ВЫСОКОТОЧНЫЙ ДВУХУРОВНЕВЫЙ ПАРАЛЛЕЛЬНЫЙ ИТЕРАЦИОННЫЙ АЛГОРИТМ РЕШЕНИЯ ПЛОХООБУСЛОВЛЕННЫХ СЛАУ

С.А. Харченко

Введение. Современные пакеты прикладных программ (CAE) часто основаны на неявных схемах аппроксимации, что приводит к необходимости решать с высокой точностью плохо обусловленные разреженные системы линейных алгебраических уравнений (СЛАУ) огромной размерности. Для пользователей пакетов прикладных программ важно быть уверенным в точности и надежности моделирования. При этом традиционные алгоритмы решения СЛАУ не гарантируют того, что в различных параллельных режимах вычислений будет получено одно и то же решение. И чем более плохо обусловлена система уравнений, тем сильнее в различных режимах могут отличаться полученные векторы решения. Если при этом требуется проведение большого числа шагов моделирования, то в конечном итоге можно получить существенно разные характеристики моделируемых процессов, что не может не вызывать у пользователя сомнение в качестве моделирования.

В связи с этим имеет смысл рассмотреть задачу о построении эффективного параллельного алгоритма решения СЛАУ, который вне зависимости от режима параллельных вычислений строил бы одно и то же решение во всех значащих цифрах.

В данной работе рассматривается один из вариантов организации параллельных вычислений для достижения этой цели. Основная идея алгоритма состоит в итерационном решении СЛАУ со значительно большей точностью, чем это делается обычно. Действительно, решение невырожденной системы уравнений определено однозначно, а значит повышение точности решения должно рано или поздно приводить к одному и тому же решению при использовании ограниченного числа значащих цифр. Одним из естественных путей достижения более высокой точности при решении СЛАУ являлось бы использование большей точности представления вещественного числа при проведении итераций. К сожалению, ведущие производители современного вычислительного оборудования (производители центральных процессоров и производители ускорителей вычислений, таких как GPU) пока отказываются аппаратно поддерживать высокую точность представления вещественных чисел, а потому этот прямой путь пока не представляется возможным. По этой причине в данной работе представлен обходной способ решения задачи, а именно строится двухуровневая организация вычислений итерационного алгоритма. На верхнем уровне проводятся небольшие вычисления со значительно повышенной точностью на основе программной эмуляции с помощью пакета QD [1] четверной или даже восьмерной точности представления вещественного числа. На нижнем уровне проводятся итерации в обычной двойной точности для последовательности правых частей, порожденных внешним уровнем вычислений. При этом каждая последующая правая часть эффективно решается комбинированным параллельным MPI+threads [2] предобусловленным итерационным алгоритмом за счет использования методики сохранения подпространства в алгоритме SOFGMRES [3]. Параллельный алгоритм построения предобусловливание основан на геометрической редукции задачи [4] и использует алгоритмы неполной блочной факторизации второго порядка точности [5]. Как показано в приведенных результатах численных экспериментов, подобная организация вычислений позволяет минимальными дополнительными затратами добиться построения такого вектора решения СЛАУ, который является однозначно определенным во всех значащих цифрах в представлении двойной точности, в том числе для очень плохо обусловленных СЛАУ. Кроме того, как показано экспериментально, получаемое качество вектора решения значительно превосходит качество вектора решения, полученного применением прямого метода решения СЛАУ в представлении двойной точности. Существенным ограничением применимости подобного двухуровневого алгоритма является неявное предположение о том, что обусловленность решаемой СЛАУ не превосходит обратную машинную точность представления числа двойной точности, т.е. величину порядка 10^{16}. В противном случае итерации внутренней итерационной схемы будут проводиться для СЛАУ, вырожденной в представлении двойной точности. При этом для корректной работы алгоритма необходимы дополнительные вычисления, не рассматриваемые в данной работе.

