БИБЛИОТЕКА GPU_SPARSE ДЛЯ ЧИСЛЕННОГО РЕШЕНИЯ ЗАДАЧ МЕХАНИКИ СПЛОШНЫХ СРЕД НА ГИБРИДНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЕ

Д.А. Губайдуллин, А.И. Никифоров, Р.В. Садовников

Введение

Методы подпространств Крылова с предобуславливанием являются наиболее эффективной группой итерационных методов, применяемой для решения разреженных систем линейных алгебраических уравнений (СЛАУ). Такие системы уравнений возникают при численном решении широкого класса задач механики сплошных сред, описываемых дифференциальными уравнениями в частных производных: задачи теплопроводности, задачи гидродинамики многофазных сред, задачи геофизики и др. Для численного решения этих задач требуются значительные вычислительные ресурсы. Повышенные требования к производительности и памяти обусловлены пространственным характером и нестационарностью протекающих процессов, многофазностью сред, нелинейностью моделей сред и другими факторами. Применение параллельных вычислений значительно расширяет возможности исследователей, занимающихся компьютерным моделированием таких сложных физических процессов [1]. Для решения разреженных СЛАУ большой размерности на различных архитектурах центральных процессоров, а также параллельных вычислительных кластерах, построенных на их основе, создано множество различных библиотек программ [2]. Как правило, библиотеки программ используют итерационные методы с предобуславливанием и отличаются форматами представления матриц, типами предобуславливателей, способами обмена сообщениями между процессорами и связаны с архитектурой параллельной вычислительной системы, типами поддерживаемых платформ и др.

В последнее время возрос интерес к параллельным вычислениям на графических процессорных устройствах (ГПУ) NVIDIA и AMD в связи с их высокой производительностью, низким энергопотреблением. При программировании с использованием графических процессоров, последнее рассматривается как вычислительное устройство, способное выполнять большое число одинаковых вычислений параллельно. Для программирования с использованием ГПУ большую популярность приобрела унифицированная архитектура компьютерных вычислений NVIDIA CUDA [3], основанная на расширении языка C. Эта технология дает возможность доступа к набору инструкций ГПУ и управления его памятью при организации параллельных вычислений.

Одна из первых работ, посвященных реализации алгоритмов решения СЛАУ с разреженной матрицей на ГПУ, является работа [4], в которой предложена реализация метода сопряженных градиентов. И хотя метод нельзя назвать универсальным, так как он ограничен использованием только симметричных матриц, тем не менее, в работе была предложена концепция использования итерационных методов на ГПУ. В работах [5,6] рассмотрены различные варианты форматов представления разреженных матриц и способы параллельной реализации операции умножения матрицы на вектор на ГПУ, а также вопросы их оптимизации. В работе [7], на основе работы [4], предложена реализация стабилизированного метода бисопряженных градиентов на ГПУ, который позволяет работать с несимметричными разреженными матрицами. В работе [8] была представлена библиотека gpu_sparse итерационных методов подпространств Крылова с предобуславливанием для решения на ГПУ разреженных СЛАУ с нерегулярной структурой, как симметричных, так и несимметричных.

Особенно перспективным является одновременное использование для расчетов нескольких графических устройств, установленных как на одном вычислительном узле, так и на нескольких вычислительных узлах, т.е построение так называемых гибридных вычислительных систем, на которых вычисления на центральных процессорах совмещаются с вычислениями на графических процессорах. Такие вычислительные системы отличаются довольно высокой производительностью и невысокой стоимостью (собираются из компьютеров и графических видеокарт серийного производства) [9]. Для расчетов на таких системах требуется структурная перестройка алгоритма с явным выделением критических фрагментов, которые можно эффективно реализовать на графических процессорах.

В данной работе представлено расширение библиотеки gpu_sparse итерационных методов подпространств Крылова с предобуславливанием, представленной в работе [8], для использования на нескольких графических устройствах одновременно, а также на гибридных вычислительных системах, состоящих из нескольких вычислительных узлов, каждый из которых снабжен, как минимум, одним графическим устройством. Библиотека итерационных методов предназначена для пользователей, которые хотят избежать трудностей и деталей параллельного программирования на графических процессорах, но которым приходиться иметь дело с большими разреженными СЛАУ и необходимо использовать эффективность параллельной архитектуры графических процессоров. Детали реализации структуры данных отделены от математического алгоритма с помощью шаблонов и объектно-ориентированного программирования на языке C++. Векторы и матрицы записаны в виде шаблонов классов, поэтому они могут использоваться для расчетов, как с одинарной, так и с двойной точностью. Методы записаны в виде функций в формате шаблона, так что они могут использоваться с любой матрицей и вектором, обеспечивая необходимый уровень функциональности.

