УДК 004.272.23; 519.612.2 doi 10.26089/NumMet.v20r439

ПАРАЛЛЕЛЬНЫЙ ПРЕДОБУСЛОВЛИВАТЕЛЬ НА ОСНОВЕ СТЕПЕННОГО РАЗЛОЖЕНИЯ ОБРАТНОЙ МАТРИЦЫ ДЛЯ РЕШЕНИЯ РАЗРЕЖЕННЫХ ЛИНЕЙНЫХ СИСТЕМ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ

А. В. Юлдашев[1], **Н. В. Репин**[2], **В. В. Спеле**[3]

Рассмотрена применимость метода AIPS, аппроксимирующего обратную матрицу на основе степенного разложения в ряд Неймана, в рамках двухступенчатого предобусловливателя CPR. Предложен ориентированный на архитектуру CUDA параллельный алгоритм решения линейных систем с трехдиагональной матрицей, состоящей из независимых блоков различного размера. Показано, что реализация предложенного алгоритма может более чем в 2 раза превосходить по быстродействию функции решения трехдиагональных систем из библиотеки cuSPARSE. Проведено тестирование метода BiCGStab с предобусловливателем CPR-AIPS на современных GPU, в том числе на гибридной вычислительной системе с 4 GPU NVIDIA Tesla V100, показавшее приемлемую масштабируемость данного предобусловливателя, а также возможность ускорить решение линейных систем, характерных для задачи гидродинамического моделирования нефтегазовых месторождений, по сравнению с CPR-AMG.

Ключевые слова: архитектура CUDA, графические процессоры, итерационные методы, параллельные вычисления, предобусловливатели, разреженные матрицы, трехдиагональные системы.

1. Введение. Одним из актуальных трендов в суперкомпьютерной индустрии является применение наряду с центральными процессорами (CPU) массивно-параллельных процессорных устройств, используемых для ускорения вычислений. Например, в 31-й редакции списка Топ50 мощнейших компьютерных систем России и стран СНГ [1] от 23.09.2019 г. более половины суперкомпьютеров оснащены ускорителями, причем 24 из них графическими процессорами (GPU) NVIDIA Tesla.

В последнее время нами ведется разработка и реализация параллельных алгоритмов, ориентированных на архитектуру GPU, решения разреженных систем линейных алгебраических уравнений (СЛАУ), возникающих при дискретизации дифференциальных уравнений в частных производных, к примеру в задаче моделирования многофазных фильтрационных потоков углеводородов в пористых средах [2]. Сначала [3] был реализован решатель, позволяющий выполнять как на CPU, так и на GPU решение разреженных СЛАУ итерационным методом бисопряженных градиентов со стабилизацией (BiCGStab) [4] с блочной модификацией неполного LU-разложения (BILU(0)) [5] в качестве предобусловливателя. Далее [6] была разработана кластерная версия решателя, позволяющая проводить расчеты на множестве GPU, распределенных по узлам вычислительного кластера. В ней, наряду с BILU(0), был реализован двухступенчатый предобусловливатель Constrained Pressure Residual (CPR) [7], а точнее его популярная версия, часто обозначаемая в литературе аббревиатурой CPR-AMG, в которой на первой ступени используется алгебраический многосеточный метод (AMG) [8], а на второй, как правило, неполное LU-разложение.

В настоящей статье рассматривается предобусловливатель CPR-AIPS, в котором в качестве альтернативы AMG на первой ступени используется метод AIPS (Approximation of Inverse by Power Series), аппроксимирующий обратную матрицу на основе степенного разложения в ряд Неймана. Основной акцент делается на разработке ориентированного на архитектуру GPU параллельного алгоритма решения СЛАУ с трехдиагональной матрицей специального вида (состоящей из множества независимых блоков различного размера), эффективная программная реализация которого и метода AIPS в целом обеспечивает высокое быстродействие предобусловливателя CPR-AIPS на современных GPU.

2.1. Метод аппроксимации обратной матрицы на основе степенного разложения. Большой потенциал распараллеливания имеют предобусловливатели на основе аппроксимации обратной матри-

[1] Уфимский государственный авиационный технический университет, общенаучный факультет, ул. Карла Маркса, д. 12, 450008, Уфа; старший преподаватель, e-mail: art@ugatu.su

[2] Государственный научно-исследовательский институт авиационных систем, ул. Викторенко, д. 7, корп. 2, 125319, Москва; инженер, e-mail: repinn@gosniias.ru

[3] Уфимский государственный авиационный технический университет, общенаучный факультет, ул. Карла Маркса, д. 12, 450008, Уфа; инженер-исследователь, e-mail: spele.vova@ugatu.su

© Научно-исследовательский вычислительный центр МГУ им. М. В. Ломоносова

цы [9], в том числе путем разложения в ряд Неймана [5, 10]:

$$(E-B)^{-1} = E + B + B^2 + B^3 + \dots, \quad \rho(B) < 1,$$

где E — единичная матрица, $\rho(B)$ — спектральный радиус матрицы B. Обратная матрица $A^{-1} = (E-B)^{-1}$ может быть аппроксимирована некоторым количеством членов ряда Неймана.

Наряду с приведенным представлением матрицы A рассматриваются и другие (см., например, [10] и ссылки в работе), в том числе как разности матриц общего вида. Если же представить A в виде

$$A = P + R = P\big(E + P^{-1}R\big) = P\Big(E - (-1)\big(P^{-1}R\big)\Big),$$

то при выполнении условия $\rho\big(P^{-1}R\big) < 1$ с использованием разложения в ряд Неймана можно получить следующую формулу для построения предобусловливателя, аппроксимирующего обратную матрицу при заданном N:

$$A^{-1} \approx M_N^{-1} = \left[E + \sum_{k=1}^{N} (-1)^k \big(P^{-1}R\big)^k\right] P^{-1}. \tag{1}$$

Малое значение параметра N способствует более быстрому вычислению M_N^{-1}, однако может не обеспечить необходимую скорость сходимости итерационного процесса решения СЛАУ ввиду низкой точности аппроксимации обратной матрицы. При увеличении N предобусловливатель становится более эффективным в плане обеспечения скорости сходимости, но высокая трудоемкость его построения может привести к замедлению решения СЛАУ в целом. Хотя в экспериментальной части нашей работы N было положено равным 10, выработка подхода к выбору оптимального значения данного параметра является предметом отдельного исследования.

Что касается структуры матрицы P, в [5] рассматривается использование диагональной либо блочно-диагональной части матрицы A, но существуют и альтернативные варианты. Например, в [11] обратные матрицы аппроксимируются с помощью разложения в ряд Неймана для решения в режиме реального времени СЛАУ малой размерности и в качестве P предложено использование трехдиагональной части A.

В работе [12] формула, аналогичная (1), используется для построения предобусловливателя LSPS, применяемого в задаче гидродинамического моделирования нефтегазовых месторождений. В предобусловливателе LSPS матрица P конструируется так, чтобы она включала значимые элементы матрицы A и оставалась легко обратимой. Наряду с более общим подходом к выбору P метод LSPS предусматривает нестандартное упорядочивание неизвестных в решаемой СЛАУ исходя из физических свойств гидродинамической модели нефтегазового месторождения, что, по всей видимости, приводит к более высокой скорости сходимости ряда Неймана. Авторы метода LSPS предлагают использовать его как одноступенчатый предобусловливатель, а также в качестве первой ступени предобусловливателя CPR и демонстрируют идеальную масштабируемость итерационного метода решения СЛАУ с соответствующими предобусловливателями на 240 процессорах вычислительного кластера.

