# ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ И ТЕХНОЛОГИИ РЕШЕНИЯ СОПРЯЖЕННЫХ ЗАДАЧ FSI

***С.П. Копысов, И.М. Кузьмин, В.Н. Рычков, Л.Е. Тонков***

*Институт механики УрО РАН, Ижевск*

Рассматривается подход к созданию универсальной открытой программной среды, предназначенной для связывания двумерных/трехмерных конечно-элементных и конечно-объемных методов, и ее дальнейшее использование в параллельных вычислениях.

## Введение

При решении сопряженных задач взаимодействия потока газа и деформируемой конструкции (FSI) в рамках разделённого подхода [1] согласование получаемых решений на границах подобластей является неотъемлемым и весьма важным этапом, существенно влияющим на адекватность получаемых результатов. Под согласованием понимается прежде всего совокупность операций по обмену результатами вычислений между соответствующими решателями физических подзадач, интерполяции значений между узлами расчётных сеток, деформирование сетки газодинамической области и, возможно, реализация семейства итерационных процедур при жестком связывании.

Актуальным является построение параллельных алгоритмов решения отдельных физических подзадач динамики деформирования и газодинамики, а также непосредственно решающих всю совокупность задач сопряжения (интерполяция, деформирование расчетных сеток, итерационное связывание) и создание прикладного программного интерфейса с выделением самостоятельного приложения для синхронизации и обмена данными, в котором реализуются отдельные подзадачи связывания.

В данной работе основное внимание уделено построению эффективных параллельных алгоритмов для гибридных вычислительных систем при решении задачи динамики деформирования, интерполяции данных, деформирования сетки и обеспечения обмена данными между отдельными подзадачами.

## 1. Решение задачи динамики деформирования

Обычно в методе конечных элементов (МКЭ) считается, что конечно-элементная сетка собрана и каждый узел принадлежит нескольким элементам. Предположим обратное, множество элементов разобрано, так что любой узел принадлежит одному элементу, и нумерация степеней свободы в элементах выполнена от $1\ldots N_e$. Тогда поэлементная сборка эффективной матрицы жесткости $\hat{K} = \sum_{e=1}^{m} C_e^T \left(K_e + a_0 M_e\right) C_e$ заменяется на $\hat{K} = C^T \tilde{K} C$ с блочно-диагональной матрицей $\tilde{K}$. Здесь $C = \left[C_e\right]_{1\times m} \in Z^{N_e \times N}$ – матрица связности, определяющая соответствие между степенями свободы элементов и независимыми перемещениями для всей области; $\hat{K} \in R^{N\times N}$ – глобальная эффективная матрица жесткости; $\tilde{K}_e \in R^{N_e \times N_e}$ – локальная (элементная) эффективная матрица жесткости, являющаяся блоком матрицы $\tilde{K} = \left[\tilde{K}_e\right]_{m\times m}$.

Таким образом, вычисления, выполняемые с матрицей $\hat{K}$, переносятся на матрицы $\tilde{K}_e$, что позволяет при перестроении системы уравнений, вызванном адаптацией сетки или локальным изменением степени аппроксимирующих полиномов, корректировать только матрицы перестраиваемых элементов и списки связности. Кроме того, применение методов подпространств Крылова без непосредственной сборки матрицы $\hat{K}$ позволяет эффективно решать системы уравнений на параллельных вычислительных системах [2, 3].

Дискретизация по времени основывается на схеме Ньюмарка. Выбирается временной шаг $\Delta t$, параметры $\alpha$ и $\delta$, по которым вычисляются постоянные $a_0, a_1, \ldots, a_7$. Формируется эффективная матрица жесткости $\hat{K} : \hat{K} = K + a_0 M$. Для каждого временного шага $t + \Delta t$ вычисляется эффективная нагрузка $\hat{R}_{t+\Delta t} = R_{t+\Delta t} + M\left(a_0 u_t + a_2 \dot{u}_t + a_3 \ddot{u}_t\right)$, находятся перемещения из решения системы $\hat{K} u_{t+\Delta t} = \hat{R}_{t+\Delta t}$. Вычисляются ускорения и скорости: $\ddot{u}_{t+\Delta t} = a_0\left(u_{t+\Delta t} - u_t\right) - a_2 \dot{u}_t - a_3 \ddot{u}_t$; $\dot{u}_{t+\Delta t} = \dot{u}_t + a_6 \ddot{u}_t + a_7 \ddot{u}_{t+\Delta t}$.