Представленные в библиотеке методы протестированы на примере численного решения задачи фильтрации. Проведено сравнение производительности вычислений на узле с несколькими ГПУ с расчетами на многопроцессорном кластере.

Организация библиотеки

На рис. 1 представлена схема библиотеки gpu_sparse. На нижнем уровне абстракции представлены основные операции с векторами, как на центральном, так и на графическом процессоре. Эти операции не зависят от структуры представления разреженной матрицы. Операции с векторами реализованы с помощью библиотеки CUDA BLAS [10]. На следующем уровне абстракции представлены классы векторов, матриц и предобуславливателей, которые поддерживают операции умножения матрицы на вектор, как на центральном процессоре, так и на графическом процессоре. Верхний уровень абстракции представлен функциями, реализующими итерационные методы Крылова. Все операции и методы реализованы таким образом, что возможна их дальнейшая модификация в связи, например, с изменением архитектуры графических процессоров.

Рис. 1. Схема библиотеки

Среди методов, которые были реализованы в составе библиотеки, следующие: IR – метод Ричардсона, Cheby – метод Чебышева, CG – метод сопряженных градиентов, CGS – квадратичный метод сопряженных градиентов, BiCGSTAB – стабилизированный метод бисопряженных градиентов, GMRES – обобщенный метод минимальных невязок и TFQMR – метод квазиминимальных невязок без использования транспонирования.

Основные этапы реализации методов на ГПУ следующие: 1) загрузить вектора и матрицу (в одном из форматов) с ЦПУ на ГПУ; 2) произвести операции умножения, сложения, вычитания с векторами, матрицами и предобуславливателями; 3) загрузить результат с ГПУ на ЦПУ.

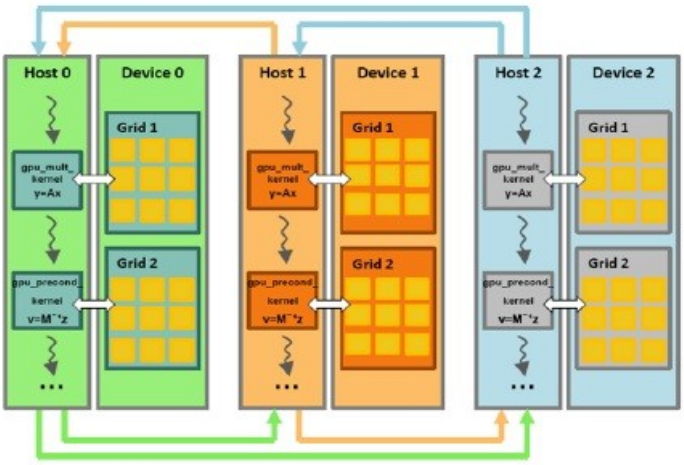


Рис. 2. Схема умножения матрицы на узле гибридной вычислительной системы]

Реализация операции умножения матрицы на вектор на системе с несколькими ГПУ

Самой критичной, относительно времени выполнения, операцией является операция умножения матрицы на вектор. Из-за непредсказуемого шаблона доступа к памяти и сложной структуры данных для представления разреженных матриц построение эффективных алгоритмов для этих операций затруднено. В работе [8] была подробно описана схема реализации операции умножения матрицы на вектор на отдельном графическом процессоре.

В данной работе рассматривается реализация этой операции и схема вычислений на системе с несколькими ГПУ.

Для использования нескольких графических устройств на одном вычислительном узле, требуется одновременно запустить несколько потоков ЦПУ. Поскольку, данные одного графического устройства недоступны напрямую для других графических устройств, необходимо организовать обмен данными между устройствами через память ЦПУ. Для этого данные из памяти графического устройства сначала пересылаются в память ЦПУ и только потом эти данные пересылаются в память другого графического устройства. Обмен данными между памятью ГПУ и ЦПУ осуществляется с помощью функций CUDA Runtime API [11] через PCI-Express x16, который поддерживает скорость передачи данных до 4 ГБ/с. Обмен данными между разными потоками ЦПУ реализуется средствами OpenMP.

