Использование графических процессоров для решения разреженных СЛАУ итерационными методами подпространств Крылова с предобуславливанием на примере задач теории фильтрации

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

В данной работе представлена реализация на современных графических процессорах NVIDIA библиотеки итерационных методов подпространств Крылова с предобуславливанием, на сегодняшний день наиболее общей группы методов, используемых в приложениях для решения больших разреженных систем линейных алгебраических уравнений (СЛАУ). Рассмотренные методы работают для матриц разреженных СЛАУ самого общего вида, с нерегулярной структурой, как симметричных, так и несимметричных. 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.

1. Введение

Наряду с темой высокопроизводительных вычислений и суперкомпьютеров сегодня все шире обсуждается новое направление в ускорении расчетов – использование графических процессорных устройств (ГПУ). Изначально графические процессоры не предназначались для высокопроизводительных вычислений, а разрабатывались для воспроизведения трехмерных изображений в реальном времени. Применение ГПУ для математических расчетов затруднялось не только сложностью архитектуры, но и отсутствием удобного программного интерфейса. Программистам приходилось использовать ГПУ через стандартный графический интерфейс — OpenGL или DirectX, когда данные к видеочипу передавались в виде текстур, а расчетные программы загружались в виде шейдеров.

Начиная с 2006 г., когда ведущие производители процессоров AMD и NVIDIA выпустили программные средства для взаимодействия с графическими процессорами в обход интерфейсов для работы с графикой, появилась возможность для написания программ, выполняемых на графических устройствах. Наибольшую популярность для расчётов на ГПУ приобрела, представленная компанией NVIDIA, унифицированная архитектура компьютерных вычислений CUDA (Compute Unified Device Architecture), которая хорошо подходит для решения широкого круга задач с высоким параллелизмом. Технология NVIDIA CUDA [1], основанная на расширении языка C, даёт возможность организации доступа к набору инструкций ГПУ и управления его памятью при организации параллельных вычислений.

При программировании с использованием ГПУ, последнее рассматривается как вычислительное устройство, способное выполнять большое число одинаковых вычислений параллельно. Это особенно удобно для применения в задачах линейной алгебры в операциях над матрицами и векторами, и к сегодняшнему дню уже разработаны алгоритмы для расчетов на ГПУ, позволяющие ускорить расчеты с плотными матрицами в десятки, а иногда и в сотни раз [2].

В настоящее время рядом авторов [3,4] разрабатываются алгоритмы для расчетов с разреженными матрицами на ГПУ, которые играют важную роль в итерационных методах решения разреженных СЛАУ. Такие системы уравнений возникают при численном решении широкого класса задач математической физики, описываемых дифференциальными уравнениями в частных производных [5]. Как правило, матрицы этих систем имеют большую размерность. Одной из самых критичных, относительно времени выполнения, операций в итерационных методах решения разреженных СЛАУ является операция умножения матрицы на вектор. Из-за непредсказуемого шаблона доступа к памяти и более сложной структуры данных

для представления разреженных матриц построение эффективных алгоритмов для этих операций затруднено. В работах [3,4] рассмотрены различные варианты форматов представления разреженных матриц и способы параллельной реализации операции умножения матрицы на вектор на ГПУ, а также вопросы их оптимизации.

Одна из первых работ, посвященных реализации алгоритмов решения СЛАУ с разреженной матрицей на ГПУ, является работа [6], 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 работе [7], на основе [6], предложена реализация стабилизированного метода бисопряженных градиентов на ГПУ, который позволяет работать с несимметричными разреженными матрицами. Однако, очевидно, что не существует одного метода, подходящего для любого класса задач, поскольку, один и тот же метод может сходиться или не сходиться к решению в зависимости от типа разреженной матрицы и предобуславливателя. Поэтому, необходимо иметь коллекцию итерационных методов, работающих с различными типами матриц и предобуславливателей.

В данной работе представлена библиотека итерационных методов подпространств Крылова с предобуславливанием для решения разреженных СЛАУ с нерегулярной структурой на ГПУ. Представленные в библиотеке методы протестированы на примере численного решения задачи фильтрации методом контрольных объемов. Проведенные тесты показали, что использование ГПУ позволяет значительно ускорить расчеты.