Работы выполнялись в рамках работ по Программе Президиума РАН П-14.

Программная эмуляция высокой точности в пакете QD. Пакет программ QD [1] (quad -double/double-double) разработан в Ливерморской Национальной Лаборатории США и предназначен для программной эмуляции четверной и восьмерной точности вычислений с плавающей запятой на обычных процессорах с поддержкой двойной точности вычислений с плавающей запятой.

Основная идея программного комплекса QD состоит в том, чтобы хранить и проводить вычисления с «четверной» double-double точностью как с не вычисленной явным образом суммой двух вещественных чисел двойной точности, т. е.,

$$X^{(q)} = x_h^{(d)} + x_l^{(d)}, \quad (1)$$

где $x_h^{(d)}$ представляет собой вещественное число двойной точности, содержащее основную часть мантиссы и экспоненты (h - «high»), а $x_l^{(d)}$ соответственно представляет собой вещественное число двойной точности, содержащее остальную часть мантиссы и экспоненты (l-«low»). Разделение числа «четверной» точности $X^{(q)}$ (q - «quad») на две части поддерживается специальной программной нормировкой вычислений и обеспечивается в том числе существенно разной экспонентой чисел $x_h^{(d)}$ и $x_l^{(d)}$. Пакет программ QD написан на языке C++ и поддерживает в виде операторов основные операции с типом данных quad, такие как сложение и вычитание, умножение и деление чисел, вычисление с высокой точностью значений основных трансцендентных функций (sin, cos, ln и т.д.). В пакете имеются в готовом виде основные константы с четверной точностью, такие как π, e, и т.д. Инициализация значения четверной точности может осуществляться как явным присвоением двух вещественных чисел двойной точности $x_h^{(d)}$ и $x_l^{(d)}$, так и через написание числа высокой точности в виде строковой константы с последующим программным преобразованием данных.

Аналогичным образом в программном комплексе QD осуществляется поддержка «восьмерной» quad-double точности вычислений как не вычисленной явным образом суммы четырех вещественных чисел двойной точности, т. е.,

$$X^{(o)} = x_1^{(d)} + x_2^{(d)} + x_3^{(d)} + x_4^{(d)}. \quad (2)$$

Как и в случае «четверной» точности, программно поддерживаются в виде операторов C++ основные операции с данными «восьмерной» точности (o - «octal»), основные трансцендентные функции, основные константы в «восьмерной» точности и инициализации числа «восьмерной» точности явным образом через указание четырех чисел типа double и через строковую константу. Поддерживаются все виды перекрестных вычислительных операций, таких как сложения и умножения double с quad и octal, quad с octal и т.д.

Учитывая, что для величин типа double в C++ машинная точность есть величина порядка 10^{-16}, то для типа quad в пакете QD обеспечивается машинная точность порядка 10^{-32}, а для типа octal в пакете QD обеспечивается машинная точность порядка 10^{-64}.

Двухуровневый параллельный итерационный алгоритм высокоточного решения СЛАУ.

Рассмотрим СЛАУ

$$Ax = b \quad (3)$$

где A - невырожденная разреженная квадратная матрица большой размерности, b - вектор правой части, x - искомый вектор решения. В равенстве (3) предполагается, что все данные хранятся в обычном формате double.

Для построения высокоточного решения системы (3) наряду с этим равенством рассмотрим также аналогичное равенство в «четверной» точности quad:

$$Ax^{(q)} = b^{(q)} \quad (4)$$

где $b^{(q)}$ - преобразованный в quad вектор правой части, $x^{(q)}$ - искомый вектор решения в формате quad. При этом в качестве высокоточного решения системы (3) будем рассматривать приближение к решению системы (4), полученное с высокой точностью в формате «четверной» точности quad и затем преобразованное на выходе обратно к формату double. Аналогично можно рассматривать переход от (3) к системе типа (4), но уже для «восьмерной» точности. Кроме того, в некоторых случаях необходимо в качестве основной системы уравнений сразу рассматривать систему типа (4) с матрицей $A^{(q)}$, изначально поданной в формате повышенной точности. Подобные рассмотрения выходят за рамки этой работы.

