Разработка параллельных алгоритмов для решения задач каротажа на графических процессорах

И.В. Сурodина, И.Б. Лабутин

Институт вычислительной математики и математической геофизики,
Институт нефтегазовой геологии и геофизики имени Трофимука

Для численного решения задач каротажа на графических процессорах (GPU) используется метод сопряженных градиентов (CG). В работе предлагается простой способ построения предобуславливателей, аппроксимирующих обратную матрицу. Эффективность данного метода продемонстрирована на задаче бокового каротажного зондирования (решение двумерного уравнения Пуассона). Предложенный метод легко обобщается на случай комплексных систем (задача высокочастотного электромагнитного каротажа) и может быть использован не только в методе CG. Данный подход позволяет улучшить и некоторые предобусловливатели, что продемонстрировано на примере SSOR-AI.

1. Введение.

Численное решение дифференциальных уравнений в частных производных приводит к системам линейных алгебраических уравнений (СЛАУ) высокого порядка. Задача решения таких систем находит широкое применение во многих инженерных и научных исследованиях в самых разных областях. Вследствие этого становятся актуальными быстрые программные реализации алгоритмов, решающие данную задачу. Этой тематике посвящено огромное количество работ. В последние годы очень активно часть вычислений переносят на графические процессоры (GPU), такие как NVIDIA Ge Force, Tesla и другие. Все современные суперкомпьютерные центры развивают гибридные кластеры, в которых для ускорения вычислений применяются и графические карты. Многие алгоритмы, успешно применяемые в ряде последовательно решаемых задачах, становятся неэффективными при решении на GPU. Разработка параллельных алгоритмов для решения многомерных задач остается очень актуальной и необходимой задачей.

В статье рассматривается численное решение уравнения Пуассона. Уравнение Пуассона возникает во многих приложениях: в гидродинамике, в электростатике, в магнитостатике, в геоэлектрике. Численное решение данного уравнения методом конечных разностей или методом конечных элементов приводит к системе линейных алгебраических уравнений с разреженной матрицей большой размерности, поэтому предпочтительнее использовать итерационные методы решения вместо прямых (метод Гаусса или факторизиция Холецкого).

Метод сопряженных градиентов (CG) является одним из лучших хорошо известных методов для решения систем с симметричными, положительно определенными матрицами. Известны его обобщения на случай комплексной области [1].

Использование предобуславливания [2,3] существенно повышает эффективность алгоритма. Предобусловленный метод сопряженных градиентов (PCG) доказал свою эффективность и работоспособность в широкой области приложений. Предобуславливание состоит в замене исходной системы уравнений на эффективно решаемую систему, имеющую то же самое решение.

Предобусловленный метод сопряженных градиентов успешно применяется для плохо обусловленных задач. Такие задачи возникают при электромагнитном каротаже, в частности при моделировании показаний зондов бокового каротажного зондирования (БКЗ). Плохая обусловленность матриц связана во-первых, с сильно неравномерной сеткой, необходимой для адекватного описания прибора (сгущающейся около источников и приемников и имеющей геометрических шаг к границам области) и имеющей сгущения около радиальных границ; во-вторых, с применением биополимерных буровых растворов, имеющих низкое значение сопротивления (0.02 -0.05 Ом м), появился большой контраст между удельным электрическим сопротивлением бурового раствора и пород-коллекторов. Все эти факторы приводит к плохо обу-

словленной задаче с сильно изменяющимися коэффициентами уравнений.

Наша цель – развить для такого рода задач PCG алгоритм на GPU архитектуре. Стандартные техники предобуславливания – LU разложение,неполная факторизация или симметричная последовательная верхняя релаксация (SSOR) остаются весьма трудоемкой для распараллеливания, хотя известны некоторые подходы к этому [4].

Простой предобуславливатель Якоби хорошо распараллеливается, но незначительно влияет на эффективность метода. Применение предобуславливателя Якоби на GPU рассматривается в [5]. Широко известны также полиномиальные предобуславливатели, многоуровневые (многосеточные) и их различные модификации [4,7] и предобуславливатели, связанные с нахождением приближенной обратной матрицы (AINV) [8,9]. Аппроксимация обратной матрицы очень привлекательна и имеет большое будущее для графических процессоров. Применение предобуславливающей матрицы в PCG алгоритме в данном случае сводится к матрично-векторным операциям, достаточно хорошо распараллеливаемым. Но техника построения приближенной обратной матрицы достаточно сложна, трудна в применении и нет никаких гарантий, что построенная матрица будет также симметричной и положительно определенной. В [6] предложен эвристический подход к построению неполного предобуславливателя Пуассона для решения уравнений Пуассона на нескольких GPU, но для нашей задачи этот подход не работает. В [10] рассматривается SSOR-AI предобуславливатель, который уже можно успешно применить к решению каротажных задач.

