РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ

А В Т О М Е Т Р И Я

2007, том 43, № 2

УДК 519.6

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ МЕТОДА ПОЛУСОПРЯЖЕННЫХ НЕВЯЗОК ДЛЯ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

С. Г. Пудов

Конструкторско-технологический институт вычислительной техники СО РАН, г. Новосибирск
E-mail: pudov@dote.ru

Представлен предобусловленный алгоритм полусопряженных невязок для решения систем алгебраических уравнений с несимметричной квадратной матрицей. Большая часть вычислений в нем приходится на векторные операции, число которых растет квадратично с увеличением количества хранимых направляющих векторов. Экспериментально исследуется параллельная реализация этого алгоритма, причем в качестве предобусловливания выбрана модификация Айзенштата. Для специального вида задач приводится алгоритм распараллеливания матричных операций.

Введение. Решается система линейных алгебраических уравнений

$$Au = f, \quad u, f \in R^N, \quad A \in R^{N,\,N}, \tag{1}$$

с квадратной несимметричной матрицей A, которая является положительно-определенной в смысле выполнения условий

$$(Au, u) \geq \delta \|u\|^2, \quad \delta > 0, \quad \|u\|^2 = (u, \mathrm{v}), \quad (u, \mathrm{v}) = (\mathrm{v}, u) = \sum_{i=1}^{N} u_i \mathrm{v}_i,$$

для всех ненулевых вещественных векторов $u = \{u_i\}$ размера N. Предполагается, что матрица A является сильно разреженной, поэтому хранится в разреженном строчном формате [1], т. е. для каждой строки определено: количество ненулевых элементов, номера столбцов, в которых они расположены, и собственно значения этих ненулевых элементов.

Для итерационного решения к исходной системе (1), как правило, применяются предобусловливающие преобразования

$$\widetilde{A}u = B^{-1}Au = B^{-1}f = \widetilde{f}. \tag{2}$$

Это позволяет уменьшить число обусловленности матрицы и соответственно ускорить сходимость методов, но существенно затрудняет распараллеливание.

Рассматривается решение предобусловленной системы (2) с помощью итерационного процесса, называемого алгоритмом обобщенных сопряженных невязок [2]:

$$\begin{aligned} r^0 &= f - Au^0; \quad p^0 = r^0; \\ u^n &= u^{n-1} + \alpha_{n-1} p^{n-1}; \\ r^n &= r^{n-1} - \alpha_{n-1} Ap^{n-1}, \end{aligned} \tag{3}$$

где p^n – линейно независимые векторы, удовлетворяющие условиям

$$(A^t Ap^k,\ p^n) = (Ap^k,\ Ap^n) = \rho_n \delta_{k,n}, \quad \rho_n = (Ap^n,\ Ap^n)$$

($\delta_{n,k}$ – символ Кронекера). Коэффициенты α_k вычисляются по формуле

$$\alpha_k = (r^k,\ Ar^k)/(Ap^k, Ap^k) \tag{4}$$

из условия минимизации нормы вектора невязки r^n в подпространстве Крылова

$$K_n(r_0, A) = \text{span}\{p^0,\ p^1, \ldots,\ p^{n-1}\} = \text{span}\{p^0, Ap^0, \ldots, A^{n-1} p^0\},$$

а направляющие векторы p^k вычисляются по формуле

$$p^n = r^n + \sum_{k=0}^{n-1} \beta_{n,k} p^k, \quad \beta_{n,k} = -(Ap^k,\ Ar^n)/(Ap^k,\ Ap^k). \tag{5}$$

Критерием остановки итераций является выполнение условия

$$(r^n,\ r^n) < \varepsilon (f, f), \tag{6}$$

где ε – задаваемая точность вычислений.

Вышеописанный метод допускает эффективное распараллеливание, если удается распараллелить матричные операции, что достаточно тяжело сделать для большинства предобусловливающих преобразований. Однако при большом количестве используемых направляющих векторов основная часть вычислений приходится на выполнение векторных операций, следовательно, уже только их параллельная реализация должна давать существенный выигрыш во времени вычислений.

