УДК 004.021, 519.612.2

СРАВНИТЕЛЬНОЕ ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ РЯДА БИБЛИОТЕК РЕАЛИЗУЮЩИХ АЛГОРИТМЫ РЕШЕНИЯ РАЗРЕЖЕННЫХ СЛАУ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ NVIDIA

М.З. Гатиятуллин, студент
А.В. Юлдашев, старший преподаватель кафедры высокопроизводительных вычислительных технологий и систем
Уфимский государственный авиационный технический университет, Уфа (Россия)

Ключевые слова: графические процессоры; итерационные методы; разреженные матрицы; предобуславливатели; СЛАУ.

Аннотация: Рассмотрены возможности ряда библиотек, реализующих алгоритмы решения разреженных СЛАУ на гибридных вычислительных системах с графическими процессорами. Представлены результаты исследования эффективности библиотек при решении разреженных СЛАУ, возникающих при моделировании процессов фильтрации в пористых средах.

ВВЕДЕНИЕ

Решение систем линейных алгебраических уравнений с разреженной матрицей (разреженных СЛАУ) является вычислительным ядром множества задач математического моделирования. Например, в основе моделирования многофазных фильтрационных потоков в пористых средах лежит система уравнений в частных производных, описывающая распределение давлений и насыщенностей фаз в пласте, которая в результате пространственно-временной дискретизации сводится к СЛАУ с сильно разреженной матрицей. В связи с этим эффективность алгоритмов решения разреженных СЛАУ играет ключевую роль при полномасштабном гидродинамическом моделировании процессов нефтегазодобычи [1].

В последнее время для ускорения вычислений начали активно использоваться гибридные (гетерогенные) вычислительные системы, в которых наряду с процессорами традиционной архитектуры используются массивно-параллельные сопроцессоры, к примеру, графические процессоры (GPU) производства компании NVIDIA с поддержкой программно-аппаратной архитектуры CUDA, обладающие относительно высокой производительностью и энергоэффективностью [2]. Распространение новой вычислительной архитектуры сопровождается адаптацией существующих и появлением новых программных комплексов и библиотек подпрограмм.

За последние несколько лет появился ряд библиотек, предназначенных для решения итерационными методами разреженных СЛАУ на гибридных системах с GPU NVIDIA. Имеются как коммерческие (CULA Sparse, SpeedIT и др.), так и открытые библиотеки (CUSP, ViennaCL и др.). Некоторые разработки доступны для ознакомления, но фактически не развиваются и не поддерживаются, например, OpenNL и CUDA ITSOL. Также есть закрытые библиотеки недоступные для скачивания, например, GAMPACK – специализированная библиотека, ориентированная на решение СЛАУ, возникающих при гидродинамическом моделировании процессов нефтегазободычи. Существуют и отечественные разработки в данной области, например, библиотека gpu_sparse, в которой реализованы итерационные методы решения разреженных СЛАУ для многопроцессорных гибридных систем. Необходимо также упомянуть, что поддержка расчетов на графических процессорах появилась в известных библиотеках PETSc и Trillinos.

В нашей работе проводится исследование эффективности библиотек CULA Sparse [3], CUSP [4], ViennaCL [5] и GAMPACK [6] в решении некоторых разреженных СЛАУ, полученных в ходе гидродинамического моделирования нефтегазовых залежей в пакете NGT BOS [7].

СТРУКТУРА ТЕСТОВЫХ МАТРИЦ

Рассмотрим структуру разреженных матриц, возникающих в ходе моделирования многофазных фильтрационных потоков в пористых средах. В зависимости от способов дискретизации системы дифференциальных уравнений в частных производных, описывающей процессы фильтрации, выделяют ряд численных схем, среди которых наиболее распространенными являются IMPES (неявная по давлению и явная по насыщенностям) и полностью неявная схема Fully Implicit.

В случае схемы IMPES на ячейку приходится 1 неизвестный параметр, что приводит к размерности матрицы СЛАУ на давление равной количеству активных ячеек. Структура получаемой матрицы близка к ленточной, матрица является сильно разреженной: при использовании 7-точечного шаблона дискретизации по пространству в строках имеем не более 7 ненулевых элементов.