Наиболее трудоемкая операция алгоритма Ньюмарка – матрично-векторное произведение, которое применяется как в основном алгоритме, так и на каждом шаге метода сопряженных градиентов для решения системы $\hat{K} u_{t+\Delta t} = \hat{R}_{t+\Delta t}$. Кроме того, в зависимости от формата хранения матриц жесткости и масс рассматриваются различные подходы к матрично-векторному произведению.

В элементной схеме метода сопряженных градиентов будем выполнять произведения $\hat{K} p = C^T \tilde{K} C p$ следующим образом: находим вектор $\bar{p} = Cp,\ p \in R^N$, вычисляем $\tilde{K}\bar{p}$, выполняем операцию $C^T \bar{p}$. Второй этап легко выполнить параллельно, распределив вычисления между ядрами вычислительной системы. При вычислениях на центральном процессоре (CPU) умножение на матрицу связности на первом и третьем этапах заменяется на косвенную адресацию к требуемым компонентам вектора. Для вычисления $\tilde{K}\bar{p}$ используются блоки $\tilde{K}$ и соответствующие компоненты $\bar{p}$, которые сгруппированы по конечным элементам, что позволяет помещать данные в кэш-память CPU, уменьшая общее время вычисления произведения $\tilde{K}\bar{p}$. В случае вычислений на графическом ускорителе (GPU) косвенная адресация на первом и третьем этапах может быть заменена на оптимизированный вариант матрично-векторного произведения, учитывающий разреженную структуру и значения элементов $C$.

В случае явной сборки матрицы $\hat{K}$ она хранилась в сжатом, упакованном по строкам формате. Исходя из того, что матрица $\hat{K} = \hat{K}_1 + \hat{K}_1^T - diag(\hat{K}_1)$ всегда получается симметричной, уместно хранить только половину матрицы для сокращения затрат памяти, например $\hat{K}_1$ – верхнюю треугольную матрицу, включая главную диагональ. Простейший вариант распараллеливания с помощью OpenMP содержал именно такое представление матрицы. Вычисление матрично-векторного произведения $\hat{K} p$ исходя из того, что хранится только половина матрицы (верхний треугольник, включая диагональ), выполнялось в два потока, в которых вычислялись произведения $\hat{K}_1 p$ и $\hat{K}_1^T p$ соответственно. Произведение $diag\left(\hat{K}_1\right)p$, а также операции сложения и вычитания результирующих векторов выполнялись последовательно.

Результаты вычислительного эксперимента показали, что использование GPU при решении системы уравнений дает ускорение от 40 до 120 раз в зависимости от геометрии конечного элемента и формата схемы хранения эффективной матрицы жесткости и матрицы масс.

**2. Решение задач интерполяции и деформирования сетки**

Для решения задач интерполяции и деформирования сетки применялся подход, связанный с использованием радиальных базисных функций [4], обладающий рядом преимуществ: возможность использовать движения отдельных точек, а не сетки (позволяет учитывать только данные об узлах без учета данных, связанных с сеткой); возможность рассмотрения систем меньшего размера при использовании компактных функций; высокая параллельная эффективность, связанная прежде всего с тем, что решение задачи сводится к решению системы уравнений, поиск которого можно достаточно эффективно распараллелить.

Основной идеей метода является поиск необходимого интерполянта в виде линейной комбинации, образованной радиальными базисными функциями. В этом случае интерполянт в виде линейной комбинации запишется следующим образом:

$$\omega_i(x) = \sum_{j=1}^{n_s} \lambda_j \phi\left(\| x - x_{s_j} \|\right) + q(x), \quad i = \{s, g\}, \quad \omega_i = \{u, pn\},$$

где $\omega_i = \{u, pn\}$ принимает значения перемещений $u$ и давления $pn$ на границе сопряжения; индексы $s$ и $g$ относятся к задаче деформирования и газовой динамики соответственно; коэффициенты $\lambda_j$ и полином $q$ определяются из условия $w_s(x_{s_j}) = W_{s_j}$ и требования $\sum_{j=1}^{n_s} \lambda_j s(x_{s_j}) = 0.$ Тогда система уравнений для неизвестных коэффициентов в матричной форме запишется в виде:

$$\begin{bmatrix} \Phi_{ss} & \mathrm{B}_s \\ \mathrm{B}_s^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} W_s^i \\ 0 \end{bmatrix}, \text{ где } \Phi_{ss} = \phi\left(\| x_{s_i} - x_{s_j} \|\right), \ \mathrm{B}_s = \begin{bmatrix} 1 x_{s_j} y_{s_j} z_{s_j} \end{bmatrix}.$$