Мы предлагаем простой и хорошо распараллеливаемый способ получения предобуславливателей, аппроксимирующий обратную матрицу. Предлагаемый метод позволяет получить не только серию новых предобуславливателей, но также улучшить и существующие, в частности SSOR-AI.

На примере двумерной задачи БКЗ (решение уравнения Пуассона) показана высокая эффективность данного подхода для расчетов на GPU.

2. Метод сопряженных градиентов

В задачах, связанных со скважинной геоэлектрикой, удобно использовать цилиндрическую систему координат (r,φ,z). Для простоты рассмотрим изотропную среду с распределением проводимости $\sigma(r,z)$. С целью выделения в явном виде особенности решения задачи, связанной с источником первичного поля, искомый потенциал электрического поля U представим в виде суммы аномального потенциала U^a и первичного потенциала U^0, связанного с источником поля, расположенным в однородной среде с проводимостью σ_0:

$$U = U^0 + U^a.$$

Записывая уравнения непрерывности плотности токов, соответствующих полному и первичному потенциалам и вычитая из первого второе, приходим к следующему уравнению [11]:

$$div(\sigma \nabla U^a) = -div((\sigma - \sigma_0)\nabla U^0). \qquad (1)$$

В цилиндрической системе координат уравнение (1) примет вид

$$\frac{1}{r}\frac{\partial}{\partial r}\left(\sigma r \frac{\partial U^a}{\partial r}\right) + \frac{\partial}{\partial z}\left(\sigma \frac{\partial U^a}{\partial z}\right) = \frac{1}{r}\frac{\partial}{\partial r}\left((\sigma_0 - \sigma) r \frac{\partial U^0}{\partial r}\right) + \frac{\partial}{\partial z}\left((\sigma_0 - \sigma)\frac{\partial U^0}{\partial z}\right). \qquad (2)$$

В зависимости от вида источника U^0 может быть следующим:

1. Точечный источник, находящийся в однородной среде на вертикальной оси в точке z=0

$$U^0 = \frac{I}{4\pi\sigma_0 R},$$

I - сила тока, $R = \sqrt{r^2 + z^2}$.

2. Кольцевой источник постоянного тока с центром в начале координат расположен на диэлектрической трубе радиуса d в однородной среде

$$U^0(r,z,d)=\frac{I}{2\pi^2\sigma_0 R}\int_0^\infty K_0(mr)I_0(md)\cos(mr)dm\,,\quad r\ge d\,,$$

K_0, I_0 - модифицированные функции Бесселя 0-го порядка.

При удалении от источника потенциал затухает как $1/R$, поэтому для функции U^a вдали от источников $U^a\big|_{r=R}=0$, $U^a|_{z=\pm Z}=0$, $\partial U/\partial r|_{r=0}=0$ или $\partial U/\partial r|_{r=d}=0$.

Дискретизация уравнения (2) конечно-разностным методом и последующая его симметризация приводит к линейной системе

$$Ax=b, \qquad (3)$$

где A действительная, симметричная положительно определенная матрица.
Существует много методов решения подобных систем. Это прямые методы и итерационные. Поскольку матрица A разреженная и имеет большую размерность
выбор делают обычно в пользу итерационных методов.

Метод сопряженных градиентов является одним из лучших итерационных методов для решения разреженных положительно определенных линейных систем линейных алгебраических уравнений. Метод достаточно гибок, прост для применения и сходится теоретически за конечное число шагов. Для его реализации необходимы только матрично-векторные операции.

Алгоритм метода сопряженных градиентов (CG):

$k=0$: Инициализация x_0, $p_0=r_0=b-Ax_0$

$k\ge 0$: Пока $\|r_k\|/\|r_0\|>\varepsilon$

1. $q_k=Ap_k$, $\alpha_k=\dfrac{r_k^T r_k}{p_k^T q_k}$

2. $x_{k+1}=x_k+\alpha_k p_,\quad r_{k+1}=r_k-\alpha_k q_k$

3. $\beta_k=\dfrac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$, $p_{k+1}=r_{k+1}+\beta_k p_k$

На каждой итерации необходимо одно умножение матрицы на вектор и два скалярных произведения векторов. Все операции можно выполнить с использованием стандартных библиотек, например BLAS.

