УДК 004.02

Чопоров С. В.

Канд. техн. наук, доцент, Запорожский национальный университет, Украина, E-mail: choporov@list.ru

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИЙ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В МЕТОДЕ КОНЕЧНЫХ ЭЛЕМЕНТОВ

В работе рассмотрена проблема использования параллельных вычислений в методе конечных элементов. Рассмотрены особенности построения матрицы жесткости и решения полученной системы линейных уравнений на базе параллельных вычислений. Проведены вычислительные эксперименты, результаты которых использованы для анализа эффективности предложенных подходов.

Ключевые слова: метод конечных элементов, параллельные вычисления, математическая модель, вычислительный метод.

ВВЕДЕНИЕ

Проектирование современных деталей и конструкций связано с необходимостью применения вычислительных методов для решения задач высокой сложности. В разработке САПР для машиностроения наиболее активно применяется один из наиболее распространенных вычислительных методов – метод конечных элементов, который находит свои приложения в широком спектре задач: от анализа напряженно деформированного состояния деталей, конструкций и строений, переноса тепла, гидродинамики до анализа микроэлектромеханических систем [1–3].

Сложность многих деталей и конструкций приводит к возникновению на подготовительных этапах сложных геометрических моделей – математических моделей соответствующих геометрических областей. Последующее применение методов дискретизации [4] с требованием учета особенностей геометрической модели и физической постановки задачи (например, сгущение сетки конечных элементов в областях контакта, трещин, учет многослойности и другие) может приводить к дискретным моделям с большим числом конечных элементов. В результате для моделирования необходимы значительные вычислительные ресурсы и, как следствие, временные затраты.

Естественным решением задачи минимизации вычислительных затрат является оптимизация числа элементов в дискретных моделях. Однако, в тех случаях, когда уменьшение числа конечных элементов приводит к неоправданной потере точности модели, одним из путей оптимизации временных затрат является использование параллельных вычислений. Стремительное увеличение числа функциональных устройств (процессоров, ядер) в современных вычислительных системах актуализирует данное направление исследований.

Таким образом, целью работы является разработка параллельных алгоритмов для основных этапов метода конечных элементов и анализ результатов их внедрения путем проведения вычислительных экспериментов.

1 ТИПЫ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И ОСНОВНЫЕ ТЕХНОЛОГИИ ДОСТИЖЕНИЯ ПАРАЛЛЕЛЬНОСТИ

Одной из наиболее распространенных классификаций архитектуры вычислительных систем является систематика Флинна [5], которая основана на анализе способов взаимодействия функциональных вычислительных устройств и обрабатываемых данных. В результате различаются основные четыре типа вычислительных систем [5–7]:

– SISD (Single Instruction, Single Data) – вычислительные системы, в которых используется единственное функциональное устройство для исполнения одного потока инструкций с одиночным потоком данных;

– SIMD (Single Instruction, Multiple Data) – системы с одним потоком инструкций, в которых одна инструкция может применяться к векторам данных (нескольким потокам данных);

– MISD (Multiple Instruction, Single Data) – системы с множественным потоком инструкций и одним потоком данных;

– MIMD (Multiple Instruction, Multiple Data) – вычислительные системы с множественным потоком инструкций и множественным потоком данных.

Большинство современных последовательных вычислительных систем можно отнести к первым двум типам. Такие компьютеры оснащены одним функциональным устройством и поддерживают различные наборы SIMD-инструкций (например, MMX, SSE, SSE2 и другие). Параллельность (квазипараллельность) в таких системах является реализацией многозадачности в планировщике задач операционной системы.

К типу MISD можно отнести отказоустойчивые вычислительные системы, которые выполняют инструкции с функциональной избыточностью с целью обнаружения ошибок. При этом ряд авторов [5] называют данный тип теоретическим, отмечая отсутствие устройств такого типа.

 DOI 10.15588/1607-3274-2013-2-15

К типу MIMD относятся системы с множеством функциональных устройств: мультипроцессоры (многопроцессорные и многоядерные системы) и мультикомпьютеры (распределенные вычислительные системы, кластеры).