Для параметров на газодинамической сетке запишем

$$\begin{bmatrix} \Phi_{gs} & \mathrm{B}_g \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = W_g^i.$$

Кроме того, значения на газодинамической сетке могут быть получены и непосредственно из соотношения

$$W_g^i = \begin{bmatrix} \Phi_{sg} & \mathrm{B}_g \end{bmatrix} \begin{bmatrix} \Phi_{ss} & \mathrm{B}_s \\ \mathrm{B}_s^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} W_s^i \\ 0 \end{bmatrix}.$$

Аналогично для задачи деформирования газодинамической сетки построим интерполирующую функцию в виде линейной комбинации, образованной радиальными базисными функциями. Для внутренних узлов сетки в $\Omega_g$ приращения $\Delta x_{\Omega_g}$ находятся как $\Delta x_j(\Omega_g) = s(x_j(\Omega_g))$.

Таким образом, рассматриваемые выше задачи сводятся к решению систем линейных алгебраических уравнений с матрицами, разреженность и порядок которых зависят от выбора радиальных базисных функций. Такие системы можно эффективно решать предобусловленным методом сопряженных градиентов.

### 3. Структура и функции связывающего приложения

Предлагаемый подход к моделированию задач взаимодействия потока газа и деформируемой конструкции/тела основан на выделении этапа согласования и взаимодействий в отдельную программную подсистему, не зависящую от используемых реализаций расчетов, газодинамических полей и напряженно-деформированного состояния. Таким образом, может быть обеспечено взаимодействие для произвольных комбинаций программ решения задач газовой динамики и механики деформирования. В этом случае сопрягаются различные типы сеток, разные виды аппроксимаций. Точность аппроксимации уравнений сохраняется как в областях расчета газодинамики и механики деформирования, так и на границе взаимодействия.

В рассматриваемом случае система уравнений газовой динамики решается произвольным лагранжево-эйлеровым конечно-объемным методом в библиотеке OpenFOAM (http://www.openfoam.com), а на этапе решения задач деформирования применяется конечно-элементный пакет FEStudio [5]. Важно отметить, что оба пакета написаны в рамках объектно-ориентированного подхода и это существенно облегчает их совместное использование.

Приложение FEStudioFSI реализовано в соответствии с моделью «клиент-сервер» на объектно-ориентированном промежуточном ПО Ice Zeroc (http://www.zeroc.com/ice.html). Клиентами называются активные сущности, запрашивающие определенные сервисы у сервера. Серверами называются пассивные сущности, предоставляющие сервисы в ответ на запросы клиентов.

Клиент использует для вызова Ice-объекта так называемый прокси (proxy). Прокси – это представитель Ice-объекта на стороне клиента, в его локальном адресном пространстве. Код прокси для определенного языка программирования генерируется Slice-компилятором. Прокси инкапсулирует в себе информацию, необходимую для вызова Ice-объекта: физический адрес сервера, идентификатор объекта и, дополнительно, идентификатор фасета. Для работы распределенной модели FEStudioFSI клиент приложения должен содержать информацию о прокси двух серверов OpenFOAM и FEStudio, каждый прокси обладает уникальным именем, которое служит идентификатором при обращении к соответствующему серверу. Удаленный вызов методов, реализованных на стороне сервера, осуществляется посредством созданных прокси. Кроме того, клиент хранит и обновляет векторы перемещения и давления для их последующего использования в расчете и управляет логикой работы всей программы.

Со стороны серверов поведение Ice-объектов реализуется с помощью сервантов. Со стороны серверов реализован ряд методов, необходимых для распределенного решения задачи: расчет перемещений (сервант FEStudio), расчет давления (сервант OpenFOAM), передача полученных решений клиенту, получение новых данных о перемещении и давлении, необходимых для следующего шага решения.