2. Методы подпространств Крылова с предобуславливанием и их реализация на ГПУ

Основное преимущество итерационных методов перед прямыми методами решения разреженных СЛАУ заключается в минимальных требованиях к памяти для хранения матриц, а также в том, что итерационные методы вместо матрично-матричных операций умножения используют матрично-векторные и работают с результирующими векторами. Другое важное преимущество итерационных методов заключается в том, что эти методы хорошо распараллеливаются [8]. Наиболее эффективной группой итерационных методов, применяемой в настоящее время в линейной алгебре для решения разреженных СЛАУ, являются методы подпространств Крылова с предобуславливанием. К этой группе принадлежат следующие методы: CG – метод сопряженных градиентов, CGS – квадратичный метод сопряженных градиентов, BiCGSTAB – стабилизированный метод бисопряженных градиентов, GMRES – обобщенный метод минимальных невязок и др.

Итерационные методы решения разреженных СЛАУ, обычно включают следующие основные операции линейной алгебры: матрично-векторные умножения, умножение матрицы на вектор, линейные комбинации векторов (операции сложения, вычитания, умножения векторов на скаляр), скалярное произведение векторов, вычисление норм векторов и матриц, вычисление предобуславливателей, операции с предобуславливателями. Производительность этих операций может быть значительно повышена применением высокопроизводительных вычислений и массовый параллелизм, которым обладают ГПУ, предоставляет возможность их использования для решения задач линейной алгебры.

2.1. Реализация операции матрично-векторного произведения на ГПУ

Наиболее критичной, с точки зрения времени выполнения, операцией в итерационных методах решения разреженных СЛАУ является операция умножения разреженной матрицы на вектор. Способ вычисления этой операции на любой архитектуре вычислителя, в том числе на ГПУ, зависит от формата представления разреженной матрицы, который, в свою очередь, связан со спецификой рассматриваемой задачи, с типом используемой сетки (количество ненулевых элементов матрицы определяется связями узлов сетки) и способом аппроксимации. Известно довольно много схем хранения разреженных матриц, которые встречаются в

расчетах. Цель каждой из схем заключается в увеличении эффективности, как в использовании памяти, так и в уменьшении количества арифметических операций [5,8].

Для матриц с произвольной нерегулярной структурой ненулевых элементов, одним из самых распространенных форматов хранения разреженной матрицы является формат CSR (CSC) – формат сжатой разреженной строки (столбца). Другой формат MSR (MSC) – модифицированный формат разреженной строки (столбца) отличается от CSR (CSC) тем, что главная диагональ матрицы хранится отдельно от недиагональных членов в каждой строке (столбце). В нашей библиотеке реализованы эти, наиболее общие и часто используемые, форматы представления матриц. Выбор этих форматов для использования мотивирован несколькими факторами: их простотой, общностью представления, широким использованием и довольно легкой параллельной реализацией. В дальнейшем можно распространить этот подход и на другие форматы матриц.

Не останавливаясь на способе реализации операции умножения матрицы в формате CSR(CSC) на вектор на ЦПУ и ГПУ, который подробно изложен в работах [3,4], рассмотрим реализацию этой операции для матрицы в формате MSR (MSC). Структура данных для хранения матрицы в формате MSR (MSC) представляет собой два массива (размером $nnz+1$): вещественный массив, содержащий ненулевые значения матрицы, которые хранятся строка за строкой (столбец за столбцом), и один целочисленный массив, первые n позиций которого содержат значения главной диагонали, а последующие позиции содержат индексы столбцов (строк) элементов и указатели на начало каждой строки (столбца) в матрице.

```
for i=1, n
  start = bindx[i]
   stop = bindx[i+1]-1
    dot = A[i]*x[i]
  for j=start, stop
   dot += A[j]*x[bindx[j]]
  end
  y[i] = dot
end
```

Рис. 1. Псевдокод умножения матрицы в MSR формате на вектор на ЦПУ

```
index = get_thread_index ();
if ( index < n )
  start = bindx[index]
   stop = bindx[index+1]-1
    dot = A[index]*x[index]
  for j=start, stop
    dot += A[j]*x[bindx[j]]
  end
  y[index] = dot
end
```