Целью предлагаемой работы является экспериментальное исследование эффективности распараллеливания описанного метода с предобусловливанием Айзенштата на системах с общей памятью, причем распараллеливаются как векторные операции, так и матричные для систем специального вида.

Метод полусопряженных невязок. Формулы (3)–(5) описывают длинную рекурсию, когда новый направляющий вектор p^n должен вычисляться

через все предыдущие, причем кроме p^k для вычисления $\beta_{n,k}$ требуется знание Ap^k, что приводит к необходимости хранения всех этих векторов.

Сделаем следующие замечания:

1) если A – симметричная положительно-определенная матрица, то $\beta_{n,k} = 0,\ k = 0, \ldots, n-1,$ и, следовательно, формулы (3)–(5) описывают метод минимальных (сопряженных) невязок [3];

2) известно, что расчет по формулам (5) при больших n численно неустойчив [4], поэтому при реализации они могут быть заменены формулами модифицированной (устойчивой) ортогонализации Грама – Шмидта:

$$\begin{aligned} & p^{n,0} = r^{n-1}; \quad Ap^{n,0} = Ar^{n-1}; \\ & \beta_{n,l} = -(Ap^{n,l}, Ap^l)/(Ap^l, Ap^l); \\ & p^{n,l+1} = p^{n,l} + \beta_{n,l} p^l; \\ & Ap^{n,l+1} = Ap^{n,l} + \beta_{n,l} Ap^l; \\ & p^k = p^{n,k}; \\ & Ap^k = Ap^{n,k}. \end{aligned} \tag{7}$$

Далее под методом полусопряженных невязок будем понимать вычисления по формулам (3), (4), (6), (7). Важным моментом при реализации метода является использование ресурсов памяти и процессора: с ростом n увеличивается как объем требуемой оперативной памяти (уже отмечалось, что необходимо сохранять $2n$ векторов длины N), так и объем вычислений, ведь направляющий вектор с номером n находится через все предыдущие n. Здесь для экономии памяти возможны два варианта:

1. По достижении номером текущей итерации некоторого значения m_r производить полный перезапуск алгоритма, т. е. использовать текущее решение в качестве начального приближения и повторять вычисления (3) заново. Более формально этот процесс будет иметь следующий вид:

$$\begin{aligned} & r^{km_r} = f - Au^{km_r}; \quad p^{km_r} = r^{km_r}; \\ & u^{km_r+m} = u^{km_r+m-1} + \alpha_{km_r+m-1} p^{km_r+m-1}; \\ & r^{km_r+m} = r^{km_r+m-1} - \alpha_{km_r+m-1} Ap^{km_r+m-1} \end{aligned} \tag{8}$$

(здесь k – номер макроитерации; m – номер шага между перезапусками, $0 \le \le m \le m_r$). При достижении $m = m_r$ вычисляется новое значение $u^{(k+1)m_r}$ и осуществляется переход на новую макроитерацию.

2. Начиная с итерации m_0, производить сдвиг направляющих векторов. В этом случае самый старый вектор «выкидывается» из памяти, а на его место помещается новый. Таким образом, новое приближение после достижения алгоритмом своего «потолка» строится по последним m_0 векторам. Следовательно, формулы (5) при $n > m_0$ преобразуются в

$$p^n = r^n + \sum_{k=n-m_0-1}^{n-1} \beta_{n,k} p^k, \quad \beta_{n,k} = -(Ap^k, Ar^n)/(Ap^k, Ap^k). \tag{9}$$

В любом случае скорость сходимости алгоритма замедляется по сравнению с первоначальным вариантом и будет существенно снижаться при уменьшении значений m_r и m_0, поскольку минимизация нормы невязки осуществляется на подпространствах Крылова меньшей размерности. Далее максимально возможное количество одновременно хранимых в памяти направляющих векторов p^k, $k = 0, \ldots, m_0 - 1$, будем называть уровнем алгоритма.