В случае более эффективной численной схемы Fully Implicit в зависимости от количества фаз на ячейку приходится до 3 неизвестных параметров, что приводит к увеличению размерности СЛАУ. Структура получаемой матрицы близка к блочно-ленточной с размерностью блока равной количеству фаз. Матрица является несимметричной и сильно разреженной: для трехфазной фильтрации при использовании 7-точечного шаблона дискретизации в строках имеем не более 21 ненулевого элемента.

Учет скважин в реальных моделях нефтегазовых месторождений приводит к наличию дополнительных зависимостей между параметрами в ячейках, поэтому в мат-

рице СЛАУ появляется, так называемая, нерегулярная часть. В связи с этим в матрицах, рассмотренных в данной работе, максимальное число ненулевых элементов в строках достигает 145.

Для эффективного решения разреженных СЛАУ, возникающих при использовании схемы Fully Implicit, предложены специальные методы, на основе двухступенчатого предобуславливания CPR [1,8]. Первым этапом двухступенчатого предобуславливания является решение СЛАУ меньшей размерности на давление (что позволяет провести аналогию со схемой IMPES) в предположении, что насыщенность меняется слабо. На втором этапе полученное решение используется для предобуславливания полной СЛАУ.

Далее будут представлены результаты исследования эффективности решения разреженных СЛАУ, полученных при моделировании процессов фильтрации в пористых средах в рамках схемы Fully Implicit на первом этапе двухступенчатого предобуславливания.

ОБЗОР ФОРМАТОВ ХРАНЕНИЯ РАЗРЕЖЕННЫХ МАТРИЦ

Рассмотрим некоторые популярные форматы хранения (представления) разреженных матриц в памяти, позволяющие значительно сократить требования к объему памяти и исключить избыточные вычисления с использованием нулевых элементов.

1. Покоординатный формат хранения (COO). Одним из наиболее простых форматов хранения является формат COO (Coordinate format). Данный формат предполагает хранение массива ненулевых элементов *Elems*, а также массивов *Row_idx* и *Col_idx*, содержащих индексы строк и столбцов соответствующих ненулевых элементов.

Данный формат подходит для представления разреженных матриц произвольной структуры и поддерживается в большинстве библиотек, ориентированных на использование GPU. Видимым недостатком данного формата является некоторая избыточность данных в массиве индексов строк.

2. Формат хранения со сжатием по строкам (CSR). Наиболее популярным форматом хранения разреженных матриц произвольной структуры является - CSR/SCR (Compressed sparse row storage format). В нем устраняется избыточность формата COO за счет того, что вместо хранения индексов строки для каждого ненулевого элемента в массиве *Row_idx*, в массиве *Row_ptr* хранятся индексы элементов массивов *Elems* и *Col_idx*, соответствующие началу каждой новой строки.

3. Формат хранения ELLPACK (ELL). Идея данного формата, ориентированного на векторные архитектуры, заключается в том, чтобы хранить одинаковое количество элементов из каждой строки, в числе которых могут быть и нулевые, равное максимальному количеству ненулевых элементов на строку во всей матрице. Следовательно, с точки зрения потребления памяти данный формат может быть эффективен для матриц, содержащих близкое количество ненулевых элементов во всех строках. Иначе, возникает необходимость хранения большого числа нулевых элементов. Для хранения используются массивы Elems и Col_idx, которые формируются после определения максимального числа ненулевых элементов на строку.

4. Гибридный формат хранения (HYB). Одним из оптимальных форматов для разреженных матриц произвольной структуры с точки зрения производительности матрично-векторных операций на графических процессорах считается гибридный формат HYB (Hybrid Storage Format) [9], который является комбинацией форматов COO и ELL. Суть гибридного формата заключается в хранении строк с «типичным» количеством элементов на строку в структуре данных ELL, а оставшихся строк в формате COO.