В рассматриваемой здесь работе мы реализовали и протестировали на GPU предобусловливатель, который строится по формуле (1), называемый методом аппроксимации обратной матрицы на основе степенного разложения (Approximation of Inverse by Power Series, AIPS). Отметим, что организовать применение степенного разложения обратной матрицы в качестве предобусловливателя в рамках итерационных методов решения СЛАУ можно несколькими способами, например явно и неявно.

1. Построить матрицы P и R, явно вычислить P^{-1} и получить M_N^{-1}. При этом применение предобусловливателя в итерационном процессе сведется к одной операции матрично-векторного умножения.

2. Перед началом итерационного процесса ограничиться построением матриц P и R, далее на этапе применения предобусловливателя на каждой итерации вычислять произведение M_N^{-1} на вектор путем выполнения последовательности операций в соответствии с (1).

К минусам первого способа относятся трудоемкость процедуры обращения матрицы P, а также рост потребления памяти в связи с увеличением заполненности матрицы в ходе вычисления M_N^{-1}, поэтому в нашей работе реализован только второй (неявный) способ.

2.2. Применение метода аппроксимации обратной матрицы на основе степенного разложения в рамках двухступенчатого предобусловливателя CPR. Одним из наиболее эффективных предобусловливателей при решении СЛАУ, возникающих в рамках задачи гидродинамического моделирования нефтегазовых месторождений в результате дискретизации уравнений многофазной фильтрации

по времени с использованием полностью неявной разностной схемы, считается специализированный двухступенчатый предобусловливатель CPR и его модификации.

Ранее [6] нами был проведен анализ масштабируемости на двух узлах вычислительного кластера, оснащенных суммарно восемью GPU NVIDIA Tesla M40, метода BiCGStab с предобусловливателем CPR-AMG, в котором на первой ступени использовался классический алгоритм AMG, а на второй — BILU(0). Было показано, что эффективность распараллеливания решателя на 8 GPU составляет не более 50%, причем основным узким местом, препятствующим достижению более высокой масштабируемости на нескольких GPU, является многопроцессорная реализация классического алгоритма AMG.

В данной работе рассматривается предобусловливатель CPR-AIPS, в котором в качестве альтернативы AMG на первой ступени используется описанный выше метод аппроксимации обратной матрицы. Разложение по формуле (1) строится для подматрицы на давление (A_p), которая формируется из исходной матрицы СЛАУ с помощью алгоритма метода CPR в предположении о малом изменении насыщенностей в расчетных ячейках.

При использовании метода AIPS на первой ступени CPR в качестве матрицы P предлагается задействовать трехдиагональную часть подматрицы A_p. Это позволяет в рамках неявного способа применения метода AIPS свести вычисление выражений вида $z = P^{-1}b$ к решению систем $Pz = b$ с трехдиагональной матрицей. Следовательно, для повышения быстродействия предобусловливателя CPR-AIPS на GPU NVIDIA необходим ориентированный на архитектуру CUDA [13] параллельный алгоритм, который рассчитан на многократное решение СЛАУ с неизменной трехдиагональной матрицей, сформированной из подматрицы на давление.

2.3. Параллельный алгоритм решения линейных систем с трехдиагональной матрицей, состоящей из множества независимых блоков различного размера. Ключевым прямым методом решения СЛАУ с трехдиагональной матрицей общего вида является метод прогонки [14], обладающий крайне низким ресурсом параллелизма. В то же время, существует множество подходов к распараллеливанию решения трехдиагональных СЛАУ [15], в том числе параллельные алгоритмы, подходящие для исполнения на GPU. Так, в популярной библиотеке cuSPARSE [16] из состава CUDA Toolkit имеются функции *cusparse<t>gtsv* и *cusparse<t>gtsv2*, основанные на параллельном алгоритме SPIKE [17], а также функции *cusparse<t>gtsv_nopivot* и *cusparse<t>gtsv2_nopivot*, в которых используются алгоритмы циклической редукции [18].

Рассмотрим структуру трехдиагональной части матрицы линейной системы, получаемой в результате применения метода конечных объемов на декартовой сетке размерности $Nx \times Ny \times Nz$ в случае полностью неявной разностной схемы, к примеру при дискретизации уравнений фильтрации углеводородов в пористых средах. Предположим, что в трехдиагональную часть матрицы СЛАУ будут входить коэффициенты, характеризующие связь соседних ячеек по оси Oz. Тогда трехдиагональная часть матрицы СЛАУ будет состоять из $Nx*Ny$ независимых квадратных блоков $Nz \times Nz$.

Такая особенность структуры трехдиагональной части матрицы СЛАУ позволяет вместо решения одной полноразмерной трехдиагональной СЛАУ проводить параллельное решение $Nx*Ny$ трехдиагональных СЛАУ меньшей размерности, что обеспечивает существенный ресурс параллелизма. К примеру, при размерностях сеточной модели $Nx = Ny = Nz = 100$ получаем 10 000 блоков, каждый из которых может независимо обрабатываться с помощью метода прогонки. Рис. 1 иллюстрирует эту идею на примере сетки с размерностями $Nx = Ny = Nz = 2$.

В терминах архитектуры CUDA для параллельной обработки можно задействовать $Nx * Ny$ нитей, каждая из которых будет решать с помощью метода прогонки свою независимую часть трехдиагональной СЛАУ, размещенной в глобальной памяти GPU в виде трех массивов l, d и u, хранящих элементы нижней, главной и верхней диагонали соответственно. В "лобовой" реализации параллельного алгоритма решения множества независимых трехдиагональных СЛАУ нити в рамках реализации концепции SIMT (Single Instruction Multiple Threads) будут одновременно обращаться к элементам, расположенным в памяти с шагом Nz (рис. 2), что при $Nz > 1$ является неэффективным шаблоном доступа к глобальной памяти.

В целях ускорения доступа к глобальной памяти в литературе, например в [19], предлагается выполнить предварительную перестановку строк трехдиагональной матрицы, обеспечивающую объединение запросов к глобальной памяти GPU от синхронно выполняющихся нитей, сгруппированных в один варп (от англ. warp) [13]. Для этого элементы каждой из диагоналей, расположенные с шагом Nz, переносятся в отдельный блок и располагаются поочередно друг за другом. В результате (рис. 3) нити обращаются к данным, расположенным в глобальной памяти в рамках более компактного сегмента, что обеспечивает более высокую скорость выборки данных из памяти.