Рис. 2. Псевдокод умножения матрицы в MSR формате на вектор на ГПУ

Версия последовательного алгоритма умножения разреженной матрицы $\mathbf{A}$ в формате MSR(MSC) на вектор $\mathbf{x}$ на ЦПУ, представлена на рис. 1. Для реализации матрично-векторного произведения $\mathbf{y}=\mathbf{Ax}$ параллельно на ГПУ, заметим, что каждая компонента результирующего вектора $\mathbf{y}$ может быть вычислена независимо, как скалярное произведение i-ой строки матрицы на вектор $\mathbf{x}$. Факт, что внешний цикл может быть выполнен параллельно, используется многими параллельными платформами. Для параллельного выполнения операции умножения разреженной матрицы в MSR (MSC) формате на вектор на ГПУ каждый из двух массивов просто копируется из памяти ЦПУ в память ГПУ, а далее каждая строка матрицы обрабатывается отдельным потоком. Псевдокод для выполнения умножения MSR (MSC) матрицы на ГПУ может быть записан в виде, представленном на рис. 2.

Конечно, представленные форматы матриц обладают тем недостатком, что количество ненулевых элементов и их положение в каждой строке (столбце) может значительно отличаться, что будет сказываться на производительности матрично-векторного произведения, так как потоки для строк (столбцов) с меньшим количеством ненулевых элементов будут выполняться быстрее, по сравнению с потоками для строк (столбцов) с большим количеством. Однако, этот недостаток компенсируется общим временем вычислений, которое уменьшается вследствие выполнения одних и тех же вычислений над разными данными одновременно большим количеством потоков.

```
for i=1, n
  start = bindx[i]
   stop = bindx[i+1]-1
   y[i] = A[i]*x[i]
  for j=start, stop
    y[bindx[j]] += A[j]*x[i]
  end
end
```

Рис. 3. Псевдокод умножения транспонированной матрицы в MSR формате на вектор на ЦПУ

Другая операция, которая часто встречается в итерационных методах решения разреженных СЛАУ это операция умножения транспонированной матрицы на вектор $\mathbf{y} = \mathbf{A}^{\mathrm{T}}\mathbf{x}$. Как видно на рис. 3, в алгоритме последовательной реализации этой операции вместо явного транспонирования матрицы, используется косвенная адресация. Сложность параллельной реализации этой операции на ГПУ, заключается в том, что разные потоки могут одновременно модифицировать один и тот же элемент вектора $\mathbf{y}$ и результаты могут быть непредсказуемыми. Способы реализации этой операции с помощью атомарных функций и разделения массива $\mathbf{y}$ на множество массивов $\mathbf{y}_{\mathrm{thld}}$ без наложения элементов оказались неэффективными. Непосредственное вычисление транспонированной разреженной матрицы также требует значительных затрат времени. Поэтому вопрос с умножением транспонированной матрицы на вектор остается пока открытым, по крайней мере, для тех форматов хранения разреженных матриц, которые рассмотрены в этой статье. Это, конечно, ограничивает возможности применения итерационных методов, которые используют такое произведение.

2.2. Предобуславливание

Скорость сходимости итерационных методов зависит от спектральных характеристик матрицы [5,8]. Для улучшения спектральных свойств исходная матрица умножается на матрицу предобуславливания, которая улучшает спектр, что увеличивает скорость сходимости. Поэтому, помимо самого итерационного метода важной частью вычислительной методики является предобуславливание матрицы. Использование предобуславливателя вносит дополнительные вычисления на этапе его построения и применения на каждой итерации. Однако, и выигрыш в скорости сходимости может быть значительным. Некоторые предобуславливатели практически не требуют никаких вычислительных затрат на этапе построения, а другие, наоборот, (например, неполная факторизация) требуют значительных затрат на вычисления и трудно распараллеливаются.

Сейчас в библиотеке реализован k – шаговый диагональный предобуславливатель Якоби (где $k>0$ - показатель степени), а также предобуславливание полиномами Неймана и полиномами наименьших квадратов [8].

2.3. Реализация операций с векторами на ГПУ

Для операций модификации векторов, вычисления скалярных произведений векторов, CUDA предоставляет реализацию процедур библиотеки BLAS (Basic Linear Algebra Subprograms) для векторов и плотных матриц, которая подробно описана в руководстве по программированию [1]. Поэтому для операций с векторами мы воспользовались CUDA BLAS.