Пусть $(x,y)_A=x^TAy$ и соответствующая A норма $\|x\|_A=\sqrt{x^TAx}$.

Известна оценка скорости сходимости метода CG [15]. Если x^* - точное решение (3), то для последовательности решений $\{x_k\}$ имеем

$$\|x^*-x_k\|_A\le 2\|x^*-x_0\|_A\left(\frac{\sqrt{\mu-1}}{\sqrt{\mu+1}}\right)^k,\quad \text{где } \mu=\lambda_{\max}/\lambda_{\min},\ \lambda_{\max} \text{ и } \lambda_{\min} \text{ - максимальное и мини-}$$

мальное собственные числа матрицы A соответственно. Очевидно, что наилучшая сходимость будет достигаться при cond(A)= $\mu\approx 1$. На практике же $\mu\gg 1$. В рассматриваемом случае из-за сильно неравномерной сетки и большого контраста сред μ может достигать 10^{16}. Эффективность метода CG существенно повышается переобуславливанием системы. Вместо уравнения (3) решается уравнение

$$M^{-1}Ax=M^{-1}b \qquad (4)$$

или

$$AM^{-1}y = b, \quad x = M^{-1}y \quad , \tag{5}$$

где M также симметрична и положительно определена. Уравнение (4) соответствует левому предобуславливанию, уравнение (5) - правому. Матрица M выбирается так, что $\det A \neq 0$, M^{-1} легко вычислима и $M \approx A$. При этом $cond(M^{-1}A) \square\ cond(A)$ или $cond(AM^{-1}) \square\ cond(A)$.

Алгоритм предобусловленного метода сопряженных градиентов (PCG):

$k = 0$: Инициализация x_0, $r_0 = b - Ax_0, Mz_0 = r_0, p_0 = r_0$

$k \geq 0$: Пока $\|r_k\| / \|r_0\| > \varepsilon$

1. $q_k = Ap_k, \quad \alpha_k = \dfrac{z_k^{\ T} z_k}{p_k^{\ T} q_k}$
2. $x_{k+1} = x_k + \alpha_k p, \quad r_{k+1} = r_k - \alpha_k q_k$
3. $Mz_{k+1} = r_{k+1}$
4. $\beta_k = \dfrac{z_{k+1}^{\ T} r_{k+1}}{z_k^{\ T} r_k}, \quad p_{k+1} = r_{k+1} + \beta_k p_k$

В этом алгоритме появляется дополнительный шаг 3, вычисление z_{k+1}.

От его решения зависит эффективность параллельного алгоритма, так как остальные операции –матрично-векторные, которые хорошо распараллеливаются.

Левый предобуславливающий CG алгоритм с M - скалярным произведением математически эквивалентен правому предобуславливающему CG алгоритму с M^{-1} - скалярным произведением.

Для левого предобуславливания (4) обычно используется неполная LU факторизация. Шаг 3 сводится к последовательному решению двух треугольных систем, что не подходит для GPU –реализации. Можно найти способ, не требующий решения системы с матрицей M, а именно вычислить M^{-1} как аппроксимацию матрицы, обратной к A [8,9,10]. К сожалению, эти подходы сложны в реализации. Аппроксимацию обратной матрицы можно получить достаточно легко и просто, используя алгоритм Хоттенлинга.

3. Алгоритм Хоттенлинга.

Пусть D_0 - некоторое начальное приближение обратной матрицы. Составим произведение исходной матрицы A и D_0. Отклонение этого произведения от единичной матрицы указывает степень неточности полученных результатов:

$$R_0 = E - AD_0 \tag{6}$$

Если $\|R_0\| \leq q < 1$, (7)

где норма может быть взята любая, хорошо вычисляемая, то тогда можно составить итерационный процесс уточнения элементов обратной матрицы A^{-1} со сколь угодно большой точностью. Впервые процесс был предложен Хоттеллингом [12], изложен в [13].