На практике трехдиагональная часть матрицы СЛАУ может состоять из независимых блоков раз-

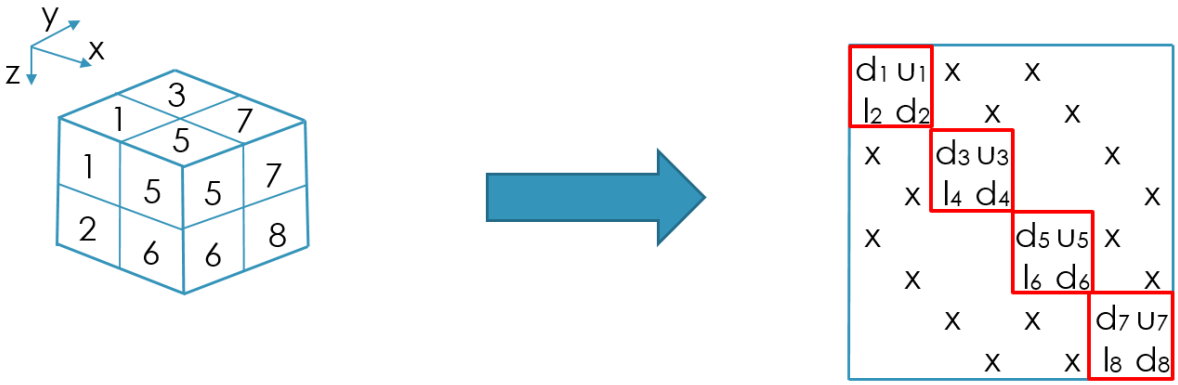

Рис. 1. Пример регулярной блочности в трехдиагональной части матрицы линейной системы

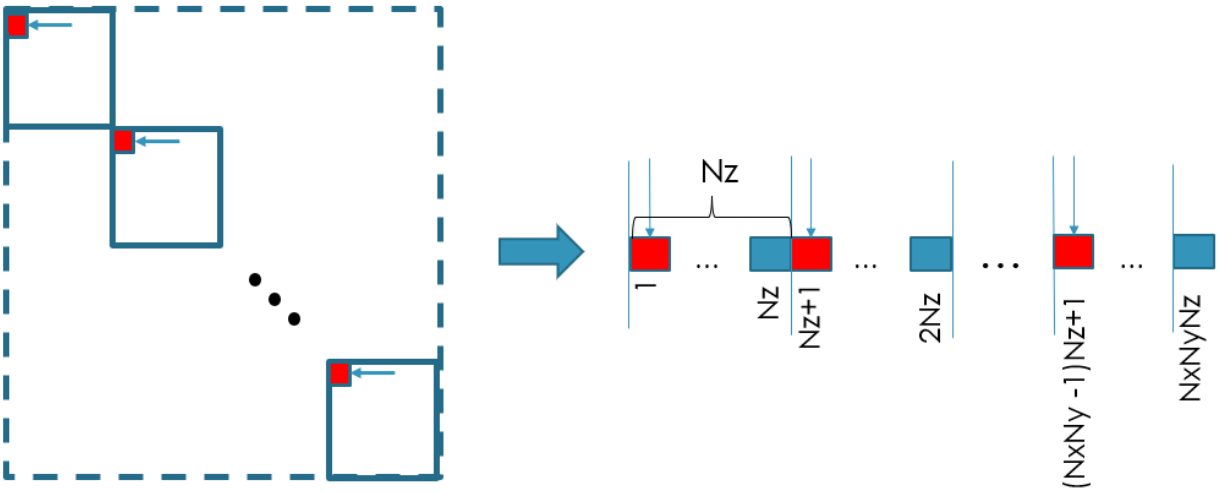

Рис. 2. Неэффективный шаблон доступа к глобальной памяти на примере главной диагонали

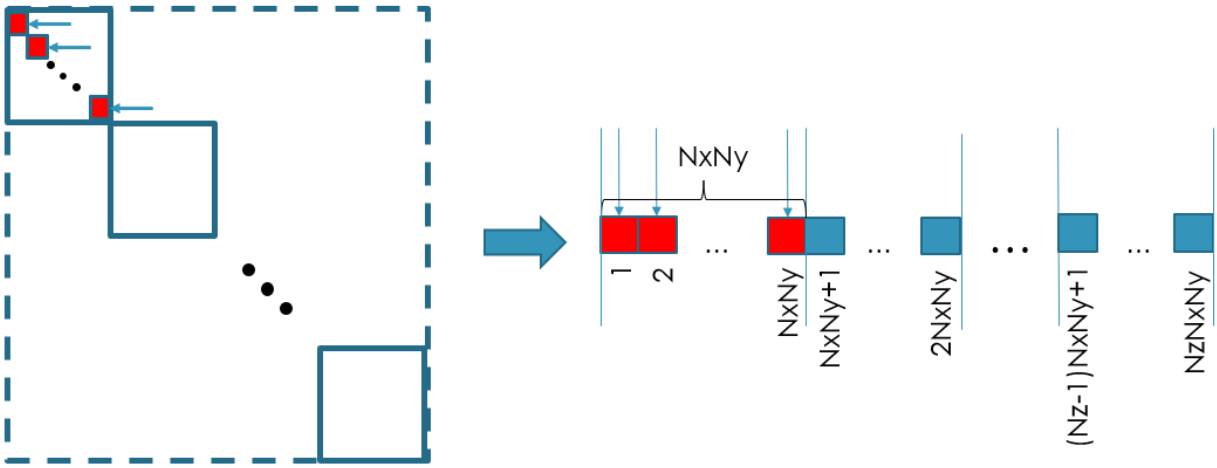

Рис. 3. Эффективный шаблон доступа к глобальной памяти на примере главной диагонали

личного размера. К примеру, в сеточных моделях реальных нефтегазовых месторождений значительное количество ячеек может попасть в разряд неактивных, которые исключаются из расчетов. На рис. 4 проиллюстрирована ситуация, когда исключение из расчета двух ячеек приводит к появлению в трехдиагональной части матрицы СЛАУ наряду с двумя блоками 2×2 двух блоков 1×1.

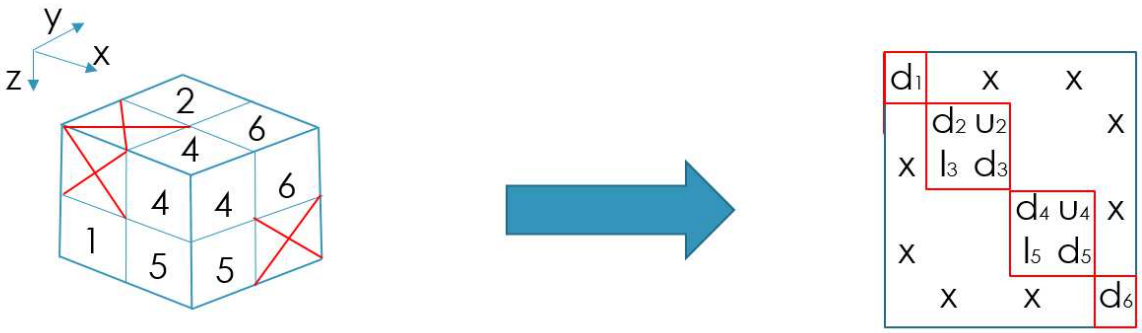

Рис. 4. Пример нерегулярной блочности в трехдиагональной части матрицы линейной системы