5. Прочие форматы хранения. Существуют и другие менее распространенные форматы хранения разреженных матриц. Одним из простейших форматов является диагональный (DIA), который подходит только для хранения матриц с диагональной структурой. При работе с матрицами, имеющими ярко-выраженную блочную структуру, целесообразно использование блочных форматов, например, блочного формата хранения со сжатием по строкам (BCSR). Недавно был предложен новый формат хранения со сжатием по множеству строк (CMSR), в работе [10] показана его высокая эффективность при выполнении матрично-векторного перемножения на графических процессорах NVIDIA.

Нами был поставлен эксперимент с целью определения влияния формата хранения на производительность алгоритмов решения СЛАУ на GPU на примере библиотеки CUSP, результаты которого приводятся далее.

ИСПОЛЬЗУЕМЫЕ МЕТОДЫ РЕШЕНИЯ СЛАУ И ПРЕДОБУСЛАВЛИВАТЕЛИ

Наиболее эффективными методами решения разреженных СЛАУ большой размерности считаются итерационные методы подпространства Крылова (крыловского типа) [11, 12]. В рассмотренных нами библиотеках преимущественно реализованы следующие методы крыловского типа: метод сопряженных градиентов (CG) и метод минимальных невязок (MINRES), предназначенные для решения СЛАУ с симметричной матрицей, метод бисопряженных градиентов со стабилизацией (BiCGStab) и обобщенных минимальных невязок (GMRES) для решения СЛАУ с несимметричной матрицей, а также их модификации.

В рамках нашего исследования был поставлен эксперимент с целью сравнения производительности на GPU реализаций метода BiCGStab, реализованного в библиотеках CULA Sparse, CUSP и ViennaCL.

Необходимо отметить, что базовой вычислительной операцией в итерационных методах крыловского типа является умножение разреженной матрицы на вектор, которое эффективно распараллеливается. Производительность выполнения данной операции на GPU NVIDIA исследована для широкого круга разреженных матриц с различной структурой (см., к примеру, [9]). Показана возможность существенного (до 10 раз и более) ускорения выполнения данной операции на GPU относительно многоядерных процессоров архитектуры x86, что связано с высокой пропускной способностью памяти, а также массивно-параллельной архитектурой GPU.

Для увеличения скорости сходимости итерационных методов в конечной арифметике широко используется предобуславливание – переход от решения исходной СЛАУ к решению эквивалентной системы с матрицей, обладающей улучшенными спектральными характеристиками.

В рассмотренных нами библиотеках реализованы различные алгоритмы предобуславливания: предобусловливатели Якоби (Jacobi и Block Jacobi), предобуславливание на основе алгебраического многосеточного метода (AMG) и неполного LU-разложения (ILU0 и ILUT), предобуславливатели семейства Sparse approximate inverse (SPAI/AINV) и др.

Необходимо отметить, что при поиске оптимального, специализированного под конкретную задачу, предобусловливателя важно учитывать как снижение количества итераций в процессе сходимости итерационного метода, так и трудоемкость построения матрицы предобусловливателя, т.к. временные затраты на ее построение

могутперевеситьвыигрышотсокращениячислаитераций. С точки зрения эффективности предобуславливания на графических процессорах, важно чтобы алгоритм предобуславливания поддавался распараллеливанию.

Нами был поставлен эксперимент для выявления наиболее эффективных комбинаций итерационного метода и предобуславливателя при решении на GPU ряда тестовых СЛАУ средствами всех рассмотренных библиотек.

РАССМОТРЕННЫЕ БИБЛИОТЕКИ

Основные характеристики библиотек, рассмотренных в ходе исследования, и их версии приведены в Таблице 1.

Таблица 1. Характеристики рассмотренных библиотек.