Образуем последовательность матриц

$$\begin{aligned} &D_1 = D_0(E + R_0), \ R_1 = E - AD_1 \\ &D_2 = D_1(E + R_1), \ R_2 = E - AD_2 \\ &\ldots \quad \ldots \quad \ldots \qquad \ldots \quad \ldots \quad \ldots \\ &D_m = D_{m-1}(E + R_{m-1}), \ R_m = E - AD_m \end{aligned} \tag{8}$$

Теорема 1. Если начальное приближение обратной матрицы D_0 выбрано так, что $\|R_0\| \le q < 1$, то погрешность $\|D_m - A^{-1}\| \le \|D_0\| \frac{q^{2^m}}{1-q}$.

Доказательство.

Вначале покажем, что матрица $R_m = R_0^{2^m}$. Действительно,

$R_m = E - AD_m = E - AD_{m-1}(E + R_{m-1}) =$

$E - (E - R_{m-1})(E + R_{m-1}) = R_{m-1}^2 = R_{m-2}^4 = \cdots = R_0^{2^m}$

Отсюда следует, что

$$D_m = A^{-1}(E - R_0^{2^m}). \qquad (9)$$

Формула (9) показывает, что D_m стремится к A^{-1}, причем сходимость процесса очень быстрая. Принимая во внимание, что $A^{-1} = D_0(AD_0)^{-1} = D_0(E - R_0)^{-1}$, дадим оценку погрешности:

$$\|D_m - A^{-1}\| = \|-A^{-1}R_0^{2^m}\| = \|-D_0(E-R_0)^{-1}R_0^{2^m}\| \le \|D_0\|\|(E-R_0)^{-1}\|\|R_0^{2^m}\| \le \|D_0\|\frac{q^{2^m}}{1-q} \quad .\square$$

Из этой оценки видно, что как только начальное приближение выбрано так, что выполняется (7), то число верных десятичных знаков возрастает в геометрической прогрессии.

Теорема 2. Если $A = A^T$, $D_0 = D_0^T$, то $D_m = D_m^T$.

Доказательство.

Заметим, что

$$D_m = D_{m-1} = (E + R_{m-1}) = D_{m-1} + D_{m-1}(E - AD_{m-1}) = 2D_{m-1} - D_{m-1}AD_{m-1}$$

Тогда

$$D_m^T = 2D_{m-1}^T - D_{m-1}^T A^T D_{m-1}^T = 2D_{m-1} - D_{m-1}AD_{m-1} = D_m \,.\square$$

Сохранение симметрии оказывается очень хорошим и полезным в дальнейшем свойством этого алгоритма.

4. Построение предобуславливателей.

Применим этот алгоритм для получения предобуславливателей. В качестве D_0 возьмем предобуславливатель Якоби, т.е. $D_0 = diag\{a_{11}^{-1}, a_{22}^{-1}, \ldots, a_{nn}^{-1}\}$. Если $\|R_0\| \le q < 1$, то можем строить процесс. На первом шаге получим $D_1 = D_0 + D_0(E - AD_0)$. D_1 будет иметь такую же структуру, как и исходная матрица. В нашем случае двумерного уравнения Пуассона она пятидиагональная. Решение вспомогательной системы в методе PCG сведется к умножению вектора на матрицу.

Рассмотрим второй член в последовательности (8) приближающих обратных матриц

$$D_2 = D_1 + D_1(E - AD_1) = 2D_1 - D_1AD_1. \qquad (10)$$

Матрица D_2 является 25-диагональной. Однако ее можно выразить через более разреженные матрицы D_1 и R_0^2.

$$D_2 = D_1(E + R_0^2) \qquad (11)$$

Тогда для умножения матрицы D_2 на вектор потребуется одно умножение на $(E + R_0^2)$ - девятидиагональную и одно умножение на D_1 - пятидиагональную матрицы.

Далее, рассмотрим третий член в последовательности приближающих обратных матриц:

$$D_3 = D_2 + D_2(E - AD_2) = (2D_1 - D_1AD_1)(2E - A(2D_1 - D_1AD_1)) = \\ 2(2D_1 - D_1AD_1) - (2D_1 - D_1AD_1)A(2D_1 - D_1AD_1) \quad (12)$$

Этот подход потребует 7 умножений на пятидиагональные матрицы. Можно уменьшить количество умножений, используя выражение D_2 через R_0^2, аналогично предыдущему.

В итоге получим

$$D_3 = D_1(E + R_0^2)(2E - AD_1(E + R_0^2)) = 2D_1(E + R_0^2) - D_1(E + R_0^2)AD_1(E + R_0^2) .$$

(13)

Всего потребуется три умножения на пятидиагональные матрицы и два умножения на девятидиагональную.