Таблица 1

Характеристики тестовых матриц СЛАУ

Название матрицы	Размерность матрицы	Количество ненулевых элементов	Среднее количество ненулевых элементов в строке
imsh	1 500 000	55 815 624	37,210
immn	2 304 102	42 859 314	18,601
krrv	4 320 921	85 471 137	19,781
mmnt	5 637 747	109 595 799	19,440
fdrv	6 610 263	118 221 633	17,885
kmms	8 630 895	167 332 329	19,388
lkms	13 665 705	254 102 823	18,594

Таблица 2

Характеристики тестовых трехдиагональных матриц

Название матрицы	Размерность матрицы	Минимальный размер блока	Максимальный размер блока	Количество независимых блоков
imsh3	500 000	10	10	50 000
immn3	768 034	1	34	221 402
krrv3	1 440 307	1	39	233 866
mmnt3	1 879 429	1	45	281 352
fdrv3	2 203 421	1	55	633 202
kmms3	2 876 965	1	121	389 199
lkms3	4 555 235	1	27	1 167 013

В табл. 1 содержатся характеристики тестовых матриц, полученных при решении уравнений трехфазной фильтрации, дискретизированных по времени с помощью полностью неявной разностной схемы, которые будут задействованы далее в экспериментальной части. В табл. 2 приведены характеристики

трехдиагональных матриц, сформированных из подматриц на давление, которые в свою очередь получены в результате применения метода CPR при решении СЛАУ с матрицами из табл. 1.

Из табл. 2 можно, с одной стороны, сделать вывод о наличии в трехдиагональных матрицах большого количества блоков, которые могут обрабатываться независимо, обеспечивая существенный ресурс параллелизма, а с другой стороны, о значительном разбросе в размерах блоков, составляющих трехдиагональные матрицы. Это говорит о необходимости балансировки нагрузки между параллельно работающими нитями для эффективной загрузки GPU.

Чтобы сбалансировать вычисления на GPU, предлагается провести предварительную сортировку блоков матрицы согласно их размерности, затем распределить блоки одинакового размера между нитями с последовательно идущими индексами. Это обеспечивает возможность в достаточной степени сбалансированного выполнения независимых друг от друга прогонок в каждом конкретном варпе и блоке нитей.

В результате мы приходим к следующему алгоритму решения линейных систем с трехдиагональной матрицей, имеющей нерегулярную блочную структуру.

Алгоритм 1.

1. *Этап анализа структуры матрицы.*

   1.1. Провести анализ структуры трехдиагональной матрицы для выявления количества независимых блоков, их размеров и местоположения.

2. *Этап подготовки матрицы.*

   2.1. Отсортировать блоки матрицы согласно их размерности и выделить группы блоков одинакового размера в целях балансировки нагрузки на этапе решения.

   2.2. Провести перестановку строк матрицы внутри каждой группы блоков одинакового размера для более производительного доступа к памяти на этапе решения.

3. *Этап решения трехдиагональной линейной системы.*

   3.1. Провести перестановку элементов вектора правой части согласно перестановкам строк матрицы, проведенным на этапе 2.

   3.2. Параллельно применить метод прогонки для каждого блока матрицы.

   3.3. Провести обратную перестановку элементов вектора решения.

Отметим, что при решении СЛАУ итерационным методом BiCGStab с предобусловливателем CPR-AIPS этапы 1 и 2 предложенного алгоритма достаточно выполнить только один раз до начала итерационного процесса (в последующих вызовах решателя СЛАУ при отсутствии изменений структуры блочности в трехдиагональной матрице шаг 1 можно вообще исключить), а шаги 3.1–3.3 необходимо выполнять многократно для поочередного решения трехдиагональных СЛАУ с различными правыми частями в рамках применения метода AIPS по формуле (1). Кроме того, учитывая то, что в рамках применения предобусловливателя AIPS этап 3 многократно повторяется с неизменной трехдиагональной матрицей, целесообразно на шаге 3.2 применять вместо классического повторный вариант алгоритма прогонки, в котором переиспользуются однократно предвычисленные коэффициенты [20].

3. Практическая часть. В этом разделе приведены результаты тестирования рассмотренных в теоретической части методов и алгоритмов на двух вычислительных платформах, оснащенных различными серверными GPU NVIDIA Tesla:

— на одном GPU P100 исследовано быстродействие предложенного параллельного алгоритма решения линейных систем с трехдиагональной матрицей, имеющей нерегулярную блочную структуру, в сравнении с алгоритмами решения трехдиагональных систем общего вида, реализованных в библиотеке cuSPARSE;

— на одном GPU P100 и V100 проведена сравнительная оценка эффективности предобусловливателей CPR-AMG и CPR-AIPS;

— на 4 GPU V100 исследована масштабируемость процедуры решения разреженных СЛАУ итерационным методом BiCGStab с предобусловливателем CPR-AIPS.

Для реализации первого эксперимента были задействованы трехдиагональные матрицы, характеристики которых приведены в табл. 2, а для второго и третьего экспериментов — разреженные матрицы, описанные в табл. 1.

Расчеты с использованием GPU P100 были проведены на одном из узлов вычислительного кластера УГАТУ [21] с 2 x CPU Intel Gold 6126 и 2 x GPU NVIDIA Tesla P100, работающего под управлением CentOS 6. Для проведения расчетов на GPU V100 была задействована облачная платформа Amazon AWS p3.8xlarge со следующими характеристиками: ОС RHEL 7, 32 x vCPU Intel Xeon E5-2686 v4 и 4 x GPU NVIDIA Tesla V100 SXM2. На обеих вычислительных платформах были установлены идентичные средства параллельного программирования: CUDA Toolkit 9.1, библиотека AmgX v.2.0.0.130-opensource, OpenMPI v.1.8.8.

Параметры тестирования: условие остановки итерационного процесса — достижение относительной невязкой величины $\varepsilon = 10^{-4}$; начальное приближение — нулевой вектор; расчеты проводились с двойной точностью.

Таблица 3

Время решения тестовых трехдиагональных СЛАУ с использованием различных алгоритмов

Алгоритм	Матрица	imsh3	immn3	krrv3	mmnt3	fdrv3	kmms3	lkms3
gtsv2	Время решения, мс	0,323	0,469	0,753	0,976	1,209	1,459	2,207
gtsv2_nopivot		0,386	0,578	1,055	1,374	1,586	2,046	3,228
Алгоритм 1		0,119	0,212	0,409	0,530	0,568	0,824	1,152
	Время подготовки, мс	0,130	0,190	0,400	0,550	0,710	0,960	1,760