Строить решение системы (4) будем следующим образом. Рассмотрим матричные соотношения вида:

$$\begin{cases} b^{(q)} = \beta^{(q)} P_{k+1}^{(q)} e_1^{(q)}, \\ A\, Y_k^{(q)} = P_{k+1}^{(q)} H_k^{(q)}, \\ Y_k^{(q)T} Y_k^{(q)} = I_k^{(q)}, \\ P_{k+1}^{(q)\ T} P_{k+1}^{(q)} = I_k^{(q)}. \end{cases} \quad (5)$$

Соотношения (5) аналогичны соотношениям алгоритма SOFGMRES(m) [3], имеют место после k итераций алгоритма верхнего уровня и поддерживаются в «четверной» точности вычислений. Первое равенство из (5) показывает, что первый вектор ортонормированного набора $P_{k+1}^{(q)}$ есть нормированный вектор правой

части. Ортонормированный набор векторов $Y_k^{(q)}$ строится через ортогонализацию текущего пробного вектора направления $\tilde{y}_k^{(q)}$ к предыдущему набору векторов. Новый вектор направления $p_{k+1}^{(q)}$ набора $P_{k+1}^{(q)}$ возникает при ортогонализации вектора $Ay_k^{(q)}$ к предыдущим k векторам набора $P_{k+1}^{(q)}$ и его нормировании, при этом коэффициенты ортогонализации и нормирования сохраняются соответствующим образом в прямоугольной верхней Хессенберговой матрице $H_k^{(q)}$. Все ортогонализации в соотношениях (5) осуществляются в «четверной» точности на основе преобразований Хаусхолдера «четверной» точности.

Приближение к решению системы (4) ищется в виде:

$$x_k^{(q)} = Y_k^{(q)} z_k^{(q)}, \tag{6}$$

откуда при условии минимизации второй нормы невязки решения системы (4) получаем в силу соотношений (5) следующее условие на вектор $z_k^{(q)}$:

$$\| \beta^{(q)} e_1^{(q)} - H_k^{(q)} z_k^{(q)} \| = min . \tag{7}$$

Задача минимизации (7) по аналогии с алгоритмом SOFGMRES(m) может быть эффективно решена преобразованиями Гивенса верхней Хессенберговой матрицы $H_k^{(q)}$.

Для построения следующего пробного вектора направления $\tilde{y}_{k+1}^{(q)}$ в четверной точности вычисляется невязка системы (4) для текущего приближения к решению $x_k^{(q)}$: $r_k^{(q)} = b^{(q)} - A x_k^{(q)}$. Далее, текущий вектор невязки $r_k^{(q)}$ преобразуется к двойной точности отбрасыванием младших мантиссы и экспоненты в представлении (1), получается вектор r_k в представлении double. Для него параллельным итерационным алгоритмом находится вектор приближения к решению СЛАУ

$$Ay_{k+1} = r_k , \tag{8}$$

при этом последовательность систем (8) решается как набор систем уравнений с одной и той же матрицей и последовательностью правых частей: для этого один раз строится предобусловливание, при переходе от одной правой части к другой производится сохранение подпространства на основе методики для алгоритма SOFGMRES(m) из работы [3]. Найденное приближение к решению системы (8) в формате double преобразуется в формат quad и рассматривается как следующий пробный вектор направления $\tilde{y}_{k+1}^{(q)}$. И т.д.

Алгоритм вычисления оценки числа обусловленности симметричной масштабированной матрицы. Многие результаты теории систем линейных уравнений, в том числе оценки точности вектора решения СЛАУ, включают в себя важную алгебраическую характеристику матрицы — ее спектральное число обусловленности. В симметричном положительно определенном случае это число есть отношение старшего к младшему собственных значений матрицы. Приведем один из простых способов вычисления оценки этой характеристики.