2.4. Схема реализации методов на ГПУ

В реализации на ГПУ итерационных методов Крылова с предобуславливанием мы ограничились рассмотрением только тех методов, которые допускают использование самой матрицы без операции транспонирования, в силу указанных выше сложностей вычислений с транспонированной матрицей. Поэтому, среди методов, которые были реализованы в составе

библиотеки, следующие итерационные методы: IR – метод Ричардсона, Cheby – метод Чебышева, CG – метод сопряженных градиентов, CGS – квадратичный метод сопряженных градиентов, BiCGSTAB – стабилизированный метод бисопряженных градиентов, и GMRES – обобщенный метод минимальных невязок. Все эти методы спроектированы с использованием предобуславливания как полиномами Неймана и полиномами наименьших квадратов, так и k - шагового диагонального предобуславливателя Якоби. Все методы записаны в виде функций в формате шаблона, так что они могут использоваться с любой матрицей и вектором, обеспечивая необходимый уровень функциональности. Детали реализации структуры данных отделены от математического алгоритма с помощью объектно-ориентированного программирования на языке C++. 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.

3. Численные результаты

В данном разделе представлены результаты тестирования описанной выше библиотеки итерационных методов подпространств Крылова с предобуславливанием на примере численного решения задачи фильтрации жидкости к скважинам в двухмерной области произвольной формы. Тестирование проводилось на трех видеокартах семейства NVIDIA GeForce: 9600M GT (32 ядра с частотой 500 МГц, 4 потоковых мультипроцессоров, 512 МБ DRAM DDR3), 9600 GT (64 ядра с частотой 650 МГц, 8 потоковых мультипроцессоров, 512 МБ DRAM DDR3 с полосой пропускания 57,4 ГБ/с), GTS 250 (128 ядер с частотой 745 МГц, 16 потоковых мультипроцессоров, 1 ГБ DRAM DDR3 с полосой пропускания 70,4 ГБ/с). А также на вычислительном модуле NVIDIA Tesla C1060 (240 ядер с частотой 1296 МГц, 4 ГБ DRAM DDR3 с полосой пропускания 102 ГБ/с). Время решения на ГПУ сравнивалось со временем решения задачи на ЦПУ настольного компьютера (Intel Core 2 Duo E6600 2,4 ГГц, 2 ГБ DRAM, 4 ГБ L2, Windows Vista Business 64 bit). Все вычисления на видеокартах проводились с одинарной точностью, поскольку, только вычислительный модуль Tesla поддерживает вычисления с двойной точностью.

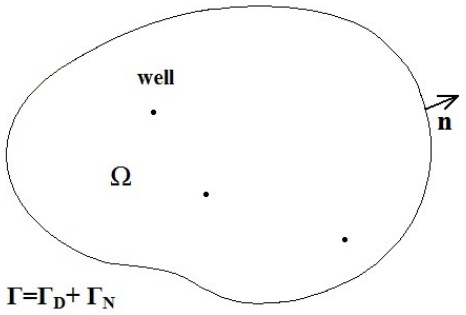

Рис. 4. Область фильтрации

3.1. Постановка задачи

Рассматривается однофазная установившаяся фильтрация несжимаемой жидкости к скважинам в двухмерной области произвольной формы (рис.4). Уравнение фильтрации жидкости в пористой среде имеет вид:

$$\nabla\left(\frac{\mathbf{k}}{\mu}\nabla p\right) = q(\delta(M_w)),\ (x,y)\in\Omega, \tag{1}$$

с граничными условиями

$$p(x,y) = p_R,\ (x,y)\in\Gamma_D, \tag{2}$$

$$\frac{\partial p}{\partial n} = q_R,\ (x,y)\in\Gamma_N, \tag{3}$$

где $p(x,y)$ - давление в жидкости, $\mathbf{k} = \begin{pmatrix} k_{xx} & k_{xy} \\ k_{yx} & k_{yy} \end{pmatrix}$ - тензор проницаемости, μ - вязкость, M_w - точка, в которой расположен центр скважины, Ω - область с границей $\Gamma = \partial\Omega = \Gamma_D \cup \Gamma_N$.