На современном рынке вычислительных систем в наиболее доступном ценовом диапазоне широко представлены мультипроцессорные системы с общей памятью, которые содержат от 2 до 16 функциональных устройств. Такие вычислительные системы можно использовать для начальной отладки параллельных алгоритмов и оценки их эффективности. Для программирования в таких системах наиболее распространенными являются решения на базе потоков (например, POSIX Threads, Boost.Thread и другие) или стандарта OpenMP (Open Multi-Processing) [8, 9].

2 ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ МЕТОДА КОНЕЧНЫХ ЭЛЕМЕНТОВ

Основная идея метода конечных элементов состоит в том, что любую искомую непрерывную величину (например, температуру, перемещения и другие) можно аппроксимировать при помощи кусочно-непрерывных функций, определенных на элементарных геометрических областях – конечных элементах, образующих дискретную модель области. Для определения значений искомой непрерывной величины в узлах дискретной модели строится система линейных алгебраических уравнений, матрица которой носит название глобальной матрицы жесткости.

Процесс построения глобальной матрицы жесткости называется ансамблированием и является суммой, подчиненной особым правилам, локальных матриц жесткости – результатов интегрирования дифференциального уравнения в матричной форме. Размер глобальной матрицы жесткости равен произведению числа узлов в дискретной модели на число степеней свободы в узле (определяется типом задачи).

Таким образом, в случае достаточно сложных дискретных моделей, этапы построения глобальной матрицы жесткости и последующего решения СЛАУ являются наиболее требовательными к вычислительным ресурсам. Следовательно, внедрение параллельных вычислений при выполнении этих этапов способно дать наиболее значимый результат с точки зрения экономии времени на проектирование и моделирование.

Первым шагом после этапов загрузки и инициализации исходных данных параллельной реализации метода конечных элементов является определение величины K – числа доступных функциональных устройств (рис. 1). Данная величина является основой для определения количества параллельных потоков. В методе конечных элементов при вычислении глобальной матрицы жесткости в случае мультипроцессорных систем операции ввода-вывода используются только, если недостаточно основной памяти для хранения исходных данных и самой глобальной матрицы жесткости. То есть, при использовании сжатого способа хранения глобальной матрицы жесткости (которая, как правило, является сильно разреженной), время ожидания операций ввода-вывода, в зависимости от сложности задачи, будет минимальным, следовательно, на практике величина K также определяет количество параллельных вычислительных потоков.

Так как ансамблирование является обобщением суммы, то каждый из K параллельных потоков должен производить ансамблирование N / K элементов (N – количество элементов в дискретной модели). При этом все данные (включая переменные циклов), связанные с интегрированием на конечном элементе должны быть локальными данными параллельных потоков (быть доступными только для потока-обладателя; схема приведена на рис. 1).

Шаг обновления глобальной матрицы жесткости, учитывая сжатый формат хранения данных, является критическим с точки зрения потокобезопасности. То есть, доступ остальным потокам к глобальной матрице жесткости должен блокироваться пока один поток модифицирует ее элементы.

Время, необходимое для построения глобальной матрицы жесткости при помощи параллельного вычислительного процесса, можно оценить величиной T_p

$$T_p = N\left(I / K + m\right) + K \cdot t_{thread},$$

где I – процессорное время, необходимое для интегрирования исходного дифференциального уравнения на одном конечном элементе, m – процессорное время, необходимое для ансамблирования результатов интегрирования на одном элементе, t_{thread} – время, необходимое для инициализации одного вычислительного потока.

Аналогично, время, необходимое для построения глобальной матрицы жесткости на базе последовательного алгоритма, можно оценить величиной T_s

$$T_s = N\left(I + m\right).$$

Применение параллельных вычислений имеет смысл, если $T_s > T_p$:

$$N\left(I + m\right) > N\left(I / K + m\right) + K \cdot t_{thread},$$

в результате упрощения которой получим следующую оценку

$$N \cdot I > \frac{K^2 \cdot t_{thread}}{K - 1}.$$

Таким образом, например, для двух параллельных потоков, произведение количества элементов на время интегрирования одного элемента должно быть вчетверо большим времени инициализации параллельных потоков, что может не выполняться при малом числе конечных элементов.

