乔莱斯基分解递归算法的研究*)

Study of Recursive Algorithm for Cholesky Factorization

陈建平　　Jerzy Wasniewski

(南通工学院　江苏南通226007)　(丹麦研究与教育计算中心)

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 FORTRAN90 that supports recursion as a language feature. The efficiency of the recursive algorithm is further improved by a method of matrix element reordering. The resulting algorithms are 15%-25% faster than the currently used block algorithm.

Keywords Numerical computation, Recursion, Matrix blocking, Memory hierarchy, FORTRAN90

1 引言

以乔莱斯基(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，RSIC 处理器的运算速度非常快，它们与存储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[3]提出了递归算法。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，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。

本文对乔莱斯基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 FORTRAN90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 乔莱斯基分解递归算法

乔莱斯基分解用于求解系数矩阵具有对称正定性的线性方程组

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

式中，A 为实对称正定矩阵，X 和 B 为矩形矩阵或向量。乔莱斯基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

$$A=U^TU \tag{2}$$

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 然后用于求解方程 $AX=B$。

本文以上三角情形即式(2)为例讨论乔莱斯基分解递归算法。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，则由乔莱斯基分解法，有

*)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资助。陈建平　副教授，主要研究方向为快速算法、数字信号处理，Jerzy Wasniewski　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。

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

$$u_{ij}=(a_{ij}-\sum_{k=1}^{i-1}u_{ki}u_{kj})/u_{ii} \quad j=i+1,i+2,\cdots,n \tag{5}$$

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

$$A_{11}=U_{11}^TU_{11} \tag{8}$$

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

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

式(8)为对子矩阵 A_{11} 进行乔莱斯基分解，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：

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

令

$$\overline{A}_{22}=A_{22}-U_{12}^TU_{12} \tag{12}$$

则式(11)成为

$$\overline{A}_{22}=U_{22}^TU_{22} \tag{13}$$

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 $\overline{A}_{22}$ 仍为上三角阵(对称正定)。式(13)为对更新后的 $\overline{A}_{22}$ 进行乔莱斯基分解，由式(4)、(5)计算出 U_{22}。

这样，原来 $n\times n$ 阶矩阵 A 的乔莱斯基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 $\overline{A}_{22}$ 的乔莱斯基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，其乔莱斯基分解为 $U_{11}=A_{11}^{1/2}$。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，乔莱斯基分解的递归算法可以描述为：

```
乔莱斯基分解 A=U^TU
如果 n>1，则
    p=n/2
    乔莱斯基分解 A11=U11^TU11
    解 U11^TU12=A12(调用 Level-3 BLAS TRSM)
    计算 Ā22=A22-U12^TU12(调用 Level-3 BLAS SYRK)
    乔莱斯基分解 Ā22=U22^TU22
否则
    U=A^1/2
```

3 递归算法的 FORTRAN90实现

FORTRAN77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。FORTRAN90支持递归过程，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。下面给出乔莱斯基分解递归算法 FORTRAN90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.DO
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,-ONE,A(1,P+1),
    LDA,ONE,A(P+1,P+1),LDA)
  CALL RCF(N-P,A(P+1,P+1),LDA)
 ENDIF
END SUBROUTINE RPOTRF
```

程序中第一句 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，第二句中的 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一个元素的位置，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

第一级递归分解,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

第二级递归分解,矩阵元素的重排顺序为

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

第三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,从而可以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乔莱斯基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FORTRAN90程序,RPOTRF1表示第4节元素重排后的算法和程序。作为比较,同时运行了乔莱斯基分解分块算法LAPACK程序DPOTRF以及IBMESSL程序DPOF。

表1 RPOTRF与DPOTRF及DPOF的运算速度比较(时间单位:S)

算法 \ 阶数	500	1000	1500	2000	2500	3000
RPOTRF	0.17	1.30	3.87	10.03	18.12	33.38
RPOTRF1	0.10	1.16	3.68	9.63	16.99	30.40
DPOTRF	0.18	1.32	3.98	10.99	19.22	39.70
DPOF	0.19	1.40	4.54	11.08	20.97	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[4],效率很高。即使如此,RPOTRF仍快于DPOF。在n=3000时,PROTRF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,RPOTRF1比RPOTRF又快了9%。

结束语 本文通过对乔莱斯基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。除了乔莱斯基分解以外,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 Math. Softw., 1990, 16(1):1~17
2 Anderson E, et al. LAPACK Users'Guide, Second Edition. Philadelphia:SIAM, 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 IBM. Engineering and Scientific Subroutine Library, Guide and Reference, 1994. SC23-0526-10
5 Demmel J W. Applied Numerical Linear Algebra. Philadelphia:SIAM, 1997
6 Metcalf S, Reid J. FORTRAN90/95 Explained. Oxford: Oxford University Press, 1996