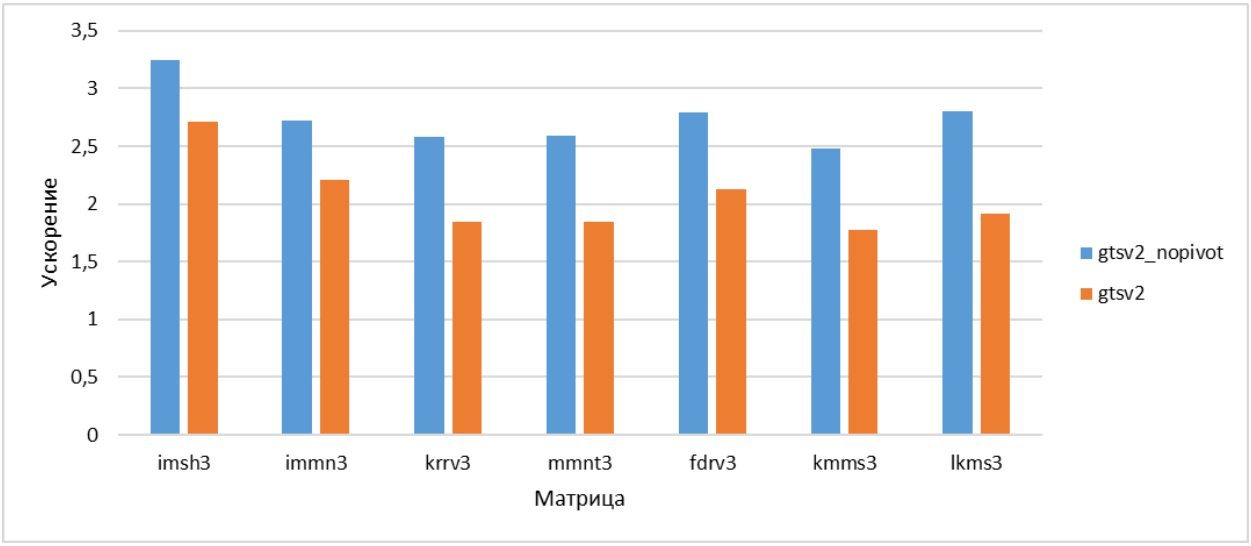

Рис. 5. Ускорение решения трехдиагональных систем с использованием предложенного алгоритма (без учета вспомогательных этапов) относительно функций *cusparseDgtsv2_nopivot* и *cusparseDgtsv2*

3.1. Исследование быстродействия параллельного алгоритма решения линейных систем с трехдиагональной матрицей. В табл. 3 приведены осредненные времена решения СЛАУ с тестовыми матрицами из табл. 2, полученные в ходе многократного решения трехдиагональных линейных систем в рамках итерационного процесса метода BiCGStab с предобусловливателем CPR-AIPS. Решение СЛАУ производилось средствами функций *cusparseDgtsv2* и *cusparseDgtsv2_nopivot* из библиотеки cuSPARSE, а также реализации параллельного алгоритма, описанного в разделе 2.3.

Из табл. 3 следует, что среди рассмотренных функций из библиотеки cuSPARSE наибольшее быстродействие демонстрирует *cusparseDgtsv2*. В то же время, решение СЛАУ с использованием предложенного нами алгоритма без учета временны́х затрат на выполнение вспомогательных этапов производится в среднем быстрее в 2,7 и 2,1 раза относительно *cusparseDgtsv2_nopivot* и *cusparseDgtsv2* соответственно. Детализацию полученных ускорений по матрицам иллюстрирует рис. 5. Наибольшее ускорение достигается на матрице imsh3, состоящей из 50 000 независимых блоков одинакового размера, что обеспечивает

идеальную балансировку нагрузки между нитями при выполнении шага 3.2 предложенного алгоритма.

Несмотря на то что с учетом этапа подготовки матрицы на большинстве тестовых СЛАУ наш алгоритм отрабатывает медленнее, чем *cusparseDgtsv2*, его преимущество проявляется при необходимости выполнения процедуры решения СЛАУ с неизменной трехдиагональной матрицей хотя бы два раза. А с учетом того, что в рамках итерационного процесса метода BiCGStab с предобусловливателем CPR-AIPS процедура решения может повторяться десятки и даже сотни раз, время, затрачиваемое на подготовку, становится относительно мало́.

Таблица 4

BiCGStab с предобусловливателями CPR-AMG и CPR-AIPS на P100

Предобусловливатель	Матрица	imsh	immn	krrv	mmnt	fdrv	kmms	lkms
CPR-AMG	Время решения, с	0,135	0,180	0,285	0,360	0,314	0,678	0,885
	Число итераций	1	2,5	2	2	1	3,5	2,5
CPR-AIPS	Время решения, с	0,070	0,120	0,189	0,205	0,182	0,738	1,409
	Число итераций	1	2,5	2	1,5	1	5	6,5

Таблица 5

BiCGStab с предобусловливателями CPR-AMG и CPR-AIPS на V100

Предобусловливатель	Матрица	imsh	immn	krrv	mmnt	fdrv	kmms	lkms
CPR-AMG	Время решения, с	0,113	0,144	0,202	0,252	0,232	0,437	0,568
	Число итераций	1	2,5	2	2	1	3,5	2,5
CPR-AIPS	Время решения, с	0,043	0,075	0,102	0,108	0,100	0,396	0,772
	Число итераций	1	2,5	2	1,5	1	5	6,5

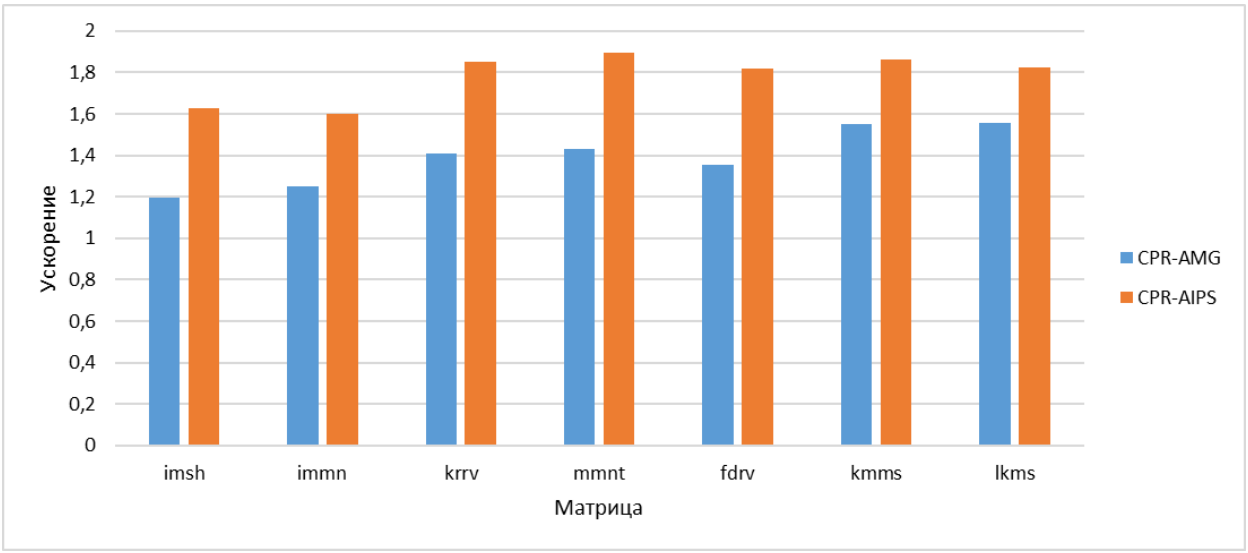

Рис. 6. Ускорение решения СЛАУ методом BiCGStab с предобусловливателями CPR-AMG и CPR-AIPS на V100 относительно P100

3.2. Сравнительная оценка эффективности предобусловливателей CPR-AMG и CPR-AIPS. В табл. 4 и 5 представлены результаты работы на GPU NVIDIA Tesla P100 и V100 итерационного метода BiCGStab с предобусловливателями CPR-AMG и CPR-AIPS.