В представленном выше уравнении, на скважине может задаваться как условие забойного давления, так и дебита. Когда на скважине задано условие забойного давления, дебит скважины является неизвестным, и наоборот, когда на скважине задан дебит, неизвестным является забойное давление. Для определения этих значений используется модель точечного источника-стока, которая широко применяется в задачах такого типа [9].

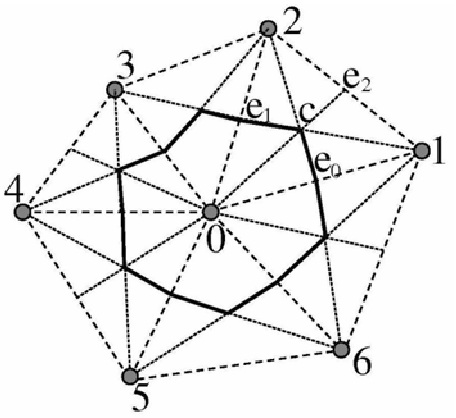

Рис. 5. Построение CVFE сетки

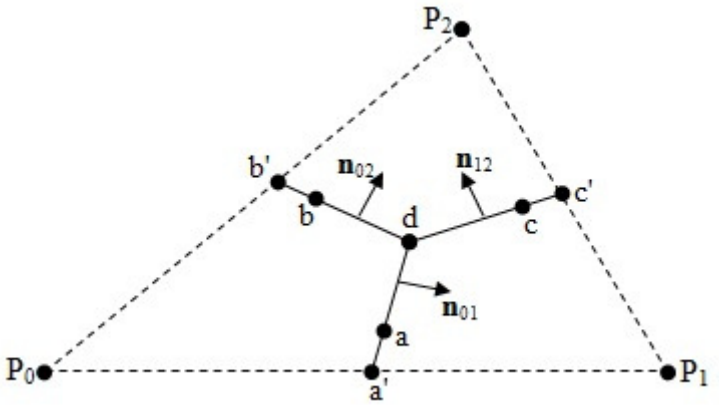

Рис. 6. Положение точек на границах контрольного объема

3.2. Аппроксимация задачи методом контрольных объемов

Для решения поставленной задачи методом контрольных объемов область сначала покрывается нерегулярной сеткой треугольников Делоне. Затем строятся контрольные объемы, в качестве вершин которых выбираются центры тяжести треугольников, лежащие на пересечении медиан. Полученный таким образом контрольный объем представляет, в общем случае, невыпуклый (звездчатый) полигон (рис. 5). Для выполнения условия непрерывности скорости и потенциалов давления на границах контрольного объема должны выполняться условия равенства скоростей и потенциалов в серединах ребер треугольников (рис. 6). Уравнение метода контрольных объемов получается из условия баланса потоков через грани контрольного объема сеточного блока и имеет вид:

$$\sum_{j=1}^{n_c}\left[\sum_{k=1}^{n_{sc}}\left(\sum_{l=1}^{n_{sc,n}}T_{kl}P_l\right)\right]=0, \tag{4}$$

где T_{kl} - среднегармоническое значение проводимости в узлах сетки, P_l - значение потенциала давления в узлах сетки.

Так как размеры сеточного блока значительно больше диаметра скважины, то давление в сеточном блоке (в котором закончена скважина) не может полагаться равным забойному давлению p_{wf}. На самом деле, это давление равно давлению на расстоянии r_0 от сеточного узла для радиального потока [9], т.е.

$$q=2\pi\frac{kh}{\mu}\frac{p_{wf}-p_0}{\ln\left(\frac{r_0}{r_w}\right)}, \tag{5}$$

где $k=\sqrt{k_{xx}k_{yy}}$, h - толщина пласта. Уравнение (5), которое дает соотношение между давлением сеточного блока p_0, забойным давлением p_{wf} и дебитом q называется моделью скважины. Для расчета забойного давления и фиктивного радиуса скважины r_0 с помощью уравнения (5) используются следующие допущения: ось скважины параллельна одной из координатных осей; скважина полностью вскрывает пласт. Тогда, рассматривая баланс

жидкости в окрестности скважины, можно получить следующее выражение для фиктивного радиуса скважины [9]:

$$\ln r_0 = \frac{\sum_u T_u \ln r_u - 2\pi kh}{\sum_u T_u}. \qquad (6)$$

Модель скважины, заданная уравнением (6), действительна только при соблюдении перечисленных выше допущений. Результирующая матрица системы уравнений (4) имеет разреженную структуру, в которой количество ненулевых позиций в каждой строке определяется количеством связей данного узла с другими узлами сетки.