Для решения полученных в результате построения глобальной матрицы жесткости СЛАУ на практике, как правило, используются итерационные методы (напри-

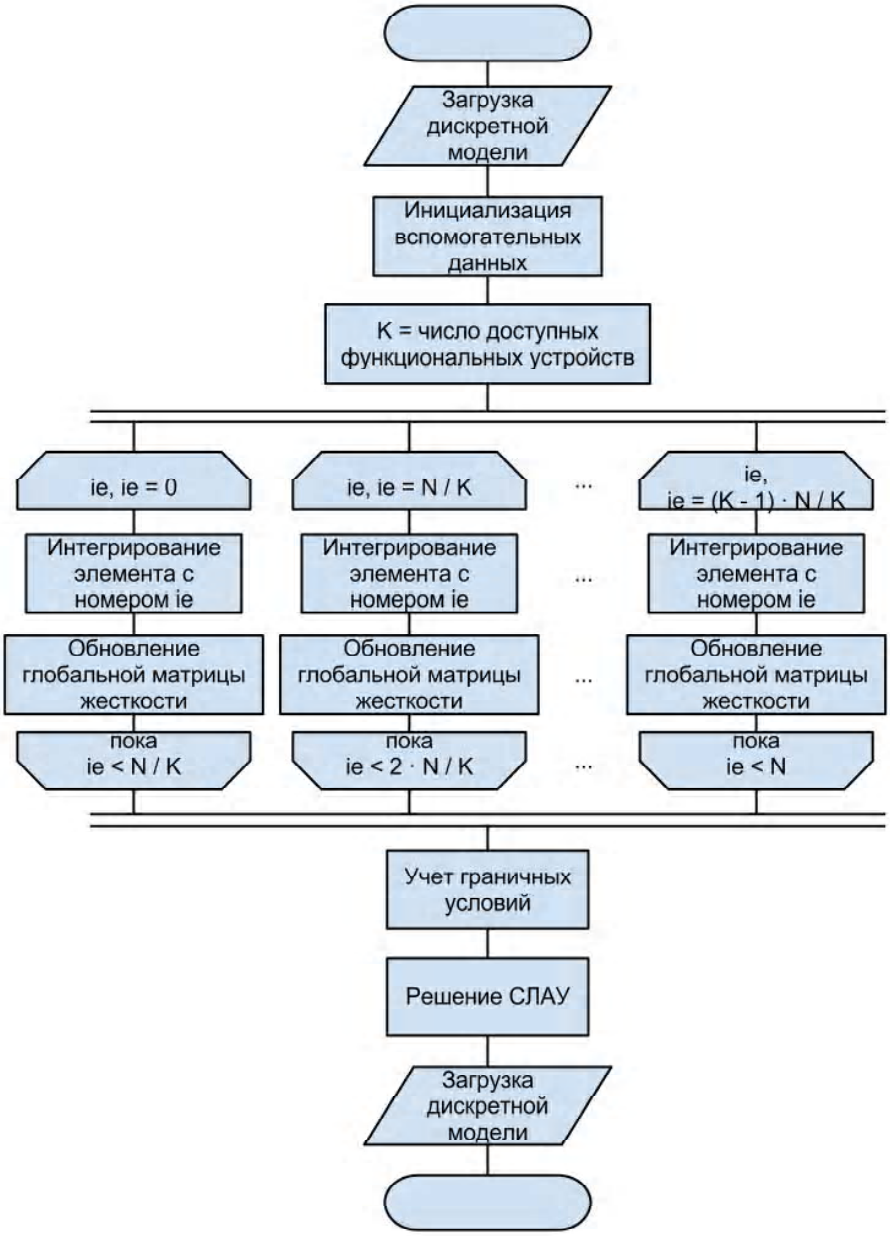

Рис. 1. Схема метода конечных элементов на базе параллельных вычислений

мер, метод сопряженных градиентов [10, 11] и другие). Их ключевой особенностью является то, что результаты текущей итерации являются исходными данными для следующей. Следовательно, для сокращения времени поиска решения необходимо ускорять внутренние операции метода: произведение матрицы на вектор, поэлементное произведение векторов, вычисления невязки и другие.