Следовательно, для использования нескольких графических устройств на одном вычислительном узле можно предложить следующую схему вычислений. Необходимо разделить область решения задачи на подобласти так, чтобы обеспечить равномерную загрузку потоков ЦПУ. Для этого применяются методы декомпозиции области на подобласти (декомпозиция сетки расчетной области) с помощью методов теории графов, реализованных в специальных библиотеках программ таких, например, как Chaco[12], Metis [13]. После сборки каждым потоком ЦПУ своей части матрицы и соответствующего вектора правой части СЛАУ, каждой подобласти будут соответствовать свои строки матрицы (соответствующие номерам узлов сетки подобласти) и свои компоненты вектора неизвестных и вектора правой части. Перед каждым умножением матрицы на вектор на ГПУ необходимо будет переслать только части этого вектора с соседних ГПУ. Такой обмен данными реализуется через общую память ЦПУ средствами OpenMP, как было описано выше. На рис. 2 представлена схема умножения матрицы на одном узле гибридной вычислительной системы с тремя графическими устройствами. Стрелками указаны обмены данными между потоками ЦПУ и памятью ГПУ.

Операции с векторами, такие как сложение (вычитание) векторов, умножение вектора на скаляр в случае использования нескольких ГПУ остаются без изменений, за исключением вычисления нормы вектора и скалярного произведения векторов. После вычисления значений этих операций локально на каждом ГПУ, требуется глобальная операция суммирования, которая в рамках одного вычислительного узла также совершается средствами OpenMP.

Такая же схема вычислений может быть использована в случае объединения нескольких вычислительных узлов, каждый из которых снабжен, как минимум, одним графическим устройством. В случае объединения нескольких вычислительных узлов с несколькими графическими устройствами обмен данными между графическими устройствами локально в рамках одного вычислительного узла выполняется с помощью функций библиотеки OpenMP, а обмен данными между отдельными узлами с помощью библиотеки MPI.

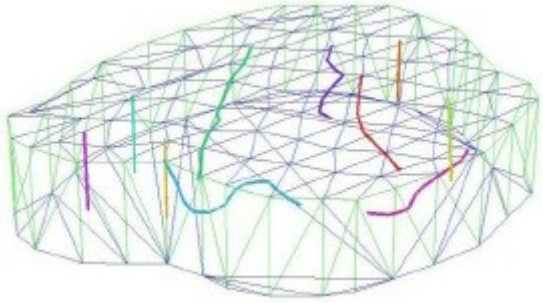


Рис. 3. Схема пласта и расположение скважин

Постановка задачи и метод решения

Рассматривается нестационарная задача фильтрации однофазной жидкости к скважинам со сложной геометрией: вертикальным, наклонным, горизонтальным, искривленным и др. в анизотропной среде с двойной пористостью в 3D области произвольной формы.

Модель фильтрации однофазной жидкости в упругом трещиновато-пористом пласте, основанная на концепции взаимопроникающих континуумов (система трещин и блоков) с учетом обмена жидкостью между ними, была предложена Г.И. Баренблаттом и др. [14]. При учете анизотропии системы трещин и блоков матрицы породы модель описывается следующей системой уравнений:

$$\begin{aligned} \beta_1^* \frac{\partial p_1}{\partial t} &= \nabla\left(\frac{\mathbf{k}_1}{\mu}\nabla p_1\right) + \frac{\alpha}{\mu}(p_2 - p_1), \\ \beta_2^* \frac{\partial p_2}{\partial t} &= \nabla\left(\frac{\mathbf{k}_2}{\mu}\nabla p_2\right) - \frac{\alpha}{\mu}(p_2 - p_1), \end{aligned} \qquad (1)$$

где $(x,y,z) \in D$, $0 < t \le T$, t - время, T - общее время исследований, $\beta_l^* = \beta_{скл} + m_l \beta_{ж}$ - коэффициент упругоемкости пласта, $\beta_{скл}$ - коэффициент сжимаемости скелета, m_l - коэффициент пористости, $\beta_{ж}$ - коэффициент сжимаемости жидкости, p_l - давление жидкости, μ - коэффициент динамической вязкости жидкости, α - параметр перетока жидкости между трещинами и блоками, $\mathbf{k}_l$ - тензор коэффициентов проницаемости. Индекс 1 относится к трещинам, 2 – к блокам матрицы породы. На рис. 3 представлена многосвязная область фильтрации D, внутренние поверхности которой образованы скважинами, представляющими собой цилиндрическую полость определенного радиуса и траектории. Рассматривались

