第 38 卷 第 8 期
2001 年 8 月

计 算 机 研 究 与 发 展
JOURNAL OF COMPUTER RESEARCH & DEVELOPMENT

Vol. 38, No. 8
Aug. 2001

Cholesky 分解递归算法与改进

陈建平[①] Jerzy Wasniewski[②]

[①](南通工学院信息工程系 南通 226007)

[②](丹麦研究与教育计算中心 哥本哈根 DK-2800)

(chen.jp@ntit.edu.cn)

摘 要 递归算法是计算稠密线性代数的一种新的有效方法. 递归产生自动、变化的矩阵分块, 能充分发挥当今分级存储高性能计算机的效率. 对 Cholesky 分解递归算法进行了研究, 给出了算法的详细推导过程, 用具有递归功能的 Fortran 90 实现了算法, 并通过矩阵元素顺序重排的方法, 进一步提高了递归算法的运算速度. 研究产生的算法比目前常用的分块算法快 15%～25%.

关键词 数值计算, 递归, 矩阵分块, 分级存储, Fortran 90

中图法分类号 TP301

RECURSIVE ALGORITHM AND IMPROVEMENT FOR CHOLESKY FACTORIZATION

CHEN Jian-Ping[①] and Jerzy Wasniewski[②]

[①](*Department of Information Engineering, Nantong Institute of Technology, Nantong*, 226007)

[②](*Danish Computing Center for Research and Education, Lyngby, Denmark, DK*-2800)

http://www.ixueshu.com

Abstract Recursion is a new effective method for computing dense linear algebra. It allows for efficient utilization of memory hierarchies of today's high-performance computers. The recursive algorithm for Cholesky factorization is studied in this paper. A detailed derivation of the recursive Cholesky algorithm is given. The algorithm is then implemented in Fortran 90 that supports recursion as a language feature. The efficiency of the recursive algorithm is further improved by using a method of matrix element reordering. The resulting algorithms are 15%～25% faster than the currently used block algorithm.

Key words numerical computation, recursion, matrix blocking, memory hierarchy, Fortran 90

1 引 言

以 Cholesky 分解、LU 分解等为代表的线性代数问题的数值计算在现代科学研究和工程技术中得到广泛应用. 随着计算机结构和技术的发展, 实现这些线性代数数值计算的计算机算法和软件也在不断发展. 通用的基本线性代数子程序库 BLAS (basic linear algebra subprograms) 从 70 年代的 Level-1 BLAS (执行向量-向量运算), 发展到 80 年代的 Level-2 BLAS (执行矩阵-向量运算), 再到 90 年代的 Level-3 BLAS (执行矩阵-矩阵运算)[1]. 以调用 BLAS 为核心运算的线性代数软件包也相应地从早期的 LINPACK 发展到现在的 LAPACK (linear algebra package)[2].

目前, 超标量、超流水线、具有多级存储结构的高性能 RISC 计算机已占据了数值计算领域的主导地位. RISC 处理器的运算速度非常快, 它们与存储

原稿收到日期: 2000-10-10; 修改稿收到日期: 2001-03-28
本课题得到江苏省教育厅留学回国人员科研启动经费项目基金资助

器之间的速度差距很大 计算机的性能能不能充分发挥, 多级存储结构即高缓(cashe)能否得到有效利用 成为关键 为此, 现行的线性代数算法(如LAPACK)通常采用分块(block)算法 通过将矩阵分块, 使各部分子矩阵的运算能够在高缓中进行, 同时尽可能地调用Level-3 BLAS, 以提高运算效率 作为一种新的线性代数的计算方法, 文献[3]提出了递归(recursive)算法 递归具有自动矩阵分块的功能, 产生的分块子矩阵的阶数逐级减小并为方阵, 带来良好的数据局部性(locality), 使得递归算法非常适合当今分层多级存储的计算机结构 同时, 递归算法结构简单, 易于实现

本文对Cholesky 分解递归算法进行了深入研究, 给出了算法的详细推导过程, 用支持递归功能的Fortran 90 语言实现了算法, 并通过矩阵元素顺序的重排进一步提高了递归算法的效率 研究产生的算法和程序在PowerPC SMP 计算机上进行了测试, 其运算速度在大矩阵情况下比LAPACK 分块算法提高15%～25%.