Проведенные эксперименты показали, что реализованный в нашей работе предобусловливатель CPR-AIPS не уступает по обеспечению скорости сходимости итерационного процесса предобусловливателю CPR-AMG на большинстве тестовых СЛАУ. Ухудшение скорости сходимости наблюдается только на двух матрицах: kmms и lkms.

CPR-AIPS позволяет ускорить относительно CPR-AMG решение 5 из 7 тестовых СЛАУ при расчете на P100 и 6 из 7 — на V100. Причем в среднем время решения СЛАУ при помощи CPR-AIPS снижается относительно CPR-AMG в 1,4 раза на P100 и в 1,9 раз на V100.

Рисунок 6 иллюстрирует повышение производительности решения СЛАУ за счет перехода с P100 на V100. Видно, что расчеты в большей степени ускоряются при использовании CPR-AIPS (в среднем в 1,8 раза), нежели CPR-AMG (в среднем в 1,4 раза). Анализ профилей выполнения программ показал, что причина относительно низкого ускорения CPR-AMG на V100 относительно P100 кроется в процедуре построения иерархии уровней AMG, которая занимает более 50% от общего времени решения СЛАУ и практически не ускоряется при переходе с P100 на V100.

Таким образом, реализованный предобусловливатель CPR-AIPS за счет более легковесной относительно CPR-AMG первой ступени демонстрирует высокое быстродействие на современном GPU NVIDIA Tesla на базе архитектуры Volta и позволяет снизить относительно CPR-AMG время решения подавляющего большинства тестовых СЛАУ.

Таблица 6

Экспресс-анализ влияния $\rho(P^{-1}R)$ на скорость сходимости метода BiCGStab с предобусловливателем CPR-AIPS относительно CPR-AMG

Матрица	imsh	immn	krrv	mmnt	fdrv	kmms	lkms
$\rho(P^{-1}R)$	0,247	0,92	0,976	0,988	0,788	0,987	0,988
Отношение числа итераций (CPR-AMG/CPR-AIPS)	1	1	1	1,3	1	0,7	0,4

3.3. Экспресс-анализ влияния $\rho(P^{-1}R)$ на скорость сходимости метода BiCGStab с предобусловливателем CPR-AIPS относительно CPR-AMG. В целях выработки рекомендаций по эффективному применению предобусловливателя AIPS в качестве первой ступени предобусловливателя CPR рассмотрена взаимосвязь между спектральным радиусом матрицы $P^{-1}R$, формально используемой для формирования предобусловливателя по формуле (1), и скоростью сходимости метода BiCGStab с предобусловливателем CPR-AIPS. Оценка значения $\rho(P^{-1}R)$ проведена путем выполнения 100 итераций степенного метода [22]. Чтобы ускорить процедуру оценки значения спектрального радиуса, мы реализовали ее на GPU с использованием описанного выше параллельного алгоритма решения линейных систем с трехдиагональной матрицей.

Таблица 7

Масштабируемость метода BiCGStab с предобусловливателем CPR-AIPS

Матрица	Количество GPU	1	2	4
imsh	Время решения, с	0,043	0,031	0,028
	Число итераций	1	1	1
immn	Время решения, с	0,075	0,054	0,047
	Число итераций	2,5	2,5	2,5
krrv	Время решения, с	0,102	0,061	0,048
	Число итераций	2	2	2
mmnt	Время решения, с	0,108	0,063	0,048
	Число итераций	1,5	1,5	1,5
fdrv	Время решения, с	0,100	0,061	0,046
	Число итераций	1	1	1
kmms	Время решения, с	0,396	0,224	0,157
	Число итераций	5	5	4,5
lkms	Время решения, с	0,772	0,397	0,215
	Число итераций	6,5	6,5	6,5

Таблица 6 содержит рассчитанные значения спектральных радиусов матриц $P^{-1}R$, возникающих при использовании предобусловливателя CPR-AIPS в процессе решения тестовых СЛАУ, а также информацию о соотношении числа итераций метода BiCGStab при использовании предобусловливателей CPR-AMG и CPR-AIPS.

Из табл. 6 следует, что предобусловливатель CPR-AIPS обеспечивает равную с CPR-AMG скорость сходимости метода BiCGStab при значениях $\rho(P^{-1}R) < 0.98$. При увеличении значения спектрального радиуса проявляются различия в скорости сходимости с предобусловливателем CPR-AIPS относительно CPR-AMG: при $\rho(P^{-1}R) \sim 0.99$ можно получить как незначительное снижение числа итераций на матрице mmnt, так и существенное увеличение на матрицах kmms и lkms.

Указанные промежуточные результаты подтверждают необходимость учета спектральных свойств матрицы $P^{-1}R$ при использовании предобусловливателя AIPS и рассматриваются нами в качестве отправной точки в разработке алгоритма автоматизированного выбора предобусловливателя в задаче мо-

делирования многофазных фильтрационных потоков углеводородов в пористых средах.

3.4. Масштабируемость метода BiCGStab с предобусловливателем CPR-AIPS. В табл. 7 представлены результаты работы метода BiCGStab с предобусловливателем CPR-AIPS на 1–4 GPU NVIDIA Tesla V100.

Положительный момент заключается в том, что увеличение количества задействованных GPU не снижает скорость сходимости итерационного процесса, чему способствует естественное эквивалентное распараллеливание метода AIPS. Для наглядности ускорение решения СЛАУ на 2 и 4 GPU относительно 1 GPU представлено на рис. 7.

Наилучшие показатели масштабируемости достигаются при решении СЛАУ большей размерности. Это вполне ожидаемый эффект, так как при декомпозиции СЛАУ меньшей размерности не обеспечивается достаточный ресурс параллелизма для загрузки нескольких GPU.

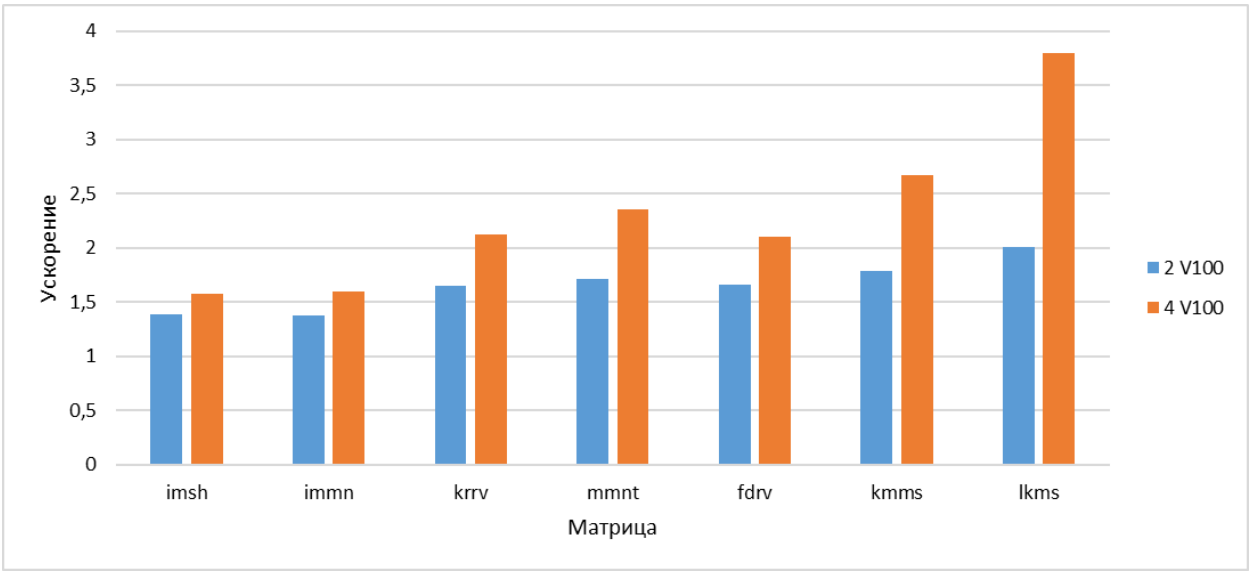

Рис. 7. Ускорение решения СЛАУ (BiCGStab + CPR-AIPS) на 2 и 4 GPU относительно 1 GPU