Пусть матрица A симметрична и положительно определена. Пусть задано положительное вещественное число d, $0 < d \ll 1$. Для сдвинутой матрицы $A + dI$, здесь I - единичная матрица, рассмотрим ее точное треугольное разложение: $A + dI = L L^T$. Имеет место следующее простое утверждение, которое приводим без доказательства.

***Лемма* 1.** *Если пара* $\{\lambda, x\}$ *есть собственная пара матрицы* A, *то пара* $\{\frac{\lambda}{\lambda + d}, x\}$ *есть собственная пара предобусловленной матрицы* $L^{-1} A L^{-T}$.

Пусть теперь решается СЛАУ (3) с симметричной и положительно определенной матрицей A. Алгоритм решения СЛАУ обычно включает в себя симметричное масштабирование матрицы — приведение всех диагональных значений матрицы к единице, для сохранения обозначений будем считать что матрица A уже масштабирована таким образом. Пусть далее для сдвинутой матрицы строится точное треугольное разложение как показано выше для некоторого значения параметра d и проводятся предобусловленные итерации алгоритма SOFGMRES(m) для нахождения решения СЛАУ (3). Из теории алгоритма SOFGMRES(m) следует, что младшее и старшее сингулярные числа треугольной матрицы R_k из соотношений этого алгоритма являются оценками изнутри соответственно младшего и старшего сингулярных значений предобусловленной матрицы $A(L L^T)^{-1}$, а значит они, очевидно, являются оценками изнутри младшего и старшего собственных значений матрицы $L^{-1} A L^{-T}$. Отсюда из Леммы 1 можно получить оценки изнутри младшего и старшего

собственных значений масштабированной матрицы A .

Результаты численных экспериментов.

Ниже представлены результаты численных экспериментов на некоторых тестовых задачах моделирования строительных конструкций, любезно предоставленных ООО «Еврософт» [6]. Характеристики тестовых задач представлены в Таблице 1. В таблице использованы обозначения: N — число неизвестных; Nz_U — число структурно ненулевых элементов в треугольнике матрицы; D_{min} — минимальное значение диагонального элемента матрицы; D_{max} — максимальное значение диагонального элемента матрицы.

Таблица 1 - Некоторые характеристики тестовых задач.

Название	N	Nz_U	D_{min}	D_{max}
«converg1»	131 106	1,70E+006	6,00E+003	1,70E+008
«mosfilm»	427 014	4,30E+006	1,50E+004	8,00E+012
«hock48d»	889 890	4,90E+007	6,10E+003	4,90E+009
«20»	3 247 200	4,40E+007	4,00E+005	6,60E+007

В Таблице 2 для задачи «mosfilm» приведены оценки младшего и старшего собственного значения матрицы, полученные на основе Леммы 1 при различных значениях сдвига в процессе решения системы уравнений. В таблице использованы обозначения: d - сдвиг масштабированной матрицы; ξ_{min} - оценка сверху младшего собственного значения предобусловленной матрицы из алгоритма SOFGMRES(m); ξ_{max} - оценка снизу старшего собственного значения предобусловленной матрицы из алгоритма SOFGMRES(m); λ_{min} - вычисленная оценка младшего собственного значения масштабированной матрицы, полученная из Леммы 1; λ_{max} - вычисленная оценка старшего собственного значения масштабированной матрицы, полученная из Леммы 1; λ_{max}^{*} - оценка старшего собственного значения масштабированной матрицы, полученная путем проведения некоторого числа итераций (в данном случае 100) с масштабированной матрицей без предобусловливания. Из таблицы хорошо видно, что для получения более точной оценки младшего собственного значения масштабированной матрицы необходимо делать б**о**льший сдвиг, чтобы при этом получить более плохую сходимость к решению СЛАУ, а значит и более точную оценку за счет б**о**льшего числа итераций. Что касается старшего собственного значения, то согласно теории в данном случае для симметричной положительно определенной матрицы старшее собственное значение предобусловленной матрицы обязано быть не больше 1, чего не наблюдается экспериментально. По видимому, это связано с возмущением, вносимым в матрицу при ее симметричном диагональном масштабировании, поскольку для этой матрицы, как следует из Таблицы 1, разброс величин диагональных значений составляет более восьми порядков. По этим причинам для получения оценки старшего собственного значения были проведены итерации для не предобусловленной масштабированной матрицы.