Название библиотеки	Версия и дата выхода в свет	Лицензия	Методы решения СЛАУ	Предобуславливатели	Форматы хранения матриц
CUSP	0.3.1 03.12	Apache license 2.0	CG, BiCG, BiCGStab, GMRES, CG-M, BiCGStab-M	AINV, AMG based on Smoothed Aggregation (SA-AMG), Diag (Jacobi)	COO, CSR, DIA, ELL, HYB
CULA Sparse	S2 01.12	Коммерческая	CG, BiCG, BiCGStab, GMRES, MINRES	Jacobi, Block Jacobi, Incomplete LU (ILU0), Reordered ILU0	COO, CSC, CSR
ViennaCL	1.2.1 03.12	MIT (X11) license	CG, BiCGStab, GMRES	ILUT (CPU), Jacobi, Row Scaling Экспериментально: AMG, AINV	COO, CSR
GAMPACK	-	-	CG, FGMRES	AMG	HYB

CUSP (CUda SParse). Свободно-распространяемая библиотека для решения разреженных СЛАУ итерационными методами крыловского типа на GPU NVIDIA. Библиотекой поддерживается широкий круг форматов хранения разреженных матриц. Для файлового ввода-вывода используется формат Matrix Market array general. Библиотека базируется на технологиях CUDA и Thrust; версия 0.3.1 работает на системах с установленным пакетом CUDA Toolkit версии не ниже 4.1.

CULA (CUda Linear Algebra) **Sparse**. Проприетарная библиотека линейной алгебры от компании EM Photonics. Существуют две версии данной библиотеки: Dense и Sparse. Как следует из названия, последняя предназначена для решения разреженных СЛАУ. Для чтения матриц из файла используется формат Matrix Market array general, а векторов – Matrix Market vector. Известно, что данная библиотека базируется на ряде других: CUSPARSE и COLAMD. В ходе нашего исследования использовалась доступная для скачивания демо-версия CULA Sparse. Данная версия представляет собой исполняемый файл, на вход которого можно подать матрицу и вектор правой части СЛАУ, а также указать метод и параметры решения. Однако возможности демо-версии несколько ограничены, к примеру, для хранения матриц возможно использование только формата COO. Необходимо отметить, что на момент написания статьи появилась информация о доступности полнофункциональной версии библиотеки для академических целей.

ViennaCL. Свободно-распространяемая библиотека линейной алгебры, реализованная на основе открытого стандарта OpenCL. Благодаря этому библиотека работает как на графических процессорах AMD и NVIDIA, так и на многоядерных центральных процессорах (CPU). Библиотека содержит богатый набор предобуславливателей, но некоторые их них (AMG и AINV) пока реализованы в экспериментальном режиме. Также необходимо отметить, что имеющийся в библиотеке предобуславливатель ILUT реализован только для CPU. Работа с файлами аналогична библиотеке CUSP. Библиотека предусматривает возможность оптимизации под целевую архитектуру посредством проведения ряда тестов с использованием базовых матрично-векторных операций, которая была задействована в ходе нашего исследования. Кроме итерационных методов библиотека содержит прямые методы решения СЛАУ, а также реализации некоторые других алгоритмов, к примеру, быстрого преобразования Фурье. В последней версии библиотеки на момент написания статьи (1.3.1) появились реализации алгоритмов нахождения собственных значений, QR-разложения и др. Кроме того, в последней версии появилась поддержка форматов хранения разреженных матриц ELL и HYB.

Последняя из рассмотренных библиотек – **GAMPACK**, является закрытой разработкой компании Stone Ridge Technology. Данная библиотека ориентирована на решение СЛАУ, возникающих в процессе гидродинамического моделирования нефтегазовых месторождений и поэтому содержит небольшой набор специализированных методов. Библиотека GAMPACK недоступна для самостоятельного ознакомления, однако, разработчики библиотеки провели эксперименты с использованием представленных нами СЛАУ на эквивалентном оборудовании.

Заметим, что все рассмотренные библиотеки являются кросс-платформенными (Windows/Linux) и поддерживают вещественные вычисления с двойной точностью.

ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

Приведенные далее результаты тестирования библиотек CUSP, CULA Sparse и ViennaCL получены на гибридном вычислительном сервере IBM iDataPlex dx360 M3 server (2x Intel Xeon 5670 Six Core, 48 GB, 2x GPU NVIDIA Tesla M2050, ОС RHEL 6.0). Тестирование библиотеки GAMPACK проводилось на гибридной системе с аналогичным графическим процессором (2x Intel Xeon X5560, GPU NVIDIA Tesla M2050, ОС Linux). Все вычисления проводились с двойной точностью с включенным режимом коррекции ошибок памяти ECC, имеющемся на GPU Tesla.

В ходе тестирования использовались следующие параметры. Условие выхода – достижение относительной невязкой заданной величины (□=1e-6). Начальное приближение – нулевой вектор. Ограничение по числу итераций – 5000. Характеристики тестовых матриц приведены в таблице 2.

Таблица 2. Характеристики тестовых матриц.

№	Название	Размерность	Кол-во ненулевых элементов во всей матрице	Среднее кол-во ненулевых элементов в строке	Макс. кол-во ненулевых элементов в строке
1	pp_matrix50	113 279	484 179	4,3	131
2	pp_matrix0	890 323	6 664 145	7,5	145
3	pp_matrix10	2 022 996	11 007 512	5,4	7

Исследование влияния формата хранения матриц на производительность алгоритмов решения СЛАУ. Данное исследование проводилось для матрицы №2 средствами библиотеки CUSP, в которой поддерживаются все описанные выше форматы: COO, CSR, ELL и HYB. Производилось решение СЛАУ с использованием итерационного метода BiCGStab со следующими реализованными в библиотеке CUSP предобуславливателями: AINV (NS Bridson), AMG, Jacobi, а также без предобуславливания (NoPre). На рис. 1 показаны полученные времена в зависимости от форматов хранения. Отметим, что в данном эксперименте итерационный метод сошелся только при использовании предобуславливателя AMG, в остальных случаях итерационный процесс был остановлен по достижению ограничения по числу итераций.

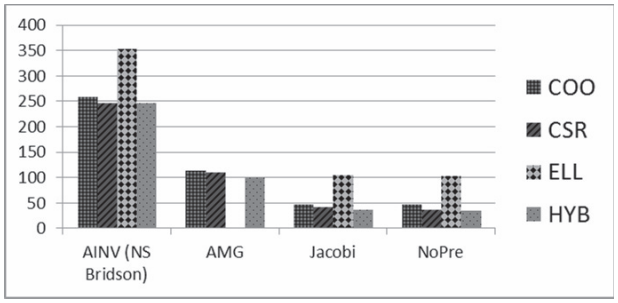

Рис 1. Время решения СЛАУ №2 в зависимости от формата хранения с использованием различных предобуславливателей (CUSP, BiCGStab).

Заметна низкая производительность решения СЛАУ при использовании формата ELL, а в случае использования предобуславливателя AMG программа завершилась из-за нехватки памяти. Это связано с большой разницей между средним и максимальным количеством ненулевых элементов на строку в тестовой матрице №2. Первоначально в связи с этим, ограничения, заложенные по умолчанию в библиотеку CUSP, не позволяли выполнить преобразование матрицы в формат ELL. Данное ограничение было преодолено редактированием исходных кодов библиотеки, но, как видно, вылилось в низкую производительность при использовании формата ELL.

В большинстве случаев наименьшее время решения СЛАУ дало использование формата HYB. Для формата CSR было отмечено замедление в среднем около 8%. При использовании формата COO производительность снизилась на 5–35 % в зависимости от алгоритма решения СЛАУ.

Исследование производительности реализаций итерационных методов. В целях исследования эффективности библиотек было проведено сравнительное тестирование производительности реализаций метода BiCGStab, реализованного в библиотеках CULA Sparse, CUSP и ViennaCL. Тестирование проводилось с одинаковыми параметрами, в частности использовался формат COO, который поддерживается во всех библиотеках, включая демо-версию библиотеки CULA Sparse. В таблице 3 приведено время выполнения 5000 итераций метода BiCGStab.

Таблица 3. Время выполнения 5000 итераций BiCGStab без предобуславливания при решении СЛАУ№2 средствами различных библиотек (формат COO).