2 Cholesky 分解递归算法

Cholesky 分解用于求解系数矩阵具有对称正定性的线性方程组

$$AX = B, \tag{1}$$

式(1)中, A 为实对称正定矩阵, X 和B 为矩形矩阵或向量 Cholesky 分解法只需用到矩阵A 的上三角部分或下三角部分元素 若给定上三角部分, A 可分解成

$$A = U^{T}U, \tag{2}$$

若给定下三角部分, A 可分解成

$$A = LL^{T}, \tag{3}$$

U 和L 分别为上三角矩阵和下三角矩阵($L = U^{T}$). 分解后的矩阵A 用于求解方程$AX = B$.

本文以上三角情形即式(2)为例讨论Cholesky 分解递归算法 给定$n \times n$ 阶对称正定矩阵A, 计算上三角矩阵U. 设A 的元素用a_{ij}表示, U 的元素用u_{ij}表示, 则由Cholesky 分解法[4], 有

$$u_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} u_{ki}^{2}\right)^{1/2}, \quad i = 1, 2, \cdots, n, \tag{4}$$

$$u_{ij} = \left(a_{ij} - \sum_{k=1}^{i-1} u_{ki}u_{kj}\right) \Big/ u_{ii},$$

$$j = i + 1,\ i + 2, \cdots, n, \tag{5}$$

现将矩阵A 和U 写成分块形式:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ & A_{22} \end{pmatrix} \text{ 和 } U = \begin{pmatrix} U_{11} & U_{12} \\ & U_{22} \end{pmatrix}, \tag{6}$$

子矩阵A_{11}和U_{11}的阶数为$p \times p$, A_{12}和U_{12}的阶数为$p \times (n - p)$, A_{22}和U_{22}的阶数为$(n - p) \times (n - p)$, 这里, $p = n/2$ 子矩阵中, $A_{11}, U_{11}, A_{22}, U_{22}$为上三角阵, A_{12}, U_{12}为矩形阵 将式(6)代入式(2)得到

$$\begin{pmatrix} A_{11} & A_{12} \\ & A_{22} \end{pmatrix} = \begin{pmatrix} U_{11}^{T} & \\ U_{12}^{T} & U_{22}^{T} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ & U_{22} \end{pmatrix}, \tag{7}$$

计算式(7)可得

$$A_{11} = U_{11}^{T}U_{11}, \tag{8}$$

$$A_{12} = U_{11}^{T}U_{12}, \tag{9}$$

$$A_{22} = U_{12}^{T}U_{12} + U_{22}^{T}U_{22}, \tag{10}$$

式(8)为对子矩阵A_{11}进行Cholesky 分解, 由式(4)、(5)可计算出U_{11}. 在解得U_{11}(亦即U_{11}^{T})后, 由式(9)计算U_{12}. 式(9)中, 系数矩阵U_{11}^{T}是三角阵, 可以调用Level-3 BLAS 程序TRSM 求解 然后通过式(10)计算U_{22}. 将式(10)改写成

$$A_{22} - U_{12}^{T}U_{12} = U_{22}^{T}U_{22}, \tag{11}$$

令

$$\overline{A_{22}} = A_{22} - U_{12}^{T}U_{12}, \tag{12}$$

则式(11)成为

$$\overline{A_{22}} = U_{22}^{T}U_{22}, \tag{13}$$

式(12)的含义为通过U_{12}对A_{22}进行修改或更新, 可利用Level-3 BLAS 程序SYRK 来计算 更新后的子矩阵A_{22}仍为上三角阵(对称正定). 式(13)为对更新后的A_{22}进行Cholesky 分解, 由式(4)、(5)计算出U_{22}.

这样, 原来$n \times n$ 阶矩阵A 的Cholesky 分解运算就转化成大小为$(n/2) \times (n/2)$的分块子矩阵A_{11}, U_{11}, A_{12}, U_{12}和A_{22}的运算 下一步对式(8)和式(13)中子矩阵A_{11}和$\overline{A_{22}}$的Cholesky 分解可以进行同样的处理, 将它们分解成大小为$(n/4) \times (n/4)$的子矩阵的运算 这种过程可以一直重复下去, 直到最终产生的子矩阵A_{11}等足够小, 或者更简单地, 直到A_{11}的阶数为1. 此时A_{11}只有一个元素, 其Cholesky 分解为$U_{11} = A_{11}^{1/2}$. 事实上, 最终的A_{11}的阶数P 的选取与实际使用的计算机及其高缓容量有关 本文为使讨论具有一般性和简单起见, 取$P = 1$. 于是, Cholesky 分解的递归算法可以描述为:

Cholesky 分解$A = U^{T}U$,

如果$n > 1$, 则

$p = n/2$

Cholesky 分解$A_{11} = U_{11}^{T}U_{11}$,

解$U_{11}^{T}U_{12} = A_{12}$ (调用Level-3 BLAS

http://www.ixueshu.com

TRSM),

计算 $\overline{A_{22}} = A_{22} - U_{12}^{T}U_{12}$ (调用 Level-3 BLAS SYRK).