С учетом того, что в результате ансамблирования глобальная матрица жесткости получается сильно разреженной, следовательно, схема вычисления произведения матрицы на вектор в K параллельных потоков примет вид, приведенный на рис. 2. Общими для потоков данными являются матрица A, размер которой $D \times D$, вектор x, вектор-результат r и число K. Так как вычислительными потоками обновляются разные элементы вектора r, то отсутствует необходимость в блокировке для синхронизации потоков. Счетчики циклов i и j, переменная t – локальные для каждого вычислительного потока. Во внутренних циклах по j вычисления производятся только для ненулевых элементов i-й строки матрицы A.

3 ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

В качестве тестового стенда использовалась программно-аппаратная конфигурация с процессором Intel Core i3-380 (2,53 ГГц, 2 физических ядра + 2 виртуальных ядра) с 3 ГБ ОЗУ под управлением операционной системы openSUSE 12.2 (компилятор gcc 4.7), для создания потоков и управления ими – библиотеки стандарта openMP.

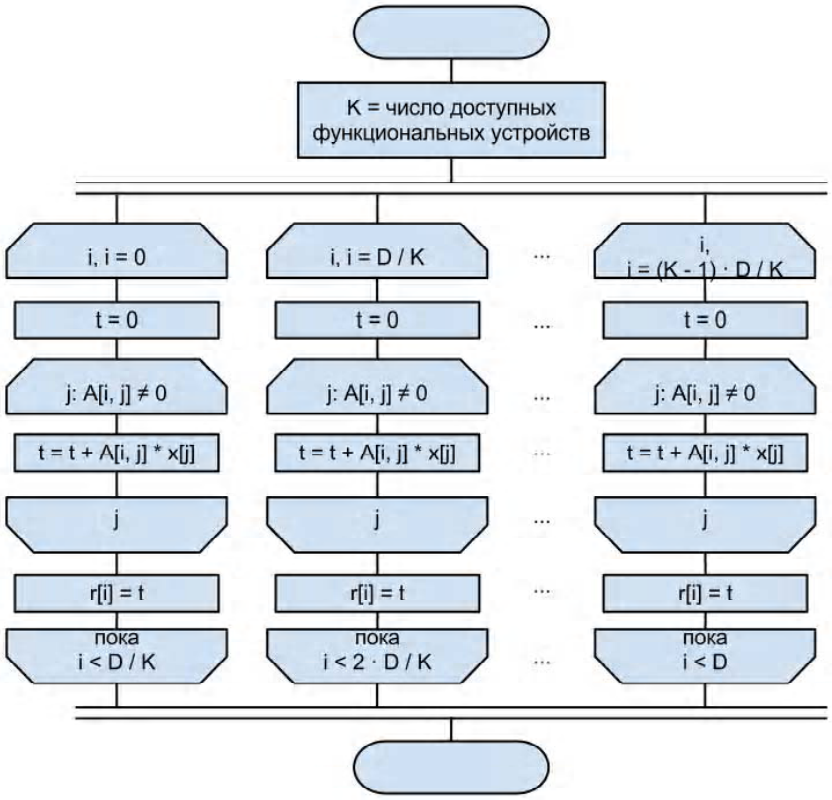

Рис. 2. Схема параллельного произведения разреженной матрицы на вектор

1. Задача Дирихле

Рассмотрим решение уравнения Лапласа с условиями Дирихле.