Таблица 2 - Оценки младшего и старшего собственных значений масштабированной матрицы «mosfilm» при различных значениях сдвига.

d	ξ_{min}	ξ_{max}	λ_{min}	λ_{max}	λ_{max}^{*}
1,00E-003	3,93E-006	2,01E+000	3,93E-009	-	1.13E+001
1,00E-004	4,58E-005	3,15E+000	4,58E-009	-	
1,00E-005	4,93E-004	4,70E+000	4,93E-009	-	

В Таблице 3 для матрицы «converg1» представлены времена счета предобусловленной итерационной схемы SOFGMRES(m) с относительной точностью 10^{-9} по норме невязки при решении СЛАУ с одним и тем же предобусловливанием для различных форматов представления вещественных векторных данных — double, quad и octal. В таблице использованы обозначения: Nit — число итераций итерационной схемы; TimeDbl — время итераций в формате double; TimeQuad — время итераций в формате quad; TimeOctal — время итераций в формате octal. Из таблицы хорошо видно, что явное проведение всех вычислений в итерационной схеме с использованием программной эмуляции высокой точности приводит к очень значительному замедлению вычислений по сравнению с использованием аппаратно поддерживаемого формата double. Все эксперименты проводились на одном узле суперкомпьютера «Ломоносов» в режиме 1MPI x 4 threads.

Таблица 3 - Времена итераций алгоритма SOFGMRES(m) для различных форматов векторных данных для матрицы «converg1».

Nit	TimeDbl, c	TimeQuad, c	TimeOctal, c
97	6,5	57,8	787

В Таблице 4 для набора тестовых матриц для одного и того же предобусловливания приведены времена и числа итераций в различных параллельных режимах для двухуровневой итерационной схемы с относительной точностью 10^{-26} по норме невязки по сравнению с числом итераций и временами для обычной итерационной схемы с относительной точностью 10^{-9} по норме невязки. Все эксперименты проводились на суперкомпьютере «Ломоносов». Из таблицы следует, что дополнительные накладные расходы для получение высокоточного решения относительно невелики в силу свойства сверхлинейной сходимости алгоритмов крыловского типа. Более того, использование техники сохранения пространства в алгоритме SOFGMRES(m) в задаче «20» позволяет получить высокоточное решение за дополнительные затраты порядка всего 25% от основного времени счета! Кроме того, на задаче «20» наблюдается хорошая параллельная масштабируемость алгоритма решения СЛАУ при увеличении числа вычислительных узлов компьютера.

Таблица 4 - Времена итераций одно и двухуровневого итерационных алгоритмов.

Задача, режим	Nit, $\epsilon=10^{-9}$	Time, c. $\epsilon=10^{-9}$	Nit, $\epsilon=10^{-26}$	Time, c. $\epsilon=10^{-26}$
«converg1», 1 x 4	97	6,5	172	11,5
«mosfilm», 1 x 4	218	69	307	112
«hock48d», 1 x 4	102	60	229	186
«20», 2 x 4	420	687	532	1002
«20», 4 x 4	427	352	537	451
«20», 8 x 4	500	196	617	254