Предобусловливание исходной системы. Рассмотрим матрицу A в виде $A = D - L - U$, где D, L, U – диагональная, строго верхняя и нижняя треугольные матрицы соответственно. В качестве предобусловливания выберем модификацию Айзенштата [5], которая определяется следующим образом:

$$\widetilde{A} = L_B^{-1} A U_B^{-1} = U_B (B^{-1} A) U_B^{-1}; \quad L_B = (G - L) G^{-1/2}; \quad U_B = G^{-1/2} (G - U);$$

$$B = L_B U_B = (G - L) G^{-1} (G - U); \quad G = \frac{1}{\omega} D - \theta S; \quad Se = \left(\frac{1 - \omega}{\omega} D + L G^{-1} U \right) e, \tag{10}$$

где ω, θ – релаксационный и компенсационный параметры соответственно; S и G – диагональные матрицы; e – вектор с единичными компонентами. Поскольку G – симметричная положительно-определенная матрица, то система (2) преобразуется к эквивалентной предобусловленной системе

$$\begin{aligned} &\widetilde{A} = (I - \widetilde{L})^{-1} + (I - \widetilde{U})^{-1} - (I - \widetilde{L})^{-1} (2I - \widetilde{D}) (I - \widetilde{U})^{-1}, \\ &\widetilde{L} = G^{-1/2} L G^{-1/2}; \quad \widetilde{U} = G^{-1/2} U G^{-1/2}; \quad \widetilde{D} = G^{-1/2} D G^{-1/2}; \\ &\widetilde{A} \widetilde{u} = \widetilde{f}, \quad \widetilde{u} = (I - \widetilde{U}) G^{1/2} u, \quad \widetilde{f} (I - \widetilde{L})^{-1} G^{-1/2} f. \end{aligned} \tag{11}$$

Тогда умножение матрицы на вектор в новой системе можно переписать следующим образом:

$$\widetilde{A} p = q + (I - \widetilde{L})^{-1} [p - (2I - \widetilde{D}) q], \quad q = (I - \widetilde{U})^{-1} p. \tag{12}$$

Представленное выше преобразование исходной системы характеризуется тем, что число обусловленности матрицы в новой системе становится меньше, причем число арифметических операций асимптотически остается таким же, как и для умножения Ap исходной матрицы на вектор. Отметим при этом, что формулы (12) практически невозможно распараллелить, за исключением матриц A специальной структуры.

Распараллеливание матричных вычислений. Рассмотрим ситуацию, когда формулы (12) допускают параллельную реализацию, например, комплексное уравнение Гельмгольца $\nabla \lambda \nabla \bar{E} - \kappa \bar{E} = \alpha \bar{J}$ в прямоугольной области. Методом конечных объемов строится (комплексная) система линейных алгебраических уравнений $A_c u_c = f_c$, $A_c \in C^{N, N}$, $u_c, f_c \in C^N$, которая затем преобразуется в систему $Au = f$, $A \in R^{2N, 2N}$, где первые N строк соответствуют действительной части решения, остальные N строк относятся к мнимой части. Матрица полученной системы имеет блочную структуру второго порядка (рис. 1), где по главной диагонали стоят две разреженные матрицы A_{11} и A_{22}, а A_{12} и A_{21} представляют собой диагональные матрицы.