$$\frac{\partial^2 U}{\partial x^2}+\frac{\partial^2 U}{\partial y^2}+\frac{\partial^2 U}{\partial z^2}=0,$$
$$U_{\delta\Omega}=f(x,y,z).$$

В качестве области Ω рассмотрим косозубое зубчатое колесо (дискретная модель: 49500 узлов, 40920 шестигранных конечных элементов, рис. 3). Функция $f(x,y,z)$ пусть имеет вид

$$f(x,y,z)=\begin{cases}0, \text{в узлах центрального отверстия},\\ 4\sin^2(5\varphi), \text{на поверхности зубьев},\end{cases}$$

где φ – угол поворота полярного радиуса до соответствующего узла в плоскости Oxy.

Формула для вычисления локальной матрицы жесткости $[K^e]$ примет вид

$$\left[K^e\right]=\iiint\left(\begin{Bmatrix}\frac{\partial H_1}{\partial x}\\ \frac{\partial H_2}{\partial x}\\ \dots\\ \frac{\partial H_8}{\partial x}\end{Bmatrix}\begin{bmatrix}\frac{\partial H_1}{\partial x} & \frac{\partial H_2}{\partial x} & \dots & \frac{\partial H_8}{\partial x}\end{bmatrix}+\right.$$
$$+\begin{Bmatrix}\frac{\partial H_1}{\partial y}\\ \frac{\partial H_2}{\partial y}\\ \dots\\ \frac{\partial H_8}{\partial y}\end{Bmatrix}\begin{bmatrix}\frac{\partial H_1}{\partial y} & \frac{\partial H_2}{\partial y} & \dots & \frac{\partial H_8}{\partial y}\end{bmatrix}+$$
$$\left.+\begin{Bmatrix}\frac{\partial H_1}{\partial z}\\ \frac{\partial H_2}{\partial z}\\ \dots\\ \frac{\partial H_8}{\partial z}\end{Bmatrix}\begin{bmatrix}\frac{\partial H_1}{\partial z} & \frac{\partial H_2}{\partial z} & \dots & \frac{\partial H_8}{\partial z}\end{bmatrix}\right)dxdydz,$$

где H_i (i = 1, 8) – функции формы конечного элемента.

В результате вычислительного эксперимента получены результаты, приведенные в табл. 1 (время в секундах).

2. Исследование прогиба трехслойной круглой оболочки под равномерной нагрузкой

Дана защемленная по контуру трехслойная оболочка, радиус которой 0,4 м, толщина – 0,01 м. Внутренний слой: толщина – 0,008 м, модуль Юнга – E_i = 72017,3327 МПа, коэффициент Пуассона – ν_i = 0,2999518536. Внешние