В таблице 5 представлена достигнутая точность вектора решения для плохо обусловленной задачи «mosfilm» (см. также Таблицу 2) при различных способах нахождения решения. В качестве «точного» вектора решения рассматривается вектор, полученный проведением итераций алгоритма SOFGMRES(m) в «восьмерной» точности octal с критерием остановки: относительное падения нормы невязки в 10^{50} раз. Производится сравнение относительной нормы «ошибки» и максимума относительной «ошибки» каждой компоненты вектора решения от «точного». Сравнение проводится для трех векторов: I) вектора, полученного прямым методом решения в формате double; II) итерационного вектора решения, найденного в формате double с относительной точностью падения нормы невязки в 10^{9} раз; III) итерационного вектора решения, найденного проведением итераций двухуровневой итерационной схемы, верхний уровень — в quad, нижний — в double, критерий остановки - относительное падения нормы невязки в 10^{26}, с последующим переводом вектора решения из формата quad в формат double. Вычисление отклонения вектора решения проводилось в формате octal. Из таблицы следует, что в double векторе решения, полученном двухуровневым алгоритмом, все значащие цифры являются правильными! При этом вектор решения, найденный традиционным способом — с использованием прямого или итерационного алгоритма — имеет относительно небольшую относительную норму ошибки, но при этом некоторые конкретные компоненты вектора решения, малые по абсолютной величине, имеют большую относительную ошибку.

Таблица 5 — Относительная точность векторов решения для задачи «mosfilm».

Задача	Прямой метод		Итерационный		Двухуровневый	
	Норма ошибки	Компоненты ошибки	Норма ошибки	Компоненты ошибки	Норма ошибки	Компоненты ошибки
«mosfilm»	$4\cdot10^{-8}$	0.04	$1\cdot10^{-9}$	0.002	$5\cdot10^{-17}$	$1\cdot10^{-16}$

Выводы. Представленный в работе высокоточный двухуровневый масштабируемый параллельный алгоритм решения СЛАУ позволяет небольшими дополнительными затратами находить высокоточное решение плохообусловленных систем линейных уравнений. В работе также предложена методика вычисления оценки числа обусловленности масштабированной симметричной положительно определенной матрицы.

ЛИТЕРАТУРА:

1. http://crd.lbl.gov/~dhbailey/mpdist/
2. Г.Б. Сушко, С.А. Харченко, "Экспериментальное исследование на СКИФ МГУ “Чебышев” комбинированной MPI+threads реализации алгоритма решения систем линейных уравнений,

возникающих во FlowVision при моделировании задач вычислительной гидродинамики". Материалы международной научной конференции "Параллельные вычислительные технологии" (ПаВТ'2009), Нижний Новгород, 30 марта – 3 апреля 2009 г., Челябинск: Изд. ЮУрГУ, 316-324, 2009.

3. С.А. Харченко, "Параллельная реализация итерационного алгоритма решения несимметричных систем линейных уравнений с частичным сохранением спектральной/сингулярной информации при явных рестартах" // "Вычислительные методы и программирование", 2010, т. 11, 373-381.
4. С.А. Харченко, "Влияние распараллеливания вычислений с поверхностными межпроцессорными границами на масштабируемость параллельного итерационного алгоритма решения систем линейных уравнений на примере уравнений вычислительной гидродинамики. Материалы международной научной конференции "Параллельные вычислительные технологии" (ПаВТ'2008), Санкт-Петербург, 28 января – 1 февраля 2008 г. Челябинск, Изд. ЮУрГУ, 2008, с.494-499.
5. I. E. Kaporin, "High Quality Preconditioning of a General Symmetric Positive Definite Matrix Based on its $U^T U + U^T R + R^T U$ decomposition // Numer. Linear Algebra Appl. – 1998. – Vol. 5. – P. 483-509.
6. В.Л. Якушев и др. "Решение плохообусловленных симметричных СЛАУ для задач строительной механики параллельными итерационными методами" (статья представлена на международную суперкомпьютерную конференцию «Научный сервис в сети Интернет 2011: экзафлопсное будущее»).