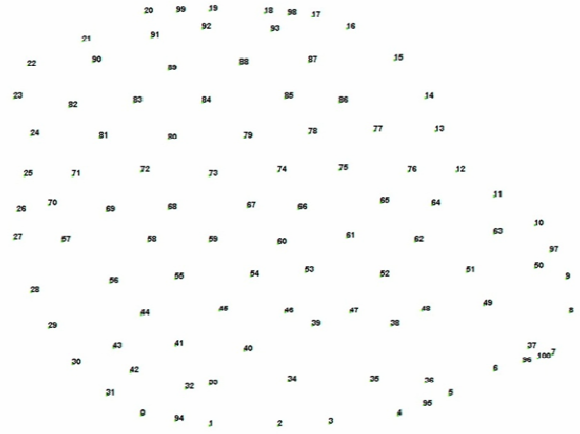

Рис. 7. Граница расчетной области и положение скважин

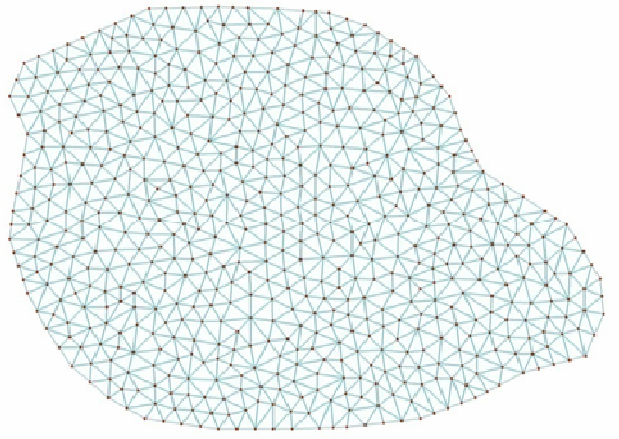

Рис. 8. Триангуляция области

3.3. Результаты расчетов

В примере рассматривалась двумерная область фильтрации, представленная на рис.7, с количеством скважин 64. Размеры области по простиранию 3500 м на 3500 м. . Значения компонент тензора коэффициентов фильтрации принимались постоянными во всей области фильтрации и равными $k_{xx} = 0{,}125$ мкм^2, $k_{xy} = 0{,}023$ мкм^2, $k_{xy} = 0{,}105$ мкм^2, $k_{yy} = 0$, 325 мкм^2. На скважинах задавались объемные дебиты, значения которых варьировались в пределах от 7,6 до 67,5 м^3/сут. Триангуляция области представлена на рис. 8. Расчеты проводились на сетках с различным количеством узлов, представленным в таблице 1.

Таблица 1. Данные для сеток с различным количеством узлов

Количество узлов сетки	Количество		
	ребер	треугольников	ненулевых элементов
178060	532562	354503	1243370
356341	1066784	710444	2490095
888616	2662294	1773679	6213390
1776694	5324793	3548162	12426404

На рис. 9-16 приведены результаты параллельного решения систем уравнений итерационными методами подпространств Крылова с диагональным предобуславливанием на ГПУ и ЦПУ. На всех диаграммах ось ординат – количество точек сетки. На рис.9-12 представлено время (в секундах), за которое невязка решения сходится к заданной точности 10^{-6}, или, в случае отсутствия сходимости, выполняется максимальное количество итераций, равное 2500. На рис.13-16 представлено ускорение вычислений на ГПУ по сравнению с ЦПУ.