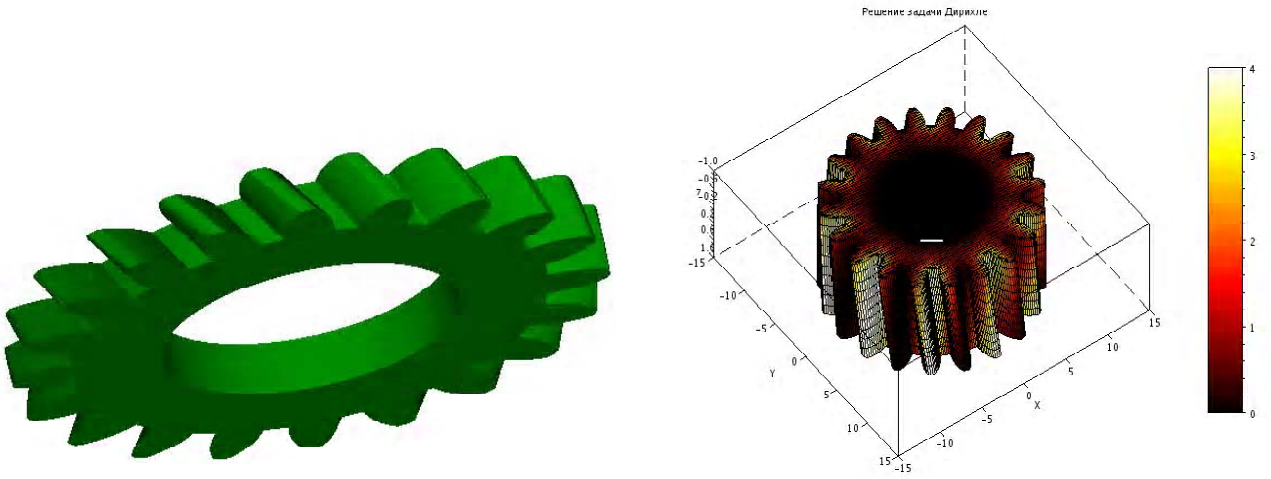

Рис. 3. Задача Дирихле: косозубое зубчатое колесо

слои: толщина – 0,001 м, модуль Юнга – E_o = 203200 МПа, коэффициент Пуассона – ν_o = 0,27. На оболочку действует равномерная нагрузка q = –0,05 МПа.

Для решения построена дискретная модель сектора оболочки (рис. 4) на базе десяти слоев шестигранных конечных элементов (59213 узлов и 52240 элементов), толщина которых 0,001 м, что позволяет рассматривать данную задачу в трехмерной постановке, используя соответствующие слоям упругие константы.

Формула для вычисления локальной матрицы жесткости $[K^e]$ примет вид

$$\left\lfloor K^e \right\rfloor = \iiint B^T DB dx dy dz,$$

где

$$B = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \times$$

$$\times \begin{bmatrix} H_1 & H_2 & \dots & H_8 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & H_1 & H_2 & \dots & H_8 & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & H_1 & H_2 & \dots & H_8 \end{bmatrix},$$

Таблица 1. Решение задачи Дирихле

Этап	Количество вычислительных потоков		
	1	2	4
	время, с		
Построение глобальной матрицы жесткости	3,07639	1,56709	1,25949
Решение СЛАУ, 73 итерации	0,304023	0,240893	0,264904

$$D = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \times$$

$$\times \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix}$$

$$E = \begin{cases} E_o, z \ge 0{,}009 \text{ или } z \le 0{,}001, \\ E_i, 0{,}001 \le z \le 0{,}009, \end{cases}$$

$$\nu = \begin{cases} \nu_o, z \ge 0{,}009 \text{ или } z \le 0{,}001, \\ \nu_i, 0{,}001 \le z \le 0{,}009. \end{cases}$$

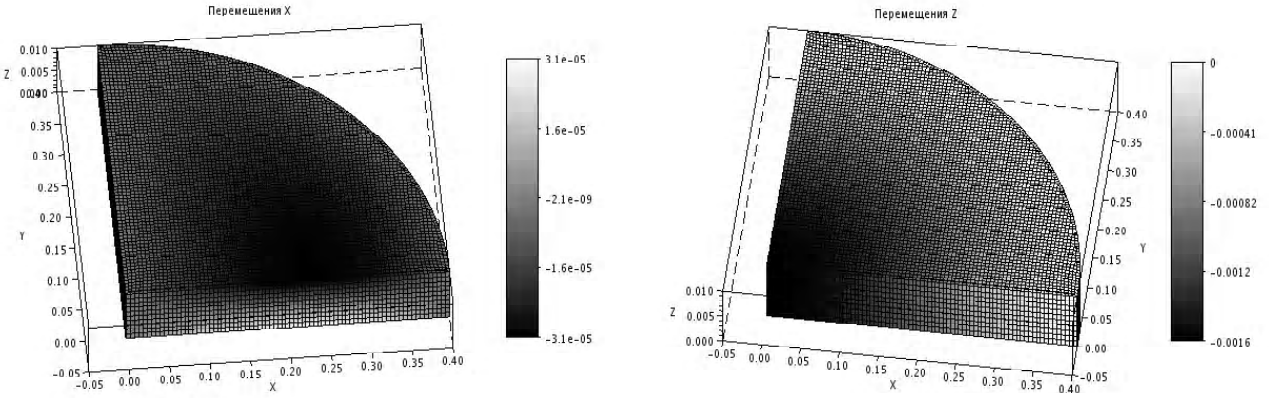

Рис. 4. Прогиб трехслойной оболочки

В результате вычислительного эксперимента получены результаты, приведенные в табл. 2 (время в секундах).