скважины с различной формой траектории ствола: вертикальные, наклонные, горизонтальные и др. На скважинах могут быть заданы граничные условия первого или второго рода:

$$p_1(x,y,z,t) = p_2(x,y,z,t) = p_{wi}(t),\ 0 \le t \le T,\ (x,y,z) \in \partial S_i^p,\ i = 1,2,\ldots,N_w^p, \tag{2}$$

$$\left(\frac{\mathbf{k}_1}{\mu}\nabla p_1, \mathbf{n}_j\right) + \left(\frac{\mathbf{k}_2}{\mu}\nabla p_2, \mathbf{n}_j\right) = q_j(x,y,z,t),\ 0 \le t \le T,\ (x,y,z) \in \partial S_j^q,\ j = 1,2,\ldots,N_w^q, \tag{2a}$$

где $p_{wi}(t)$ - давление на поверхности скважины ∂S_i^p, N_w^p - количество таких скважин, q_j - объемный расход жидкости, приходящийся на единицу поверхности скважины ∂S_j^q, $\mathbf{n}_j$ - вектор внешней нормали, N_w^q - количество таких скважин. Граничные условия на внешней поверхности пласта также могут быть заданы условиями первого или второго рода:

$$p_1(x,y,z,t) = p_2(x,y,z,t) = p_{пл}(x,y,z,t),\ 0 \le t \le T,\ (x,y,z) \in \partial D^p, \tag{3}$$

$$\left(\frac{\mathbf{k}_1}{\mu}\nabla p_1, \mathbf{n}_j\right) + \left(\frac{\mathbf{k}_2}{\mu}\nabla p_2, \mathbf{n}_j\right) = q^*(x,y,z,t),\ 0 \le t \le T,\ (x,y,z) \in \partial D^q, \tag{3a}$$

где $p_{пл}$ - давление жидкости на части внешней поверхности пласта ∂D^p, q^* - объемный расход жидкости, приходящийся на единицу внешней поверхности пласта ∂D^q, $\partial D = \partial D^q \bigcup \partial D^q$ - внешняя поверхность пласта. Начальные условия имеют вид:

$$p_1(x,y,z,0) = p_1(x,y,z),\ p_2(x,y,z,0) = p_2(x,y,z),\ (x,y,z) \in D. \tag{4}$$

Объемный дебит скважины вычисляется по формуле:

$$\int_{\partial S_j^q} q_j(x,y,z,t)ds = Q_j(t),\ j = 1,2,\ldots,N_w^q. \tag{5}$$

Для получения значения забойного давления на скважине используются дополнительные условия: равенство давления в трещинах и блоках породы на поверхности скважины

$$p_1(x,y,z,t) = p_2(x,y,z,t),\ (x,y,z) \in \partial S_j^q,\ j = 1,2,\ldots,N_w^q. \tag{6}$$

и постоянство давления на поверхности скважины. Эти два условия в сочетании с формулой (5) позволяют записать для скважины, на которой задан объемный расход, дополнительное уравнение для определения забойного давления.

Для решения задачи (1)-(6) использовался метод конечных элементов на неструктурированной сетке тетраэдров. Аппроксимация уравнений строилась методом взвешенных невязок в сочетании с методом Галеркина, а для аппроксимации производной по времени использовалась неявная схема [15].

Результаты расчетов

Для тестирования библиотеки gpu_sparse на нескольких графических устройствах использовался вычислительный модуль, состоящий из центрального процессора Intel Core i7 с 6 ГБ оперативной памяти, а также двух видеокарт NVIDIA: Palit GTS 250 (128 ядер с частотой 745 МГц, 16 потоковых мультипроцессоров, 1 ГБ DRAM DDR3 с полосой пропускания 70,4 ГБ/с), MSI GTS 250 (128 ядер с частотой 760 МГц, 16 потоковых мультипроцессоров, 1 ГБ DRAM DDR3 с полосой пропускания 70,4 ГБ/с) и высокопроизводительного вычислительного модуля Tesla C1060 (240 ядер с частотой 1296 МГц, 4 ГБ DRAM DDR3 с полосой пропускания 102 ГБ/с). Результаты расчетов сравниваются с решением этой же задачи с помощью библиотеки подпрограмм Aztec [16] на вычислительном кластере с параллельной архитектурой. Кластер состоит из 4 модулей. Каждый модуль оснащен двумя процессорами AMD Opteron 246 2.0 ГГц и 2 Гб оперативной памяти. Выполнение параллельных задач обеспечивает система управления прохождением задач МВС-1000/7. Подробное описание расчетов на кластере и основные данные приведены в работе [15].