Название библиотеки	Количество итераций	Время,с
CULA Sparse	5000	39,98
CUSP	5000	46,06
ViennaCL	5000	74,39

Из таблицы видно, что наиболее оптимизированные реализации метода BiCGStab представлена в библиотеке CULA Sparse и CUSP. Тесты, проведенные на других матрицах, в том числе с использование прочих форматов хранения матриц показали аналогичные результаты.

Поиск эффективных предобуславливателей. Конечной целью данного исследования являлось нахождение наиболее эффективной реализации предобуславливателя для разреженных СЛАУ, возникающих на первом этапе двухступенчатого предобуславливания (в рамках схемы Fully Implicit) при решении задачи численного моделирования фильтрационных потоков в пористых средах.

Таблица 4. Минимальное время решения тестовых СЛАУ средствами различных библиотек.

М-ца	Название библиотеки	Итерац-й метод	Предобуславливатель	Кол-во итер-й	Время предоб-я, с.	Время реш-я, с.	Общее время, с.
1	**CUSP**	**BiCGstab**	**Jacobi**	**128**	**0,001**	**0,156**	**0,157**
	CULA	**GMRES (12)**	**Block Jacobi (3)**	**10**	**0,001**	**0,150**	**0,152**
	ViennaCL	GMRES (12)	AMG (OP Direct)	7	1,009	0,181	1,190
	GAMPACK	**FGMRES**	**AMG**	5	**0,086**	**0,042**	**0,128**
2	CUSP	GMRES (12)	AINV (NS Bridson)	30	58,323	0,345	58,668
	CULA	**GMRES (12)**	**ILU0**	**4**	**1,096**	**1,095**	**2,191**
	ViennaCL	BiCGStab	AMG (RS Classic)	79	11,816	30,003	41,820
	GAMPACK	**FGMRES**	**AMG**	**8**	**0,536**	**0,269**	**0,805**
3	CUSP	GMRES (12)	AINV (NS Bridson)	142	116,000	3,685	119,686
	CULA	**GMRES (12)**	**ILU0**	**6**	**0,955**	**3,576**	**4,531**
	ViennaCL	BiCGStab	AMG (OP Direct)	74	25,631	40,858	66,489
	GAMPACK	**FGMRES**	**AMG**	**11**	**0,885**	**0,459**	**1,344**

Для достижения указанной цели на выбранных матрицах было проведено массовое тестирование с вариацией всевозможных методов решения СЛАУ и предобуславливателей для каждой библиотеки. В ходе тестирования библиотек CUSP и ViennaCL был задействован формат хранения CSR, CULA – COO и GAMPACK – HYB. В таблице для каждой матрицы показаны наилучшие времена решения СЛАУ, которые удалось получить средствами той или иной библиотеки, с указанием соответствующего итерационного метода и предобуславливателя. Жирным шрифтом для каждой матрицы отмечены строки, соответствующие минимальному времени решения СЛАУ, а также временам, не превосходящим минимальное время более чем на порядок.

Видно, что абсолютное лидерство среди рассмотренных библиотек принадлежит библиотеке GAMPACK, реализующей метод FGMRES в связке с предобуславливателем AMG. Необходимо отметить библиотеку CULA Sparse, в которой реализован предобуславливатель ILU0, позволяющий существенно снизить число итераций, требуемое для сходимости при решении СЛАУ с матрицами большой размерности. Но время, затрачиваемое

на построение предобуславливаеля, а также его применение в процессе работы итерационного метода не позволяет добиться минимального времени решения СЛАУ. Что касается библиотеки CUSP, несмотря на относительно высокую производительность реализаций итерационных методов, что было показано в предыдущем эксперименте, отсутствие эффективных для решаемой задачи предобуславливателей, приводит к существенно более низким результатам относительно коммерческих библиотек CULA и GAMPACK, а также свободно-распространяемой библиотеки ViennaCL. Достаточно большое число итераций в процессе решения СЛАУ средствами библиотеки ViennaCL может быть связано с использованием параметров настройки предобуславливателя AMG, заложенных в библиотеку, по умолчанию, в то время как разработчики библиотеки GAMPACK перед тестированием подбирали оптимальные параметры предобуславливателя. Однако достаточно низкая производительность библиотеки ViennaCL, продемонстрированная в предыдущем эксперименте, не позволяет говорить о возможности кардинального снижения времени решения средствами ViennaCL.

