LU 分解递归算法的研究*)

陈建平
（南通工学院信息工程系　南通226007）

摘　要　将递归方法引入稠密线性代数的计算，能产生自动的矩阵分块，使算法适合于当今分级存储高性能计算机的结构，提高运算速度。文中对解线性代数方程组的LU分解递归算法进行了研究，给出了算法的详细推导过程。

关键词　数值计算，矩阵分解，递归

Study of Recursive Algorithm for LU Factorization

CHEN Jian-Ping
(Department of Information Engineering, Nantong Institute of Technology, Nantong 226007)

Abstract　Recursion leads to automatic matrix blocking in the computation of dense linear algebra. It makes a good use of memory hierarchies of today's high-performance computers and hence improves the efficiency of the algorithm. The recursive algorithm for LU factorization of a matrix that is used to solve linear systems of equations is studied in this paper. A detailed derivation of the recursive algorithm is presented.

Keywords　Numerical analysis, Matrix factorization, Recursion

1　引言

目前，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）通常采用分块算法[1,2]。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，以提高运算速度。将递归引入线性代数的计算，是一种新的方法和改进[3,4]，递归算法自动产生矩阵分块，非常适合当今分层多级存储的计算机结构。本文对矩阵运算的 LU 分解递归算法进行了研究，给出了算法的详细推导过程。

2　矩阵的 LU 分解

LU 分解用于求解线性方程组 $AX=b$，若系数矩阵 A 非奇异，则 A 可以分解为一个下三角矩阵 L 和一个上三角矩阵 U 的乘积

$$A=LU \tag{1}$$

或

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{pmatrix} \tag{2}$$

由 Doolittle 分解，三角阵 U 和 L 的元素如下计算

$$\begin{cases} u_{1j}=a_{1j}(j=1,2,\cdots,n) \\ l_{i1}=a_{i1}/u_{11}(i=2,3,\cdots,n) \\ u_{kj}=a_{kj}-\sum\limits_{m=1}^{k-1} l_{km}u_{mj}(j=k,k+1,\cdots,n) \\ l_{ik}=(a_{ik}-\sum\limits_{m=1}^{k-1} l_{im}u_{mk})/u_{kk}(i=k+1,k+2,\cdots,n) \\ \qquad (k=2,3,\cdots,n) \end{cases} \tag{3}$$

分解后的 L、U 的元素 l_{ik} 和 u_{kj} 可放回 A 的相应元素 a_{ik} 和 a_{kj} 所在的位置，不需占用新的存储单元，即最终有

$$A=\begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ l_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & u_{nn} \end{pmatrix} \tag{4}$$

在上述基础上，下面推导 LU 分解的递归算法。

3　LU 分解递归算法

3.1　递归分解过程

不失一般性，设矩阵 A 的阶数为 $m\times n$，且 $m\geqslant n$。令 $d=n/2$，将式(2)写成分块形式

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ & U_{22} \end{pmatrix} \tag{5}$$

式中，分块子矩阵 A_{11}、L_{11} 和 U_{11} 的阶数为 $d\times d$，A_{12} 和 U_{12} 的阶数为 $d\times(n-d)$，A_{21} 和 L_{21} 的阶数为 $(m-d)\times d$，A_{22} 和 L_{22} 的阶数为 $(m-d)\times(n-d)$，U_{22} 的阶数为 $(n-d)\times(n-d)$。子矩阵中，L_{11}、L_{22} 为下三角阵，U_{11}、U_{22} 为上三角阵，其余为矩形阵。计算式(5)可得

$$A_{11}=L_{11}U_{11} \tag{6}$$

$$A_{21}=L_{21}U_{11} \tag{7}$$

$$A_{12}=L_{11}U_{12} \tag{8}$$

$$A_{22}=L_{21}U_{12}+L_{22}U_{22} \tag{9}$$

*）本文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　教授，主要研究方向为快速算法、数字信号处理。

万方数据

式(6)和式(7)可合并写成

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} (U_{11}) \tag{10}$$

其含义是对$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$进行 LU 分解,解出 L_{11}、L_{21}和 U_{11}后,由式(8)计算 U_{12}

$$U_{12}=L_{11}^{-1}A_{12} \tag{11}$$

令

$$A'_{22}=A_{22}-L_{21}U_{12} \tag{12}$$

则式(9)成为

$$A'_{22}=L_{22}U_{22} \tag{13}$$

式(12)的含义为通过 L_{21}和 U_{12}(已解出)对 A_{22}进行修改或更新。式(13)为对更新后的 A'_{22}进行 LU 分解。

这样,矩阵 A 的 LU 分解运算就转化成分块子矩阵$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$和 A'_{22}的 LU 分解以及 U_{12}的求解和 A_{22}的更新计算。以上完成了一级分解,下一步对子矩阵$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$和 A'_{22}的 LU 分解可以进行同样的处理,将它们各自分解成列数约为$(n/4)$的子矩阵的运算。这种递归分解过程可以一直重复下去,直到最终产生的子矩阵 $A_{11}(A_{21})$和 A'_{22}的列数足够小,或者更一般地,直到它们的列数为1。此时$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$和 A'_{22}为列向量,其 LU 分解由式(3)中的前两式即可求得。于是,LU 分解递归算法可以描述为