Таблица 2. Исследование прогиба трехслойной круглой оболочки

Этап	Количество вычислительных потоков		
	1	2	4
	время, с		
Построение глобальной матрицы жесткости	83,7293	41,5063	31,7687
Решение СЛАУ, 6952 итерации	298,287	191,299	189,134

ВЫВОДЫ

Таким образом, вычислительные эксперименты показали, что использование параллельных вычислений в методе конечных элементов позволяет сократить время на проектирование и моделирование деталей и конструкций. Из таблиц видно незначительное преимущество во времени при использовании четырех вычислительных потоков относительно двух, что может быть связано с использованием двух виртуальных ядер (которые не обладают аналогичной двум физическим ядрам производительностью). Реализация разработанных алгоритмов на базе библиотек стандарта openMP позволяет рассматривать полученную в ходе исследования кодовую базу в качестве основы для построения кластерной реализации. Перспективным направлением исследований в данном направлении является изучение влияния количества потоков на производительность (или затраченное время). Также важным направлением является изучение производительности многопоточных вычислений в зависимости от количества узлов/элементов в сетке, особенно в системах с раздельной памятью (вычислительных кластерах), в которых существуют значительные затраты на синхронизацию вычислительных потоков.

СПИСОК ЛИТЕРАТУРЫ

1. *Zienkiewicz, O. Z.* The Finite Element Method. V. 1: The Basis / O. Z. Zienkiewicz, R. L. Taylor. – Oxford : Butterworth-Heinemann, 2000. – 689 p.
2. *Smith, I. M.* Programming the finite element method / I. M. Smith, D. V. Griffits. – Chichester : Wiley, 2004. – 628 p.
3. *Ahmad, A.* Parallel Programming in the Finite Element Methos / Atique Ahmad // In. Proceedings of Failure of engineering Materials and Structures, UET Taxila, October 2007. – P. 87–93.
4. *Thompson, J. F.* Hand book of grid generation / Joe F. Thompson, Bharat K. Soni, Nigel P. Weatheril. – New York : CRC Press, 1999. – 1200 p.
5. *Гергель, В. П.* Основы параллельных вычислений для многопроцессорных вычислительных систем / В. П. Гергель, Р. Г. Стронгин. – Нижний Новгород : ННГУ им. Н.И. Лобачевского, 2003. – 184 с.
6. *Корнеев, В. В.* Параллельные вычислительные системы / В. В. Корнеев. – М. : Нолидж, 1999. – 320 с.
7. *Hwang, K.* Scalable Parallel Computing: Technology, Architecture, Programming / Kai Hwang, Zhiwei Xu. – New York : McGraw-Hill, 1998. – 802 p.
8. *Антонов, А. С.* Параллельное программирование с использованием технологии OpenMP : учебное пособие / А. С. Антонов. – М. : МГУ, 2009. – 77 с.
9. *Chapman, B.* Using OpenMP : portable shared memory parallel programming / Barbara Chapman. – Cambridge : MIT, 2008. – 353 p.
10. *Notay, Y.* Flexible Conjugate Gradients / Yvan Notay // SIAM Journal on Scientific Computing. – 2001. – Volume 22, Issue 4. – P. 1444–1460.
11. *Knyazev, A. V.* Steepest Descent and Conjugate Gradient Methods with Variable Preconditioning / Andrew V. Knyazev, Ilya Lashuk // SIAM Journal on Matrix Analysis and Applications. – 2008. – Volume 29, Issue 4. – P. 1267–1280.

Стаття надійшла до редакції 15.02.2013.