Предложенный способ построения предобуславливателей может быть применен к матрице любой структуры, если выполнено условие (7). Сделав хотя бы первый шаг - D_1 - получаем предобуславливатель по структуре соответствующий исходной матрице.

Способность сохранять структуру исходной матрицы является достоинством предобуславливателя D_i, так как применение его в итерационном процессе сводится к операциям над матрицами схожей структуры. Такие операции могут быть эффективно реализованы на GPU.

Выбор подходящего предобуславливателя D_i является по сути компромиссом между точностью приближения обратной матрицы и производительностью (количеством операций над матрицей в итерационном процессе).

5. Предобуславливатель SSOR-AI

В работе [10] R.Helfenstein and J.Koko предложили предобуславливатель SSOR-AI, основанный на аппроксимации обратной матрицы из метода симметричной верхней релаксации (SSOR). Далее изложим принцип его построения.

Пусть $A = L + D + L^T$, D - матрица диагональных элементов A, L -нижняя треугольная часть A. SSOR предобуславливатель определяется как $M = KK^T$, где

$$K = \frac{1}{\sqrt{2-\omega}}(\overline{D} + L)\overline{D}^{-1/2},\ 0 < 2 < \omega \text{ и } \overline{D} = (1/\omega)D .$$

Матрица K может быть представлена следующим образом:

$$K = \frac{1}{\sqrt{2-\omega}}\overline{D}(I + \overline{D}^{-1}L)\overline{D}^{-1/2} \text{ и тогда } K^{-1} = \sqrt{2-\omega}\overline{D^{1/2}}(I + \overline{D}^{-1}L)^{-1}\overline{D}^{-1} .$$

Далее, предполагая что спектрадьный радиус $\rho(\overline{D}^{-1}L) < 1$, аппроксимируем K^{-1} рядом Неймана:

$$K^{-1} \approx \sqrt{2-\omega}\overline{D}^{-1/2}[I - \overline{D}^{-1}L + (\overline{D}^{-1}L)^2 - (\overline{D}^{-1}L)^3 + \ldots]\overline{D}^{-1}$$

Возьмем только первый член ряда Неймана

$$\overline{K} := \sqrt{2-\omega}\overline{D}^{1/2}(I - \overline{D}^{-1}L)\overline{D}^{-1} = \sqrt{2-\omega}\overline{D}^{-1/2}(I - L\overline{D}^{-1})$$

SSOR-AI предобуславливатель определим как $\overline{M} = \overline{K}^T\overline{K}$.

Мы предлагаем в качестве D_0 взять $\overline{M}$ в алгоритме Хоттенлинга. Это будет эквивалентно использованию второго, третьего и т.д. членов ряда Неймана.

Какой предобуславливатель использовать в качестве стартового - диагональный Якоби или SSOR-AI – все зависит от конкретной задачи. В SSOR-AI есть возможность управлять релаксационным параметром ω, что может оказаться более успешным в некоторых случаях.

6. Реализация на GPU.

CUDA (Compute Unified Device Architecture) – это программно-аппаратная архитектура параллельных вычислений от NVIDIA, позволяющая существенно увеличить вычислительную

производительность благодаря использованию GPU (графических процессоров). Программа в модели CUDA состоит из хост-программы исполняемой на центральном процессоре и ядра (kernel-program) исполняемого параллельно на GPU. Хост программа подготавливает данные и копирует их на GPU. Ядро обрабатывает данные, используя потенциально большое число параллельных потоков. Потоки ядра сгруппированы в массив блоков. Потоки внутри блока имеют доступ к разделяемой локальной памяти и поддерживают барьерную синхронизацию. Потоки из разных блоков не могут быть синхронизованы. Современные NVIDIA GPU состоят из набора мультипроцессоров с разделяемой памятью. Каждый мультипроцессор состоит из 8 скалярных процессоров и 16 kB высокоскоростной памяти. В Cuda Programming Guide[14] описаны приемы достижения максимальной производительности на GPU.

В алгоритме PCG наиболее трудоемкой операцией является умножение матрицы на вектор. В то время как для остальных операций линейной алгебры, необходимых для реализации метода сопряженных градиентов, использовались функции библиотеки cublas, поставляемой вместе с CUDA SDK.