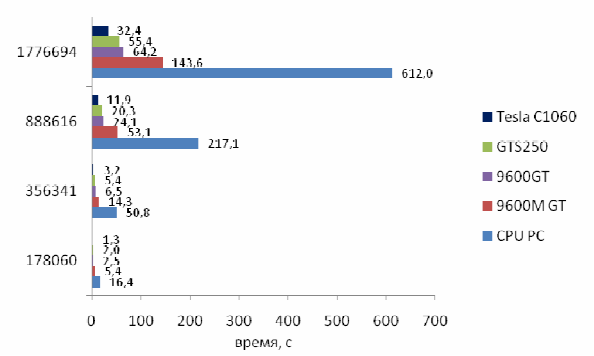

Рис. 9. Метод CG

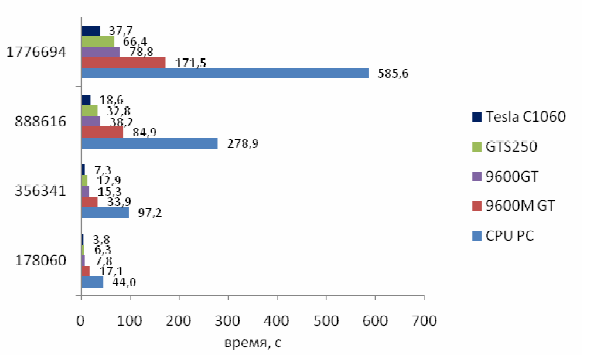

Рис. 10. Метод IR

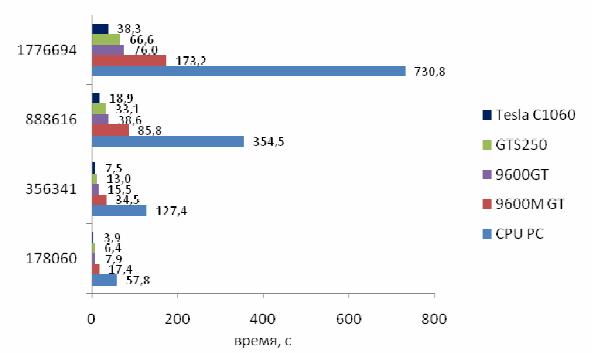

Рис. 11. Метод CHEBY

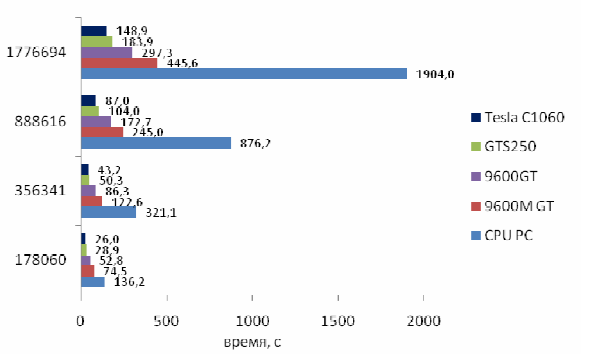

Рис. 12. Метод GMRES

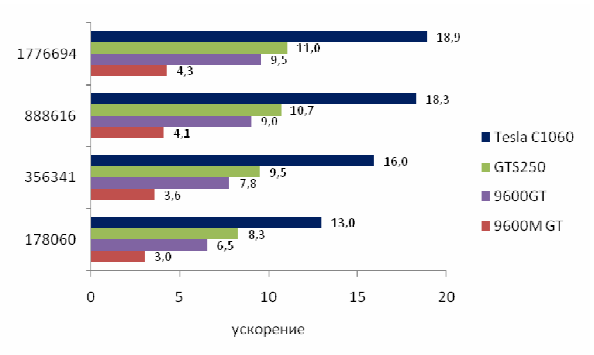

Рис. 13. Ускорение расчетов по методу CG

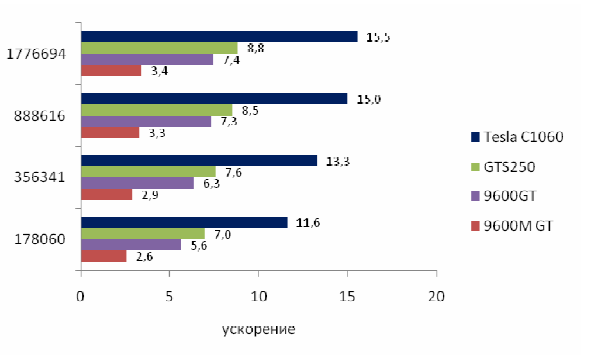

Рис. 14. Ускорение расчетов по методу IR

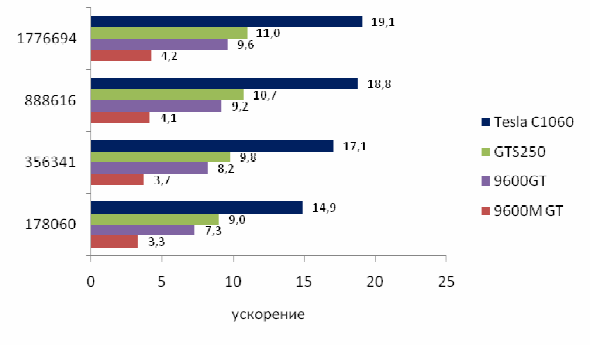

Рис. 15. Ускорение расчетов по методу CHEBY

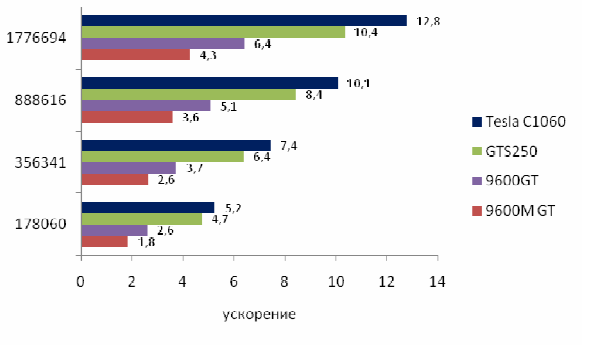

Рис. 16. Ускорение расчетов по методу GMRES

Ускорение подсчитывалось делением времени решения задачи на ЦПУ на время решения на ГПУ.

Из представленных результатов видно, что с увеличением количества неизвестных эффективность расчетов на ГПУ повышается. Ускорение расчетов на стандартных видеокартах NVIDIA GeForce варьируется в пределах 3-12 раз в зависимости от метода, количества точек и типа видеокарты. Так как возможности использования видеокарт NVIDIA GeForce в качестве отдельного вычислителя у нас не было, то часть их ресурсов была занята воспроизведением видеоизображения на экране монитора. Поэтому приведенные результаты не отражают полную производительность методов на видеокартах, но даже в этом случае, позволяют судить о значительном выигрыше в производительности. На вычислительном модуле NVIDIA Tesla C1060 ускорение достигает 19 раз по сравнению с расчетами на ЦПУ.

Таким образом, мы продемонстрировали некоторые эффективные применения итерационных методов подпространств Крылова с предобуславливанием на графических процессорах NVIDIA с помощью CUDA.

4. Заключение