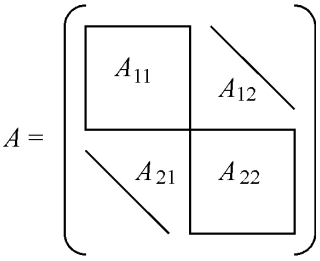

Рис. 1. Блочная структура матрицы для комплексного уравнения Гельмгольца

Рассмотрим умножение матрицы A на вектор по формулам (12), которое сводится к решению треугольных систем алгебраических уравнений. Опишем идею распараллеливания на примере решения системы уравнений с верхнетреугольной матрицей. Систему начинаем решать снизу. После того как определено несколько значений искомого вектора (например, в области 1 на рис. 2), далее шаги можно выполнять параллельно: последующие компоненты в нижней части матрицы и первые компоненты в верхней части матрицы (этап 2 решения системы). По окончании вычислений шаги с номерами 3–5 также можно производить параллельно. В итоге портрет матрицы системы позволяет вычислять последовательно только первый и последний этапы решения системы. Промежуточные этапы можно выполнять попарно параллельно: чем мельче разбиение этих блоков, тем больше вычислений может быть проведено параллельно и тем больше усилий требуется на синхронизацию вычислений. Оптимальное разбиение зависит как от вычислительной системы, на которой решается задача, так и от размеров решаемой задачи.

Особенности реализации алгоритма. Для проведения численных экспериментов была разработана программа, реализующая вышеописанный алгоритм предобусловленных полусопряженных невязок с устойчивой ортогонализацией Грама – Шмидта, позволяющая организовывать вычисления как в режиме перезапуска алгоритма (8), так и в режиме сдвига векторов (9), а также в любой их комбинации. Отдельно задается уровень алгоритма (level), максимальное количество итераций ($n_{\max}$) и количество итераций (restart) между перезапусками. Это позволяет гибко организовывать процесс вычислений.

З а м е ч а н и е. Сдвиг направляющих векторов в программе реализован через косвенную адресацию, т. е. через циклический сдвиг индексов массива.

Например, пусть $n_{\max}$ = restart = 1000, level = 100. В этом случае перезапуск не происходит, а на каждой итерации с номером больше 100 производится сдвиг направляющих векторов. Если же $n_{\max}$ = 1000, а restart = level = = 100, то после каждой итерации, кратной 100, программа перезапускается без сдвига. И наконец, если $n_{\max}$ = 1000, restart = 500, level = 100, то программа работает следующим образом: на 100-й итерации алгоритм заполняет векторами всю выделенную память, после чего на итерациях до 500-й производится сдвиг направляющих векторов. На 500-й итерации происходит полный перезапуск алгоритма, и все начинается за-

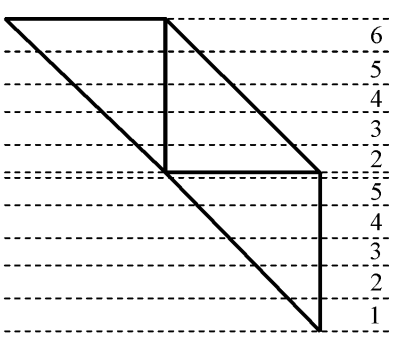

Рис. 2. Разбиение треугольной матрицы на попарно независимые участки вычислений

ново (если задача не будет решена раньше).

1. Теоретически известно, что чем больше уровень алгоритма, тем быстрее алгоритм сходится к решению. Ограничение сверху накладывается, во-первых, требованиями устойчивости вычислений (для каждой матрицы определяется индивидуально), во-вторых, объемом имеющейся оперативной памяти системы. Например, если рассмотреть задачу со 100000 неизвестных, то можно подсчитать, сколько памяти потребуется на различные уровни алгоритма. Для $n = 50$ получаем объем памяти для хранения векторов $2 \times 50 \times 100000 \times 8$ байт = = 80000000 байт, т. е. около 80 Мбайт. При уровне $n = 500$ получим уже 800 Мбайт, соответственно при $n = 1000$ объем требуемой памяти превысит 1,5 Гбайт. Если размер решаемой задачи больше, то меняется и объем требуемой памяти.