Cholesky 分解 $\overline{A_{22}} = U_{22}^{T}U_{22}$,

否则

$U = A^{1/2}$.

3　递归算法的 Fortran 90 实现

Fortran 77 不具备递归功能, 若用其实现递归算法, 递归过程中递归的级数、分块子矩阵的位置、阶数等参数的堆栈管理必须通过程序语句实现, 比较复杂. Fortran 90 支持递归过程[5], 递归自动地由编译器处理, 非常便于递归算法的实现, 产生的程序简单、高效. 下面给出 Cholesky 分解递归算法 Fortran 90 程序的主要语句.

```
RECURSIVE SUBROUTINE RPOTRF(N,A,LDA)
  USE F90_RCF, ONLY: RCF => RPOTRF
  IMPLICIT NONE
  INTEGER, INTENT(IN) :: N, LDA
  DOUBLE PRECISION, INTENT(INOUT) :: A
           (LDA, *)
  INTEGER :: P
  DOUBLE PRECISION, PARAMETER :: ONE=
           1.D0
  IF (N.EQ. 1) THEN
    A(1,1)=SQRT(A(1,1))
  ELSE
    P=N/2
    CALL RCF(P,A,LDA)
    CALL DTRSM('L','U','T','N',P,N-P,
      ONE,A(1,1),LDA,A(1,P+1),LDA)
    CALL DSYRK('U','T',N-P,P,-ONE,
      A(1,P+1),LDA,ONE,A(P+1,P+1),
      LDA)
    CALL RCF(N-P,A(P+1,P+1),LDA)
  END IF
END SUBROUTINE RPOTRF
```

程序中第 1 句 RECURSIVE SUBROUTINE *RPOTRF* 说明子程序 *RPOTRF* 为递归型子程序, 即可以自己调用自己. 第 2 句中的 *RCF*⇒*RPOTRF* 为指针说明, 表示 *RCF* 与 *RPOTRF* 等同. 在此说明下, 第 12 行及第 15 行的 CALL *RCF* 即为调用 *RPOTRF* 即该递归子程序本身. 第 13, 14 行为调用 Level-3 BLAS TRSM 和 SYRK. 本程序以双精度情形为例, 故为 *DTRSM* 和 *DSYRK*. 在递归调用 *RCF*, *TRSM* 和 *SYRK* 的过程中, 参数 *LDA* (矩阵 A 的 leading dimension) 保持不变. 只要给出每个分块子矩阵第 1 个元素的位置, 就可以根据该 *LDA* 访问各个分块子矩阵的元素. 这样, 在每一次调用 *RCF*, *TRSM* 或 *SYRK* 对分块子矩阵进行计算时, 就不需要对有关子矩阵的元素进行拷贝, 节省了内存的使用. 算法的全部运算均通过调用 Level-3 BLAS (*TRSM* 和 *SYRK*) 完成, 保证了算法在现代高性能计算机上的高效运行.

4　矩阵元素的顺序重排

上一节讨论的程序在递归计算 *TRSM* 和 *SYRK* 的过程中, 使用的是固定的 *LDA* 参数来访问分块子矩阵的元素. 尽管随着递归运算的进行, 需要计算的子矩阵的尺寸越来越小, 但存储器访问的跨距 (memory stride) 保持不变并且很大 (等于矩阵的阶数). 如果我们将递归产生的分块子矩阵的元素重新排列, 使各个子矩阵的元素依次集中放在一起, 那么计算时随着递归的进行, 存储器访问的跨距就会越来越小, 从而能减小存储器访问时间, 提高运算速度.