Расчеты проводились на сетке с 761,730 тыс. узлами. Количество тетраэдров сетки составило 4263,344 тыс. элементов. Количество ненулевых элементов матрицы системы 46340,996 тыс. Поскольку каждый узел имеет две степени свободы (в каждом узле два давления жидкости), то количество неизвестных составляет 1523,460 тыс. Для разделения конечно-элементной сетки на подобласти использовалась библиотека подпрограмм Metis, предназначенная для разделения графов. В таблице 1 представлены данные загрузки процессоров.

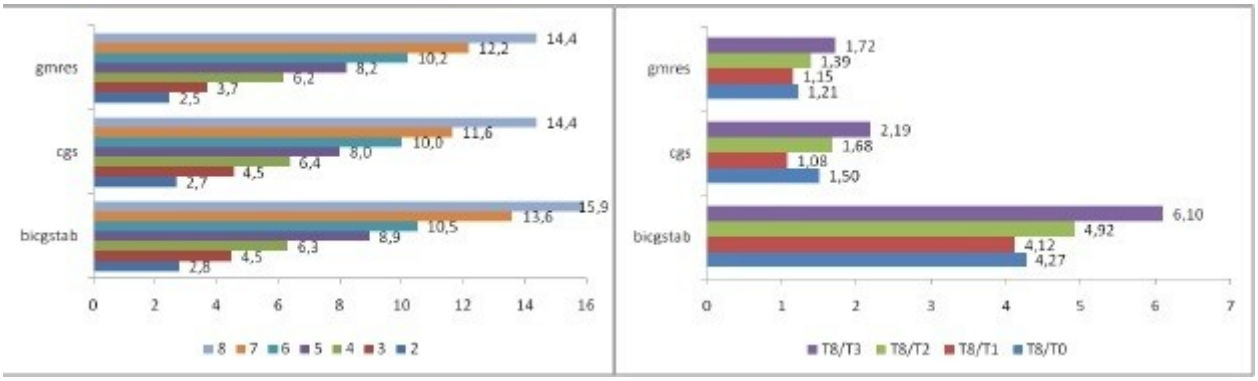
Таблица 1. Загрузка процессоров

Кол-во процессоров	Кол-во элементов (тыс.)	Кол-во неизвестных (тыс.)	кол-во ненулевых элементов матрицы (тыс.)
1	4263,344	1523,460	46340,996
2	2098,826 2205,684	743,479 779,991	22573,737 23767,259

3	2098,826 2205,684 2205,684	502,123 512,741 508,606	15249,685 15591,495 15499,816

Решение задачи на кластере

На рис. 4а представлены результаты расчетов с помощью библиотеки Aztec на вычислительном кластере. Из представленных результатов, видно, что применение параллельных вычислений на 8 процессорах позволяет ускорить вычисления почти в 13-16 раз (в зависимости от метода) по сравнению с последовательными вычислениями на одном процессоре. Ускорение вычислялось делением времени решения задачи на одном процессоре, на время решения на n_p - процессорах.



[рисунок 4. Ускорение вычислений (для полиномиального предобуславливания): а) на кластере; б) на модуле гибридной вычислительной системы]

Решение задачи на модуле гибридной вычислительной системы

На рис. 4б представлены результаты расчетов задачи на вычислительном модуле с тремя графическими устройствами. Ускорение вычислений подсчитывалось делением времени расчетов на восьми процессорах (T8) указанного выше кластера на время решения задачи на одном графическом устройстве GTS 250 (T1), на двух графических устройствах GTS 250 (Palit и MSI) (T2) и на трех графических усройствах 2xGTS 250 (Palit и MSI) и Tesla C1060 (T3). Для сравнения приведены результаты для вычислений на Tesla C1060 (T0). Из представленных результатов видно, максимальное ускорение для расчетов на трех графических устройствах составляет 6,1 раза по сравнению с 8 процессорами Opteron 246, в зависимости от метода.