ЗАКЛЮЧЕНИЕ

В данной работе были исследованы возможности библиотек, позволяющих проводить решение разреженных СЛАУ на графических процессорах NVIDIA с поддержкой программно-аппаратной архитектуры CUDA. В ходе исследования были рассмотрены вопросы производительности реализаций итерационных методов, эффективности предобуславливателей, а также форматов хранения разреженных матриц применительно к задаче решения СЛАУ, возникающих в ходе гидродинамического моделирования процессов нефтегазодобычи.

В настоящее время авторы продолжают начатые исследования. Основными направлениями исследования являются поиск эффективных предобуславливателей при решении полных СЛАУ в рамках схемы Fully Implicit и исследование возможностей новых библиотек.

СПИСОК ЛИТЕРАТУРЫ

1. Борщук О.С. О модификации двухступенчатого метода предобуславливания при численном решении задачи многофазной фильтрации вязкой сжимаемой жидкости в пористой среде // Вестник УГАТУ. 2009. Т. 12. № 1. С. 146–150.
2. Графический вызов суперкомпьютерам. Адинец А., Воеводин В. Открытые системы. СУБД. 2008. № 4. С. 35–41.
3. Сайт «CULA Sparse». (http://www.culatools.com/sparse/)
4. Сайт «CUSP». (http://code.google.com/p/cusp-library/)
5. Сайт «ViennaCL». (http://viennacl.sourceforge.net/)
6. K.P. Esler, V. Natoli, A. Samardzic GAMPACK (GPU Accelerated Algebraic Multigrid Package) // ECMOR XIII – 13th European Conference on the Mathematics of Oil Recovery.
7. Суперкомпьютерные технологии в науке, образовании и промышленности / Под ред.: акад. В.А. Садовничего, акад. Г.И. Савина, чл.-корр. РАН Вл.В. Воеводина.-М.: Изд-во МГУ, 2009.-232 с., ил.
8. J. R. Wallis, R. P. Kendall, T. E. Little. Constraint residual acceleration of conjugate residual Method // SPE 13536, 1985.
9. N. Bell and M. Garland, Efficient sparse matrix-vector multiplication on CUDA, NVIDIA Technical Report, NVR-2008–004, NVIDIA Corp., 2008.
10. Z. Kozaa, M. Matykaa, S. Szkodaa, L. Miros. Compressed Multiple-Row Storage Format. (http://arxiv.org/pdf/1203.2946v1.pdf)
11. Баландин М.Ю., Шурина Э.П. Методы решения СЛАУ большой размерности: учебное пособие / М. Ю. Баландин, Э. П. Шурина. – НГТУ, 2000. – 69 с.
12. Saad Y. Iterative methods for sparse linear systems. – 2nd edition. – SIAM Society for Industrial & Applied Mathematics, 2003. – 477 p.

A COMPARATIVE STUDY OF EFFICIENCY OF SEVERAL LIBRARIES IMPEMENTING ALGORITHMS FOR SOLVING SPARSE LINEAR SYSTEMS ON NVIDIA GPUS

M.Z. Gatiyatullin, student
A.V. Yuldashev, senior lecturer of the chair
«High Performance Computing technologies and systems»
Ufa State Aviation Technical University, Ufa (Russia)

Keywords: graphics processors; iterative methods; sparse matrices; preconditioners; systems of linear algebraic equations.

Annotation: Capabilities of several libraries implementing algorithms for solving sparse linear systems on hybrid computing systems with graphics processors are considered. Results of experimental study of libraries efficiency in solving sparse linear systems arising from modeling of filtration processes in porous media are presented.