Для хранения матрицы был выбран диагональный формат. Поскольку матрица симметрична, то храним только 3 из 5 диагоналей матрицы в одномерных массивах в глобальной памяти. Для оптимазации работы с памятью видеокарты использовались текстуры .

7. Численные эксперименты.

Решалась задача для пятидиагональной матрицы размером 17139*17139 с числом обусловленности

$$cond(A) = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{1,44*10^{4}}{1,68*10^{-6}} = 8,57*10^{9}$$

Время решения последовательного варианта с предобуславливателем SSOR составило ~1.5 с на компьютере Intel(R) Core (TM) 2 Quad CPU (при относительной норме невязки 10^{-9}).

В примерах мы ограничились построением D_3, так как дальнейшее построение потребует гораздо большего числа умножений матрицы на вектор.

Таблица 1.

	Iter	Time(sec)
D_0 (Jacobi)	2241	0.65
D_1	1427	0.32
$D_2(D_1)$	925	0.22
$D_2(R_0^2)$	926	0.2
$D_3(D_1)$	714	0.17
$D_3(R_0^2)$	716	0.14

В таблице приведены количество итераций и время решения системы уравнений на GPU NVIDIA GeForce GTX 480 с различными предобуславливателями, вычисленными согласно формулам (8)-(13).

Таблица 2.

	Iter	
SSOR-AI	1522	
D1(SSOR-AI)	1384	
D2(SSOR-AI)	656	

В таблице приведены количество итераций для последовательности предобуславливателей, по-

лученных методом Хоттелинга с начальной матрицей SSOR-AI.

8. Заключение

Предложенный метод построения приближенной обратной матрицы оказался достаточно простым и эффективным для параллельных вычислений. Метод применим для любых матриц - как вещественных так и комплексных и будет достаточно полезным для многих научных и инженерных расчетов.

Данный подход является достаточно гибким, сохраняет симметрию (если матрица и первое приближение симметричны) и разреженность, позволяет строить различные предобусловливатели с учетом конкретной структуры матрицы.

Литература

1. R.W.Freund. Conjugate Gradient type methods for linear systems with complex symmetric coefficient matrices. Numer.Math., 57:285 312, March 1990
2. R.Barrett, M.Berry, T.Chan, J.Demmel, J.Donato, J.Dongarra, V.Eijkhout, R.Rozo, C.Romine, H. van der Vorst. Templates for the solution of linear system: building blocks for iterative methods. www.netlib.org/templates/templates.pdf
3. J. Dongarra, I.Duff, D.,Sorensen, H.van der Vorst. Numerical Linear Algebra for High-Performance Computers, SIAM, Philadelphia, PA, 1998.
4. R.Li, Y.Saad GPU-Accelerated Preconditioned Iterative Linear Solvers Technical Report umsi-2010-112, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 2010.
5. S. Georgescu, H. Okuda, Conjugate gradients on graphic hardware: Performance & Feasibility, 2007. 07-129
6. M. Ament, G. Knittel, D. Weiskopf, W. Strasser, A parallel preconditioned conjugate gradient solver for the poisson problem on a multi-gpu platform, PDP ’10: Proceedings of the 2010 18th Euromicro Conference on Parallel, Distributed and Networkbased Processing (Washington, DC, USA), IEEE Computer Society, 2010, pp. 583–592.
7. Z. Feng, Z. Zeng. Parallel Multigrid Preconditioning on Graphics Processing Units (GPUs) for Robust Power Grid Analysis. DAC 2010: 661-666.
8. Jun Zhang. A sparse approximate technique for parallel preconditioning general sparse matrices.
9. G.Meurant A multilevel AINV preconditioner. Numerical Algorithms v 29 n 1-3 (2002) pp 107-129.
10. R.Helfenstein, J.Koko Parallel preconditioned conjugate gradient algorithm on GPU.
11. Ю.А. Дашевский, И.В. Суродина, М.И. Эпов. Квази-трехмерное математическое моделирование диаграмм неосесимметричных зондов постоянного тока в анизотропной среде. СИБЖВМ 2002, т.V,№3(11),с.76-91
12. Hottelling H. Analysis of a complex of statistical variables into principal components, J.Educ.Psych.,1933, **24**, 417-441,498-520.
13. Д.К.Фаддеев, В.Н.Фаддеева. Вычислительные методы линейной алгебры. Физматгиз, Москва-Ленинград, 1963.
14. NVIDIA Corporation, NVIDIA CUDA Programming Guide, NVIDIA, 2011