Таким образом, мы продемонстрировали некоторые эффективные применения библиотеки gpu_sparse итерационных методов подпространств Крылова с предобуславливанием на модуле гибридной вычислительной системы с графическими процессорами NVIDIA с помощью CUDA.

Заключение

Построена библиотека gpu_sparse итерационных методов подпространств Крылова с предобуславливанием для решения разреженных СЛАУ с нерегулярной структурой, как симметричных, так и несимметричных на ГПУ NVIDIA. Библиотека предназначена для использования, как на отдельном, так и на нескольких ГПУ одновременно, а также на гибридных вычислительных системах, на каждом из которых имеется, как минимум, одно графическое устройство. Как видно из представленных расчетов, использование графических процессоров, а также построенных на их основе решений для высокопроизводительных вычислений, позволяет значительно повысить производительность расчетов при низкой себестоимости и низком энергопотреблении.

Работа выполнена в рамках программы Президиума РАН №14 «Интеллектуальные информационные технологии, математическое моделирование, системный анализ и автоматизация».

ЛИТЕРАТУРА:

1. В.В. Воеводин , Вл.В. Воеводин Параллельные вычисления — С.-Пб: БХВ-Петербург, 2002. 608 с.
2. http://www.netlib.org/utk/people/JackDongarra/la-sw.html2
3. NVIDIA Corporation. NVIDIA CUDA Programming Guide. February, 2010. Version 3.0.
4. L. Buatois, G. Cauman, B. Levy "Concurrent Number Cruncher: An Efficient Sparse Linear Solver on GPU". In High Performance Computation Conference (HPCC), Springer Lecture Notes in Computer Sciences, 2008
5. N. Bell, M. Garland "Efficient Sparse Matrix Vector Multiplication on Cuda". NVIDIA Technical Report NVR-2008-004. December 11, 2008.
6. M. Baskaran, R. Bordawekar Optimizing sparse matrix-vector multiplication on GPUs. IBM Tech. Rep. 2009.
7. С.Н. Чадов "Реализация алгоритма решения несимметричных систем линейных уравнений на графических процессорах"// Вычислительные методы и программирование. 2009. Т.10. С. 321 -326.

8. Д.А. Губайдуллин, Р.В. Садовников, А.И. Никифоров "Использование графических процессоров для решения разреженных СЛАУ итерационными методами подпространств Крылова с предобуславливанием на примере задач теории фильтрации" // Сборник трудов международной научной конференции "Параллельные вычислительные технологии (ПаВТ'2010)", г. Уфа, – С. 132–140.
9. С.С. Андреев, А.А. Давыдов, С.А. Дбар, А.Б. Карагичев, А.О. Лацис, Е.А. Плоткина "Макет гибридного суперкомпьютера МВС-экспресс". XVII Всероссийская конференция "Теоретические основы и конструирование численных алгоритмов для решения задач математической физики с приложением к многопроцессорным системам", посвященная памяти К.И. Бабенко. Абрау-Дюрсо, 2008.
10. NVIDIA Corporation. CUBLAS Library. February, 2010. Version 3.0.
11. NVIDIA CUDA. Reference Manual. February, 2010. Version 3.0.
12. Hendrickson B. and Leland R. " The Chaco user's guide - version 1.0" . Technical Report Sand93-2339. Sandia National Laboratories. Albuquerque NM, 87185, August, 1993.
13. METIS – Family of Multilevel Partitioning Algorithms. - http://glaros.dtc.umn.edu/gkhome/views/metis
14. Г.И. Баренблатт, Ю.П. Желтов, И.М. Кочина Об основных представлениях теории фильтрации однородных жидкостей в трещиноватых породах // ПММ. 1960. 123, №3. С. 852-864.
15. Д.А. Губайдуллин, Р.В. Садовников "Применение параллельных алгоритмов для решения задачи фильтрации жидкости в трещиновато-пористом пласте к скважинам со сложной траекторией" // Вычислительные методы и программирование. 2007. Т. 8. № С. 244-251.
16. Aztec. A Massively Parallel Iterative Solver Library for Solving Sparse Linear Systems. – http://www.cs.sandia.gov/CRF/aztec1.html.