下面以 8 阶矩阵为例, 说明矩阵元素重排的过程. 为了方便起见, 用顺序数字代表矩阵元素, 虚线表示逐级递归分解产生的分块. 输入矩阵为

1	9	17	25	33	41	49	57
2	10	18	26	34	42	50	58
3	11	19	27	35	43	51	59
4	12	20	28	36	44	52	60
5	13	21	29	37	45	53	61
6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63
8	16	24	32	40	48	56	64

第 1 级递归分解, 矩阵元素重排后的顺序为

1	17	5	21	33	49	37	53
2	18	6	22	34	50	38	54
3	19	7	23	35	51	39	55
4	20	8	24	36	52	40	56
9	25	13	29	41	57	45	61
10	26	14	30	42	58	46	62
11	27	15	31	43	59	47	63
12	28	16	32	44	60	48	64

第 2 级递归分解, 矩阵元素的重排顺序为

http://www.ixueshu.com

1	17	5	21	33	49	37	53
2	18	6	22	34	50	38	54
9	25	7	23	35	51	45	61
10	26	8	24	36	52	46	62
3	19	13	29	41	57	39	55
4	20	14	30	42	58	40	56
11	27	15	31	43	59	47	63
12	28	16	32	44	60	48	64

第 3 级即最后一级递归分解将 2×2 阶的子矩阵分解成 1×1 阶的分块, 矩阵元素的重排顺序与前一级相同

在重排后的数据格式下, 递归产生的每个分块子矩阵的元素均以按列存放的方式连续放在一起 给定每个子矩阵第一个元素的地址, 则子矩阵中的元素就可以使用大小等于该子矩阵尺寸的 LD 参数来访问 由于每一级递归分解子矩阵的尺寸减小一半, 使得 LD 参数即存储器访问的跨距越来越小, 从而减少存储器访问时间 同时, 各个子矩阵的元素集中连续放在一起, 在递归计算时, 参与运算的有关子矩阵的元素更容易一起同时全部进入高缓, 提高了高缓的使用效率

5 运行结果

我们在 IBM PowerPC SMP 计算机上运行测试了上述 Cholesky 分解递归算法和程序. 表 1 列出了不同矩阵阶数下它们的运算时间 其中, *RPOTRF* 为第 3 节中的 Fortran 90 程序, *RPOTRF*1 表示第 4 节元素重排后的算法和程序. 作为比较, 同时运行了 Cholesky 分解分块算法 LAPACK 程序 *DPOTRF* 以及 IBM ESSL 程序 *DPOF*.

表 1 RPOTRF 与 DPOTRF 及 DPOF 的运算速度比较 /s

阶数	算法			
	RPOTRF	*RPOTRF*1	*DPOTRF*	*DPOF*
500	0.17	0.10	0.18	0.19
1000	1.30	1.16	1.32	1.40
1500	3.87	3.68	3.98	4.54
2000	10.03	9.63	10.99	11.08
2500	18.12	16.99	19.22	20.97
3000	33.38	30.40	39.70	35.86

由表 1 可见, 递归算法 *RPOTRF* 优于 LAPACK *DPOTRF*. 这种优势在矩阵阶数较大时更加明显, 如 $n=3000$ 时, *RPOTRF* 比 *DPOTRF* 快 16%. IBM ESSL (engineering and scientific subroutine library) 程序 *DPOF* 对大矩阵计算作了专门设计[6], 效率很高 既使如此, *RPOTRF* 仍快于 *DPOF*. 在 $n=3000$ 时, *RPOTRF* 比 *DPOF* 快 7%. 表 1 还表明, 矩阵元素顺序重排后的递归算法运算速度进一步提高, 在 $n=3000$ 时, *RPOTRF*1 比 *RPOTRF* 又快了 9%.

6 结束语

本文通过对 Cholesky 分解递归算法的研究表明, 递归是计算稠密线性方程组的有效方法, 非常适合当今多级存储高性能计算机的结构 除了 Cholesky 分解以外, 递归算法同样适用于 LU 分解, QR 分解等其它线性代数问题 事实上, BLAS 库中的基本单元 (如 *TRSM*, *SYRK* 和 *GEMM* 等) 也可以采用递归的方法实现 在递归算法基础上, 对矩阵元素的排列方式进行调整, 将每一级递归产生的各个分块子矩阵的元素以标准的按列存放 (或按行存放) 的顺序依次连续放在一起, 可以进一步提高递归算法的效率

参 考 文 献

1 Dongarra J *et al*. A set of level 3 basic linear algebra subprograms. ACM Trans on Mathematical Software, 1990, 16(1): 1~17

2 Anderson E *et al*. LAPACK Users' Guide. 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics, 1995

3 Gustavson F. Recursion leads to automatic variable blocking for dense linear algebra. IBM Journal of Research and Development, 1997, 41(6): 737~755