В работе представлена библиотека итерационных методов подпространств Крылова с предобуславливанием для решения на современных графических процессорах NVIDIA разреженных СЛАУ общего вида с нерегулярной структурой, как симметричных, так и несимметричных. 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. Использование графических процессоров, а также построенных на их основе решений для высокопроизводительных вычислений, позволяют значительно повысить производительность расчетов при низкой себестоимости и низком энергопотреблении.

Литература

1. NVIDIA Corporation. NVIDIA CUDA Programming Guide, June 2008. Version 2.0.
2. Volkov V. and Demmel J. W. Benchmarking GPUs to tune dense linear algebra. In Proc. 2008 ACM/IEEE Conference on Supercomputing, November 2008.
3. Bell N., Garland M. Efficient Sparse Matrix Vector Multiplication on Cuda, NVIDIA Technical Report NVR-2008-004, December 11, 2008.
4. Baskaran M., Bordawekar R. Optimising sparse matrix-vector multiplication on GPUs. IBM Tech. Rep. 2009.
5. Barret R. et al. Temlates for the solution of linear systems: building blocks for iterative methods. Philadelphia: SIAM, 1994.
6. Buatois L., Cauman G., Levy B. “Concurrent Number Cruncher: An Efficient Sparse Linear Solver on GPU”. In High Performance Computation Conference (HPCC), Springer Lecture Notes in Computer Sciences, 2008
7. Чадов С.Н. Реализация алгоритма решения несимметричных систем линейных уравнений на графических процессорах. Вычислительные методы и программирование. 2009. Т.10. С. 321 -326.
8. Saad Y. Iterative methods for sparse linear systems. NY: PWS Publish. 1996
9. Verma S., Aziz Kh. A control volume scheme for flexible grids in reservoir simulation. Paper SPE 37999 presented at Reservoir Simulation Symposium, Dallas, Texas, 8-11 June, 1997.