LU 分解 A:$A=LU$(递归过程)

如果 $n>1$,则

$d=n/2$

LU 分解$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$:$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}=\begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix}(U_{11})$(递归调用)

解 U_{12}:$U_{12}=L_{11}^{-1}A_{12}$

计算 A'_{22}:$A'_{22}=A_{22}-L_{21}U_{12}$

LU 分解 A'_{22}:$A'_{22}=L_{22}U_{22}$(递归调用)

否则($n=1$)

$u_{11}=a_{11}$

$l_{i1}=a_{i1}/u_{11}(i=2,3,\cdots,m)$

图1以4×4阶矩阵为例,用图形显示出递归分解的过程。

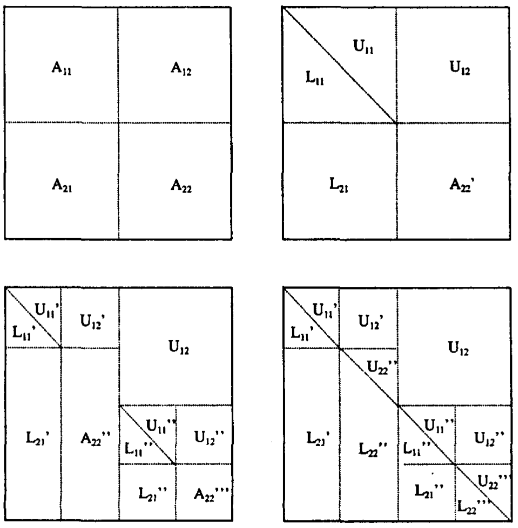

图1 4×4阶矩阵 LU 分解递归算法的分解过程

3.2 增加选主元的步骤

为了避免式(3)中的除数为零或绝对值相对很小,LU 分解在实际应用中需要增加选主元的步骤。因此,在上述 LU 分解递归算法中,也需要加入选主元的过程。通常采用部分选主元,即选取列主元,进行行交换的方法。对矩阵 A 进行一系列行交换,相当于用一个排列矩阵 P 左乘矩阵 A,然后再对 PA 进行 LU 分解。PA 的 LU 分解递归转化成 $P_1\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$和 $P_2A'_{22}$的 LU 分解。P_1和 P_2分别为对$\begin{pmatrix} A_{11} \\ A_{22} \end{pmatrix}$和 A'_{22}进行选主元,作行交换形成的排列矩阵。在解得 $P_1\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$后,需要用 P_1对$\begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix}$进行同样的换行,然后再对$\begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix}$进行计算和分解。同样,在解得 $P_2A'_{22}$后,需要用 P_2对 $A_{21}(L_{21})$进行同样的换行以得到 $A_{21}(L_{21})$中元素的正确位置。这样,最终形成的带有部分选主元的 LU 分解递归算法的描述为

LU 分解 PA:$PA=LU$(递归过程)

如果 $n>1$,则

$d=n/2$

LU 分解 $P_1\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$:$P_1\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}=\begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix}(U_{11})$(递归调用)

$\begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix}$换行:$\begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix}=P_1\begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix}$

解 U_{12}:$U_{12}=L_{11}^{-1}A_{12}$

计算 A'_{22}:$A'_{22}=A_{22}-L_{21}U_{12}$

LU 分解 $P_2A'_{22}$:$P_2A'_{22}=L_{22}U_{22}$(递归调用)

A_{21}换行:$A_{21}=P_2A_{21}$

否则($n=1$)

找出$(a_{11},a_{21},\cdots,a_{m1})$中的主元,设为 a_{x1}

将 a_{11}和 a_{x1}换位(同时产生排列矩阵 P),然后计算

$u_{11}=a_{11}$

$l_{i1}=a_{i1}/u_{11}(i=2,3,\cdots,m)$

4 算法的实现

上节导出的 LU 分解递归算法可通过 FORTRAN90语言在计算机上实现。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。算法中的核心运算为求解 U_{12}(即式(8)的计算)和更新 A_{22}(式(12)的计算),可以分别调用高效率的 Level-3 BLAS 程序 TRSM 和 GEMM。算法中通过 P_1和 P_2对$\begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix}$和 A_{21}的换行操作可调用 LAPACK 程序 LASWP。最终找主元以及各元素除以主元的操作可通过 BLAS 程序 IAMAX 和 SCAL 来完成。有关实现程序和算法的运算测试结果将在以后的文章中介绍。

参考文献

1 Anderson E, et al. LAPACK Users' Guide, Second Edition. Philadelphia:SIAM,1995

2 Dongarra J, et al. A Set of Level 3 Basic Linear Algebra Subprograms. ACM Trans. on Math. Softw., 1990,16(1):1~17

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 陈建平,Wasniewski J. Cholesky 分解递归算法与改进. 计算机研究与发展,2001,38(8):923~926

5 Burden R L, Faires J D. Numerical Analysis (Seventh Edition). Thomson Learning, Inc. 北京:高等教育出版社(影印版),2001

6 关治,陈景良. 数值计算方法. 北京:清华大学出版社,1990

7 封建湖,等. 数值分析原理. 北京:科学出版社,2001

万方数据