4. Заключение. В настоящей статье рассмотрена применимость метода AIPS, аппроксимирующего обратную матрицу на основе степенного разложения в ряд Неймана, в рамках двухступенчатого предобусловливателя CPR. Предложен ориентированный на архитектуру CUDA параллельный алгоритм решения линейных систем с трехдиагональной матрицей специального вида, состоящей из независимых блоков различного размера. Показано, что реализация предложенного алгоритма может более чем в 2 раза превосходить по быстродействию функции *cusparseDgtsv2_nopivot* и *cusparseDgtsv2* решения трехдиагональных линейных систем общего вида из библиотеки cuSPARSE. На гибридных вычислительных системах с GPU NVIDIA Tesla P100 и V100 проведена сравнительная оценка эффективности предобусловливателей CPR-AIPS и CPR-AMG, которая показала, что CPR-AIPS позволяет снизить относительно CPR-AMG время решения большинства тестовых СЛАУ, полученных при решении уравнений трехфазной фильтрации. Кроме того, отмечена приемлемая масштабируемость процедуры решения СЛАУ методом BiCGStab с предобусловливателем CPR-AIPS: ускорение на двух GPU V100 составляет от 1,4 до 2 раз, на четырех — от 1,6 до 3,8 раза относительно одного GPU. Рассмотрена взаимосвязь между спектральным радиусом используемых матриц и скоростью сходимости метода BiCGStab с предобусловливателем CPR-AIPS относительно CPR-AMG, которая может быть использована в целях автоматизации выбора наиболее эффективного предобусловливателя.

Работа выполнена при частичной финансовой поддержке Минобрнауки России, государственное задание № 1.3103.2017/4.6.

Статья рекомендована к публикации Программным комитетом Международной научной конференции “Суперкомпьютерные дни в России 2018” (http://russianscdays.org).

СПИСОК ЛИТЕРАТУРЫ

1. Топ50. http://top50.supercomputers.ru.

2. *Азиз Х., Сеттари Э.* Математическое моделирование пластовых систем. М.: Институт компьютерных исследований, 2004.
3. *Газизов И.И., Юлдашев А.В.* Разработка параллельного линейного решателя для задачи гидродинамического моделирования нефтегазовых месторождений // Эвристические алгоритмы и распределенные вычисления. 2014. **1**, № 1. 88–96.
4. AlgoWiki: Открытая энциклопедия свойств алгоритмов. Стабилизированный метод бисопряженных градиентов (BiCGStab).
`http://algowiki-project.org/en/Biconjugate_gradient_stabilized_method_(BiCGStab)`.
5. *Саад Ю.* Итерационные методы для разреженных линейных систем. М.: Изд-во Моск. ун-та, 2013.
6. *Губайдуллин Р.Р., Репин Н.В., Сайфутдинов Р.Ф., Юлдашев А.В.* Специализированный решатель разреженных систем линейных алгебраических уравнений на вычислительных кластерах, оснащенных графическими процессорами // Суперкомпьютерные дни в России: труды международной конференции (26–27 сентября 2016 г., Москва). М.: Изд-во МГУ, 2016. 673–682.
7. *Wallis J.R., Kendall R.P., Little T.E.* Constrained residual acceleration of conjugate residual methods // Proc. SPE Reservoir Simulation Symposium. 1985. https://doi.org/10.2118/13536-MS.
8. *Ruge J.W., Stuben K.* Algebraic multigrid (AMG) // Multigrid Methods. Vol. 3. Philadelphia: SIAM Press, 1987. 73–130.
9. *Недожогин Н.С., Копысов С.П., Новиков А.К.* Параллельное формирование предобуславливателя на основе обращения Шермана–Моррисона // Параллельные вычислительные технологии (ПаВТ'2015): труды международной научной конференции (31 марта–2 апреля 2015 г., г. Екатеринбург). Челябинск: Издательский центр ЮУрГУ, 2015. 436–441.
10. *Chen K.* Matrix preconditioning techniques and applications. Cambridge: Cambridge University Press, 2005.
11. *Prabhu H., Edfors O., Rodrigues J., Liu L., Rusek F.* Hardware efficient approximative matrix inversion for linear pre-coding in massive MIMO // IEEE International Symposium on Circuits and Systems (ISCAS). 2014.
`https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6865481`
12. *Fung L.S.K., Dogru A.H.* Parallel unstructured solver methods for complex giant reservoir simulation // Proceedings of the SPE Reservoir Simulation Symposium. 2007. https://doi.org/10.2118/106237-MS.
13. *Боресков А.В., Харламов А.А., Марковский Н.Д. и др.* Параллельные вычисления на GPU. Архитектура и программная модель CUDA. М.: Изд-во Моск. ун-та, 2012.
14. AlgoWiki: Открытая энциклопедия свойств алгоритмов. Прогонка, точечный вариант.
http://algowiki-project.org/ru/Прогонка,_точечный_вариант.
15. *Фролов А.В.* Еще один метод распараллеливания прогонки с использованием ассоциативности операций // Суперкомпьютерные дни в России: Труды международной конференции (28–29 сентября 2015 г., Москва). М.: Изд-во МГУ, 2015. 151–162.
16. cuSPARSE. https://docs.nvidia.com/cuda/cusparse/.
17. *Chang L.-W., Stratton J.A., Kim H.-S., Hwu W.-M.W.* A scalable, numerically stable, high-performance tridiagonal solver using GPUs // The International Conference for High Performance Computing, Networking, Storage and Analysis. Los Alamitos: IEEE Press, 2012. doi 10.1109/SC.2012.12
18. *Хокни Р., Джесссхоуп К.* Параллельные ЭВМ. Архитектура, программирование и алгоритмы. М.: Радио и связь, 1986.
19. *László E., Giles M., Appleyard J.* Manycore algorithms for batch scalar and block tridiagonal solvers // ACM Trans. Math. Softw. 2016. **42**, N 4. doi 10.1145/2830568
20. AlgoWiki: Открытая энциклопедия свойств алгоритмов. Классическая монотонная прогонка, повторный вариант. 2019.
http://algowiki-project.org/ru/Классическая_монотонная_прогонка,_повторный_вариант.
21. Суперкомпьютер УГАТУ. http://www.ugatu.su/supercomputer/.
22. *Вержбицкий В.М.* Вычислительная линейная алгебра. М.: Директ-Медиа, 2013.