На рис. 1 показана распределенная модель приложения FEStudioFSI. Она представляет собой объединение трех Ice-объектов: серверов OpenFOAM и FEStudio и клиента FEStudioFSI. В целом функционирование распределенной программы выглядит следующим образом. На процессорах запускаются серверные объекты OpenFOAM и FEStudio, в управляющем процессе создается клиентский объект FEStudioFSI, который содержит прокси к соответствующим сервантам. Во время запуска серверных объектов инициализируются переменные, необходимые для расчетов каждой из программ OpenFOAM и FEStudio, а также загружаются расчетные сетки. Далее, после инициализации, как только серверы сигнализируют о готовности, клиент запускает расчет и обмен данными, выполняя соответствующие запросы серверам. Кроме обеспечения взаимодействия двух серверов, клиент в рамках построенной модели занимается интерполяцией сетки. Время продолжительности расчета определяется клиентом и не зависит

от серверов. Описанный механизм связывания объектно-ориентированной модели и инфраструктуры Ice позволяет путем замены объектов получить распределенную объектно-ориентированную программу. Переход к распределенным/параллельным вычислениям осуществляется при минимальной модификации последовательного кода.

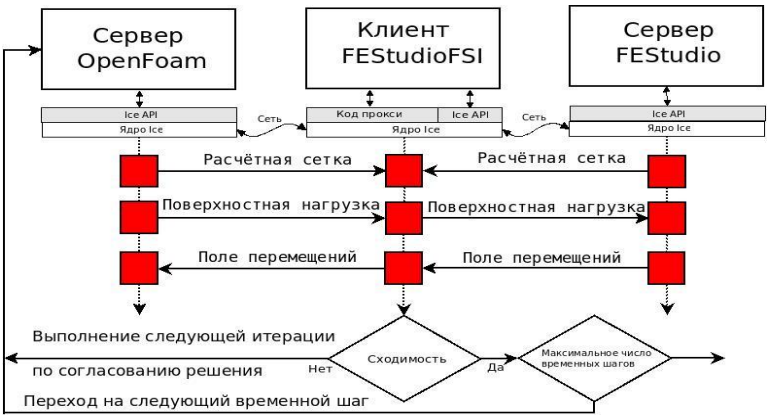


Рис. 1. Распределенная модель решения задачи FSI

Ключевые возможности разрабатываемого промежуточного ПО FEStudioFSI:

1. Возможность выбора алгоритмов для слабого связывания приложений (последовательные алгоритмы, обмен данными происходит на каждом временном шаге один раз) и сильного связывания (в пределах одного шага по времени реализуется итерационная процедура совместного решения уравнений с последовательной передачей нагрузок и перемещений).
2. Поддержка связывания различных независимых приложений посредством интерфейса прикладного уровня.
3. Поддержка различных методов переноса данных для несогласованных сеток, функций интерполяции и перераспределения данных, деформации расчетных сеток.

Разработанные алгоритмы и промежуточное ПО тестировалось на трехмерных задаче, описывающей математическую модель взаимодействия сверхзвукового потока газа и деформируемой панели в ударной трубе с различными условиями связывания и распределенного выполнения на гибридных кластерах. Выполненное сравнение результатов численного моделирования с экспериментальными данными позволяет заключить, что предложенные модели и алгоритмы адекватно описывают происходящие физические процессы как в газе и деформированной панели, так и при их взаимном влиянии.

Работа выполнена при поддержке РФФИ (проект 11-01-00275-а, 12-07-00080-а) и программы Президиума РАН N 18 при поддержке УрО РАН проект 12-П-1-1034.

**Литература**

1. Farhat C., Lessoine M., Le Tallec P. Load and motion transfer algorithms for fluid-structure interaction problems with non-matching discrete interfaces: momentum and energy conservation, optimal discretization and application to aeroelasticity – Computer Methods in Applied Mechanics and Engineering. 1998. V. 157, № 1-2. P. 95–114.
2. Kopysov S.P., Novikov A.K. Parallel element-by-element conjugate gradient method with decreased communications costs // The International Summer School Iterative Methods and Matrix Computations (2-9 June 2002) / Eds. Gene H. Golub, Lev A. Krukier. Rostov-on-Don, Russia, 2002. P. 450–454.

3. Копысов С.П., Новиков А.К. Метод декомпозиции для параллельного адаптивного конечно-элементного метода // Вестник Удмуртского университета. Математика. Механика. Компьютерные науки. 2010. № 3. С.141–154.
4. Buhmann M.D. Radial Basis Function. Theory and implementations. Cambridge University Press, 2003. 259 p.
5. Копысов С.П., Новиков А.К., Пономарёв А.Б., Рычков В.Н., Сагдеева Ю.А. Программная среда расчётных сеточных моделей для параллельных вычислений // Программные продукты и системы. 2008. № 2. С. 87–89.