Таблица 1

Разбиение матрицы на участки	Время выполнения операций, с
Без разбиения	19,4
3	14,8
5	13,2
7	13,1
10	13,05

2. При больших уровнях алгоритма основной объем вычислений (до 90 %) приходится на векторные операции, оставшийся объем – на операции с матрицами. Поэтому, даже если матричные операции сохранить последовательными, распараллеливание только векторных вычислений может принести существенное ускорение работы программы, особенно за счет увеличения объема кэш процессоров, участвующих в вычислениях.

Программа распараллелена с использованием библиотеки OpenMP и протестирована на различных матрицах и разных уровнях алгоритма на вычислительной системе, состоящей из четырех процессоров Itanium2 (1,5 ГГц, кэш 4 Мбайт, 12 Гбайт оперативной памяти), под управлением операционной системы RedHat Enterprise Linux v.3. При этом максимальное количество нитей OpenMP ограничивалось двумя.

Результаты экспериментов. В качестве тестовой была взята система размером $n = 257985$ (половина действительной матрицы), количество ненулевых элементов 761796. Цель экспериментов состояла не в решении задачи, а в получении коэффициентов ускорения векторных и матричных операций.

1. Результаты распараллеливания матричных операций даны в табл. 1. Уже при разбиении каждой из половинок матрицы на пять последовательных участков ускорение составило 1,5 и практически не изменялось при увеличении количества участков разбиения.

2. Результаты распараллеливания векторных операций (матричные вычисления производятся последовательно) даны в табл. 2: level = 200, miter = = 400, restart = 200. Векторные операции распараллелились с ускорением более чем в 2 раза по сравнению с последовательной версией, а общее время работы программы уменьшилось в 1,8 раза.

Таблица 2

Матричные вычисления, с		Векторные вычисления, с		Общее время, с	
1 процессор	2 процессора	1 процессор	2 процессора	1 процессор	2 процессора
19,4	19,5	70,0	31,0	89,4	50,5

Таблица 3

Матричные вычисления, с		Векторные вычисления, с		Общее время, с	
1 процессор	2 процессора	1 процессор	2 процессора	1 процессор	2 процессора
19,4	20,2	40,2	17,3	59,9	38,1

3. Результаты распараллеливания векторных операций (матричные вычисления производятся последовательно) представлены в табл. 3: level = 100, miter = 400, restart = 100. Здесь векторные операции распараллелились с ускорением более чем в 2 раза, а общее время работы программы уменьшилось более чем в 1,5 раза.

Заключение. Алгоритм полусопряженных невязок эффективно распараллеливается на системах с общей памятью, при этом за счет большого объема векторных вычислений последовательное или параллельное (которое возможно только для узкого класса задач) выполнение матричных операций практически не влияет на коэффициент распараллеливания.

Отметим, что увеличение итерационной скорости сходимости метода полусопряженных невязок может быть достигнуто за счет применения более эффективных предобусловливающих матриц [6], однако при этом вопросы распараллеливания требуют специальных исследований.

СПИСОК ЛИТЕРАТУРЫ

1. **Писсанецки С.** Технология разреженных матриц. М.: Мир, 1988.
2. **Eisenstat S. C., Elman H. C., Schultz M. H.** Variational iterative methods for nonsymmetric systems of linear equations // SIAM Journ. Numer. Anal. 1983. **20**, N 3. P. 345.
3. **Saad Y.** Iterative Methods for Sparse Linear Systems. N. Y.: PWS Publishing, 1996.
4. **Ильин В. П.** Численный анализ. Ч. 1. Новосибирск: Изд-во ИВМиМГ СО РАН, 2004.
5. **Andreeva M. Yu., Il'in V. P., Itskovich E. A.** Two solvers for nonsymmetric SLAE // Bull. of the Novosibirsk Computing Center. Ser. Numer. Anal. 2004. N 12. P. 1.
6. **Ильин В. П., Пудов С. Г.** Методы неполной факторизации с полусопряженными невязками // Автометрия. 2007. **43**, № 2. С. 66.

Поступила в редакцию 3 ноября 2006 г.