{N}$ и, соответственно, $\widehat{N} \approx \widetilde{N}$. Этот вариант задания условий первого рода более перспективен на этапе решения системы уравнений, где дополнительно потребуется хранить только блоки матриц $\widehat{C}$ и $\widehat{C}^T$, что благодаря разряженности $\widehat{C}$ и $\widehat{C}^T$ не существенно отличается от хранения только блоков $\widetilde{K}$ и поэтому требует дальнейших исследований.

Как показал опыт исследований, граничные условия первого рода предпочтительнее задавать на этапе формирования конечно-элементных систем. Таким образом, предлагаемые варианты алгоритмов задания граничных условий Дирихле и их реализации на GPU исключают пересылки подмножеств матриц K_e и блоков матрицы K между вычислительными узлами, оперативной памятью и памятью ускорителя, сборку системы уравнений и ее разделение на блоки, увеличивая ускорение конечно-элементных вычислений на гибридных архитектурах в десятки раз, что подтверждается результатами вычислительных экспериментов.

Список литературы

1. Норри Д., де Фриз Ж. Введение в метод конечных элементов. М.: Мир. 1981. 304 с.
2. Копысов С.П., Новиков А.К. Метод декомпозиции для параллельного адаптивного конечно-элементного алгоритма // Вестник Удмуртского университета. Математика. Механика. Компьютерные науки. 2010. № 3. С. 141–154.
3. Даутов Р.З. Программирование МКЭ в MATLAB. Казань: КГУ, 2010. 71 с.
4. Шабров Н.Н. Метод конечных элементов в расчетах деталей тепловых двигателей. Л.: Машиностроение, 1983. 212 с.
5. Новиков А.К., Копысов С.П., Кузьмин И.М., Недожогин Н.С. Формирование матриц конечно-элементных систем в GPGPU // Вестник Нижегородского университета им. Н.И. Лобачевского. 2014. № 3 (1). С. 120–125.
6. Копысов С.П., Кузьмин И.М., Недожогин Н.С., Новиков А.К., Рычков В.Н., Сагдеева Ю.А., Тонков Л.Е. Параллельная реализация конечно-элементных алгоритмов на графических ускорителях в программном комплексе FEStudio // Компьютерные исследования и моделирование. 2014. Т. 6. № 1. С. 79–97.

Поступила в редакцию 01.10.2015

Новиков Александр Константинович, к. ф.-м. н., старший научный сотрудник, Институт механики УрО РАН, 426067, Россия, г. Ижевск, ул. Т. Барамзиной, 34.
E-mail: sc_work@mail.ru

Кузьмин Игорь Михайлович, младший научный сотрудник, Институт механики УрО РАН, 426067, Россия, г. Ижевск, ул. Т. Барамзиной, 34.
E-mail: imkuzmin@gmail.com

Недожогин Никита Сергеевич, младший научный сотрудник, Институт механики УрО РАН, 426067, Россия, г. Ижевск, ул. Т. Барамзиной, 34.
E-mail: nedozhogin@inbox.ru

A. K. Novikov, I. M. Kuz'min, N. S. Nedozhogin

Several features of the finite element technologies for hybrid computational architecture

Keywords: parallel computation, finite elements method, Dirichlet boundary conditions, hybrid computational architecture.

MSC: 65M60, 65Y05

We consider the features of parallel finite element computation using hybrid architectures related to the boundary conditions of the first kind (the Dirichlet conditions). Variants of the Dirichlet conditions setting in finite element system of equations forming and solving are compared to ensure the preservation of the symmetry of the coefficient matrix.

REFERENCES

1. Norrie D.H., de Vriz G. *An introduction to finite element analysis*, London: Academic Press, 1978, 301 p. Translated under the title *Vvedenie v metod konechnykh elementov*, Moscow: Mir, 1981, 304 p.
2. Kopysov S.P., Novikov A.K. Domain decomposition for parallel adaptive finite element algorithm, *Vestnik Udmurtskogo Universiteta. Matematika. Mekhanika. Komp'yuternye nauki*, 2010, no. 3, pp. 141–154 (in Russian).
3. Dautov R.Z. *Programmirovanie MKE v MATLAB* (FEM programming in MATLAB), Kazan: Kazan State University, 2010, 71 p.
4. Shabrov N.N. *Metod konechnykh elementov v raschetakh detalei teplovykh dvigatelei* (Finite element method in the calculation of parts of heat engines), Leningrad: Mashinostroenie, 1983, 212 p.
5. Novikov A.K., Kopysov S.P., Kuzmin I.M., Nedozhogin N.S. Forming matrices of finite element systems in GPGPU, *Vestnik Nizhegorodskogo Universiteta im. N.I. Lobachevskogo*, 2014, no. 3 (1), pp. 120–125 (in Russian).
6. Kopysov S.P., Kuzmin I.M., Nedozhogin N.S., Novikov A.K., Rychkov V.N., Sagdeeva Yu.A., Tonkov L.E. Parallel implementation of a finite-element algorithms on a graphics accelerator in the software package FEStudio, *Komp'yuternye Issledovaniya i Modelirovanie*, 2014, vol. 6, no. 1, pp. 79–97 (in Russian).

Received 01.10.2015

Novikov Aleksandr Konstantinovich, Candidate of Physics and Mathematics, Associate Professor, Department of Computational Mechanics, Udmurt State University, ul. Universitetskaya, 1, Izhevsk, 426034, Russia; Senior Researcher, Institute of Mechanics, Ural Branch of RAS, ul. T. Baramzinoi, 34, Izhevsk, 426067, Russia.
E-mail: sc_work@mail.ru

Kuz'min Igor' Mikhailovich, Junior Researcher, Institute of Mechanics, Ural Branch of RAS, ul. T. Baramzinoi, 34, Izhevsk, 426067, Russia.
E-mail: imkuzmin@gmail.com

Nedozhogin Nikita Sergeevich, Junior Researcher, Institute of Mechanics, Ural Branch of RAS, ul. T. Baramzinoi, 34, Izhevsk, 426067, Russia.
E-mail: nedozhogin@inbox.ru