Поступила в редакцию
29.09.2019

A Parallel Preconditioner Based on the Approximation of an Inverse Matrix by Power Series for Solving Sparse Linear Systems on Graphics Processors

A. V. Yuldashev [1], **N. V. Repin** [2], **and V. V. Spele** [3]

[1] *Ufa State Aviation Technical University, Faculty of General Science; ulitsa Karla Marksa 12, Ufa, 450008, Russia; Senior Lecturer, e-mail: art@ugatu.su*

[2] *State Research Institute of Aviation Systems; ulitsa Viktorenko 7, Moscow, 125319, Russia; Engineer, e-mail: repinn@gosniias.ru*

[3] *Ufa State Aviation Technical University, Faculty of General Science; ulitsa Karla Marksa 12, Ufa, 450008, Russia; Research Engineer, e-mail: spele.vova@ugatu.su*

Received September 29, 2019

Abstract: The applicability of the AIPS method approximating an inverse matrix using Neumann series is considered in the framework of the CPR two stage preconditioner. A parallel CUDA-oriented algorithm is proposed for solving linear systems with tridiagonal matrices consisting of independent blocks of different sizes. It is shown that the implementation of the proposed algorithm can be more than twice the speed of the similar functions from the cuSPARSE library. Experimental evaluation of the BiCGStab method with the CPR-AIPS preconditioner on modern GPUs, including a hybrid computing system with 4 GPU NVIDIA Tesla V100, is performed. Numerical experiments show an adequate scalability of this preconditioner as well as the possibility (compared to the CPR-AMG) to accelerate the solution of linear systems being typical for the reservoir modeling problems.

Keywords: CUDA, graphics processors, iterative methods, parallel computing, preconditioners, sparse matrices, tridiagonal systems.

References

1. Топ50. http://top50.supercomputers.ru. Cited October 25, 2019.

2. K. Aziz and A. Settari, *Petroleum Reservoir Simulation* (Appl. Sci. Publ., London, 1979; Comput. Sci. Inst., Moscow, 2004).

3. I. I. Gazizov and A. V. Yuldashev, "Development of Parallel Linear Solver for Hydrodynamic Modeling of Oil and Gas Fields," Evristich. Algortmy i Raspredel. Vychisl. **1** (1), 88–96 (2014).

4. AlgoWiki: Open Encyclopedia of Parallel Algorithmic Features. Biconjugate gradient stabilized method (BiCGStab).
`http://algowiki-project.org/en/Biconjugate_gradient_stabilized_method_(BiCGStab)`. Cited October 15, 2019.

5. Y. Saad, *Iterative Methods for Sparse Linear Systems* (SIAM, Philadelphia, 2003; Mosk. Gos. Univ., Moscow, 2013).

6. R. R. Gubaidullin, N. V. Repin, R. F. Sayfutdinov, and A. V. Yuldashev, "Specialized Solver of Sparse Linear Systems of Algebraic Equations on GPU Clusters," in *Proc. Int. Conf. on Russian Supercomputing Days, Moscow, Russia, September 26–27, 2016* (Mosk. Gos. Univ., Moscow, 2016), pp. 673–682.

7. J. R. Wallis, R. P. Kendall, and T. E. Little, "Constrained Residual Acceleration of Conjugate Residual Methods," in *Proc. SPE Reservoir Simulation Symposium, Dallas, USA, February 10–13, 1985*, https://doi.org/10.2118/13536-MS

8. J. W. Ruge and K. Stüben, "Algebraic Multigrid (AMG)," in *Multigrid Methods* (SIAM Press, Philadelphia, 1987), Vol. 3, pp. 73–130.

9. N. S. Nedozhogin, S. P. Kopysov, and A. K. Novikov. "Parallel Formation of a Preconditioner Based on the Sherman–Morrison Inversion," in *Proc. Int. Conf. on Parallel Computing Technologies, Ekaterinburg, Russia, March 31–April 2, 2015* (South Ural State Univ., Chelyabinsk, 2015), pp. 436–441.

10. K. Chen, *Matrix Preconditioning Techniques and Applications* (Cambridge Univ. Press, Cambridge, 2005).

11. H. Prabhu, O. Edfors, J. Rodrigues, et al., "Hardware Efficient Approximative Matrix Inversion for Linear Pre-coding in Massive MIMO," in *IEEE Int. Symp. on Circuits and Systems (ISCAS), Melbourne, Australia, June 1–5, 2014*
`https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6865481`

12. L. S. K. Fung and A. H. Dogru, "Parallel Unstructured Solver Methods for Complex Giant Reservoir Simulation," in *Proc. SPE Reservoir Simulation Symposium, Houston USA, February 26–28, 2007*, https://doi.org/10.2118/106237-MS

13. A. V. Boreskov, A. A. Charlamov, N. D. Markovskii, et al., *Parallel Computing on GPU. Architecture and CUDA Programming Model* (Mosk. Gos. Univ., Moscow, 2012) [in Russian].

14. AlgoWiki: Open Encyclopedia of Parallel Algorithmic Features. Thomas Algorithm, Pointwise Version.
`http://algowiki-project.org/en/Thomas_algorithm,_pointwise_version`. Cited October 15, 2019.

15. A. V. Frolov, "Yet Another Tridiagonal Matrix Algorithm Parallelizing Method," in *Proc. Int. Conf. on Russian Supercomputing Days, Moscow, Russia, September 28–29, 2015* (Mosk. Gos. Univ., Moscow, 2015), pp. 151–162.

16. cuSPARSE. The API Reference Guide for cuSPARSE, the CUDA Sparse Matrix Library. https://docs.nvidia.com/cuda/cusparse/. Cited October 25, 2019.

17. L.-W. Chang, J. A. Stratton, H.-S. Kim, and W.-M. W. Hwu, "A Scalable, Numerically Stable, High-Performance Tridiagonal Solver Using GPUs," in *Proc. Int. Conf. on High Performance Computing, Networking, Storage and Analysis, Salt Lake City, November 10–16, 2012* (IEEE Press, Los Alamitos, 2012), doi 10.1109/SC.2012.12

18. R. W. Hockney and C. R. Jesshope, *Parallel Computers: Architecture, Programming and Algorithms* (Adam Hilger, Bristol 1981; Radio i Svyaz', Moscow, 1986).

19. E. László, M. Giles, and J. Appleyard, "Manycore Algorithms for Batch Scalar and Block Tridiagonal Solvers," ACM Trans. Math. Softw. **42** (2016). doi 10.1145/2830568

20. AlgoWiki: Open Encyclopedia of Parallel Algorithmic Features. Repeated Thomas Algorithm, Pointwise Version.
`http://algowiki-project.org/en/Repeated_Thomas_algorithm,_pointwise_version`. Cited October 15, 2019.

21. USATU's Supercomputer. https://www.ugatu.su/en/research/supercomputer/. Cited October 15, 2019.

22. V. M. Verzhbitskii, *Numerical Linear Algebra* (Direkt-Media, Moscow, 2013) [in Russian].