Чопоров С. В.
Канд. техн. наук, доцент, Запорізький національний університет, Україна

ВИКОРИСТАННЯ ТЕХНОЛОГІЙ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ У МЕТОДІ СКІНЧЕННИХ ЕЛЕМЕНТІВ

В роботі розглянута проблема впровадження паралельних обчислень у метод скінченних елементів. Розглянуто особливості побудови матриці жорсткості та вирішення отриманої системи лінійних рівнянь на базі паралельних обчислень. Проведено обчислювальні експерименти, які використані для аналізу ефективності запропонованих підходів.

Ключові слова: метод скінченних елементів, паралельні обчислення, математична модель, обчислювальний метод.

Choporov S. V.
Ph.D. Candidate, associate Professor, Zaporizhzhya National University, Ukraine

PARALLEL COMPUTING TECHNOLOGIES IN THE FINITE ELEMENT METHOD

Nowadays engineers and researchers are faced with solving very complex problems in a mathematical modeling and design. Numerical analysis naturally finds applications in all fields of engineering and the physical sciences. The finite element method is a powerful tool for the numerical simulation of a wide range of problems. Implementation of the finite element method in CAD systems on the basis of modern computers allows researchers to solve large scale problems.

The finite element method uses a discretization of a continuum domain into a mesh as the starting point. The discretization of complex domains may give large numbers of elements, thus increasing the requirements of computer memory and speed.

Modern parallel computers use multiple processing elements simultaneously to solve a problem. Thus implementation of parallel computing into the finite element method is urgent direction of the research.

In the finite element method for the numerical solution of partial differential equations, the stiffness matrix represents the system of linear equations that must be solved in order to ascertain an approximate solution to the differential equation. Therefore the article describes parallel algorithms for assembly of stiffness matrix and for solution of linear equations.

Also this article contains two numerical experiments: 1) Dirichlet problem on the complex domain (gear); 2) Elasticity problem of three-layer shell.

In the end of the article author compares the speed of a solution of these problems with one, two and four parallel cores and makes discussion about the effectiveness of parallel finite element processing.

Keywords: finite element method, parallel computing technologies, mathematical model, numerical method.

REFERENCES

1. Zienkiewicz O. Z., Taylor R.L. The Finite Element Method. V. 1: The Basis. Oxford, Butterworth-Heinemann, 2000, 689 p.
2. Smith I. M., Griffits D. V. Programming the finite element method. Chichester, Wiley, 2004, 628 p.
3. Ahmad A. Parallel Programming in the Finite Element Methos, *In. Proceedings of Failure of engineering Materials and Structures*, UET Taxila, October 2007, pp. 87–93.
4. Thompson J. F., Bharat K. Soni, Nigel P. Weatheril. Hand book of grid generation. New York, CRC Press, 1999, 1200 p.
5. Gergel' V. P., Strongin R. G. Osnovy parallel'nyx vychislenij dlya mnogoprocessornyx vychislitel'nyx system. Nizhnij Novgorod, NNGU im. N. I. Lobachevskogo, 2003, 184 p.
6. Korneev V. V. Parallel'nye vychislitel'nye sistemy. Moscow, Nolidzh, 1999, 320 p.
7. Hwang K., Zhiwei Xu. Scalable Parallel Computing: Technology, Architecture, Programming. New York, McGraw-Hill, 1998, 802 p.
8. Antonov A. S. Parallel'noe programmirovanie s ispol'zovaniem texnologii OpenMP: Uchebnoe posobie. Moscow, MGU, 2009, 77 p.
9. Chapman B. Using OpenMP : portable shared memory parallel programming. Cambridge, MIT, 2008, 353 p.
10. Notay Y. Flexible Conjugate Gradients, *SIAM Journal on Scientific Computing,* 2001, Volume 22, Issue 4, pp. 1444–1460.
11. Knyazev A.V., Lashuk Ilya Steepest Descent and Conjugate Gradient Methods with Variable Preconditioning, *SIAM Journal on Matrix Analysis and Applications,* 2008, Volume 29, Issue 4, pp. 1267–1280.