4 Demmel J W. Applied Numerical Linear Algebra. Philadelphia: Society for Industrial and Applied Mathematics, 1997

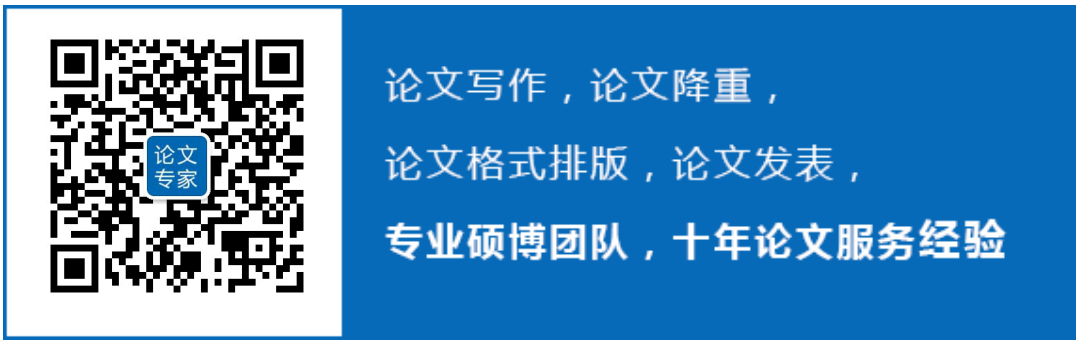
5 Metcalf S, Reid J. FORTRAN 90/95 Explained. Oxford: Oxford University Press, 1996

6 Engineering and Scientific Subroutine Library, Guide and Reference. New York: IBM, 1994

陈建平 男, 1960 年生, 副教授, 主要研究方向为快速算法、数字信号处理 曾先后在美国宾西法尼亚大学和丹麦研究与教育计算中心进修和合作研究, 已在国内外发表论著十多篇 (部).

Jerzy Wasniewski 男, 1936 年生, 首席计算机科学家 (principal computer scientist), 主要研究方向为计算机数值计算及应用、并行计算 多年来与美、英等国高校以及 IBM、NAG 等科研机构合作, 从事数值计算理论、应用与软件的研究工作, 发表论著数十篇 (部), 近期的成果有 LAPACK90.

http://www.ixueshu.com



免费论文查重：http://free.paperyy.com
3亿免费文献下载：http://www.ixueshu.com
超值论文自动降重：http://www.paperyy.com/reduce_repetition
PPT免费模版下载：http://ppt.ixueshu.com

阅读此文的还阅读了：

1. 递归算法在程序设计中的应用
2. 集群下Cholesky分解的数据重用算法
3. 离散正弦变换—II的一种快速递归算法
4. 递归函数的非递归模拟
5. C语言程序交互式虚拟算法动画的开发与教学应用
6. 浅析程序设计中的递归算法
7. Cholesky分解细粒度并行算法
8. 递归算法的非递归化实现
9. 程序设计的有力手段--递归技术
10. 无线正交频分复用系统信道序惯盲检测
11. 分治算法的两种思路和形式
12. 矢量数据压缩的Douglas-Peucker算法的实现与改进
13. 循环赛日程表的递归和非递解
14. 矩阵链乘积最优计算次序问题的算法及其复杂性分析
15. 基于链栈数组的二叉树按层遍历递归算法
16. 基于树型数据库的岗位能力管理系统的设计与实现

17. 递归程序的非递归化算法

18. 流水作业调度问题的算法研究

19. C语言中递归的探讨

20. 程序设计中的递归算法分析

21. 反均值问题的一种递归算法

22. 关于递归与递推的讨论

23. 分治算法在循环赛赛程分配中的应用

24. 浅析《数据结构》教学关键点

25. 基于分治思想的二分搜索技术研究

26. 无返回地址的递归消除方法研究

27. 数据结构中递归算法的描述与实现

28. 堆栈在递归调用中的应用

29. 浅谈数据结构教学中递归算法的描述与实现

30. 数据库算法系列讲座 (二)

31. 栈的数据结构与递归算法

32. 奇妙的“倒三角形”

33. 基于MATLAB的神秘数算法设计与实现

34. 集群下Cholesky分解的核外预取算法

35. 矩阵链乘积最优计算次序问题的算法及其复杂性分析

36. 循环赛日程表的递归和非递解

37. 分治算法的两种思路和形式

38. 递归算法的参数设置

39. 一种Cholesky分解重叠算法

40. 一种基于数组的递归算法

41. 递归算法的非递归化策略

42. 汉诺塔问题的计算机辅助教学

43. 一种基于数组的递归算法

44. 浅析C语言递归算法

45. 改进的快速排序算法与递归

46. 递归的非递归算法研究

47. 递归算法应用分析

48. 基础算法的“一二三四五”例析

49. 数据库算法系列讲座(二)

50. 递归在程序设计中的应用