矩阵的 Doolittle 递归分解算法及符号程序设计

智东杰[1] ,智慧来[2]
(1. 河南理工大学 计算机学院 ,河南 焦作 454150 ;2. 西华大学 能源与环境学院 ,四川 成都 610039)

摘 要:将矩阵 $A_{n\times n}$ 的 Doolittle 分解推广到 $A_{m\times n}$ 上 ,并在常规的迭代算法上加以创新 ,给出了递归的分解算法. 在实现算法的过程中 ,对数据进行了巧妙处理 ,使中间数据及最终计算结果都具有分数形式 ,提高了结果的精确度 ,而且更符合人们阅读的习惯. 经过运行测试 ,算法设计合理 ,程序运行高效准确. 程序是对 MathSoft 公司的交互式的数学文字软件 Mathcad 的矩阵分解的数值计算扩充到符号运算.

关键词:矩阵 ;Doolittle 分解 ;算法
中图分类号: TP311. 1 **文献标识码**: A **文章编号**: 1673-4785(2007) 01-0090-04

Recursive algorithm and symbolic programming for matrix Doolittle 's factorization

ZHI Dong-jie[1] , ZHI Hui-lai[2]
(1. School of Computer Science & Technology , Henan Polytechnic University , Jiaozuo 454150 , China ; 2. School of Energy and Environment , Xihua University , Chengdu 610039 , China)

Abstract: Apply Doolittle 's factorization of matrix $A_{n\times n}$ to matrix $A_{m\times n}$ that has m rows and n columns , and conceive a new recursive algorithm of Doolittle 's factorization of matrix on the basis of traditional iterative algorithm. In the realization of algorithm , the data is skillfully dealt , so the interim data and the result have the form of fraction. These merits improve the result 's exactness. And above that it conforms to readers 'reading habit compared to double or float forms. After running it , the result proves that the algorithm is reasonably designed and the program is running with high proficiency and exactness. The program is an augmentation of matrix factorization in the MathSoft 's interactive mathematic language software , which promotes matrix factorization from numerical computation to symbolic computation.

Key words: matrix ; Doolittle 's factorization ; algorithm

将矩阵分解为形式比较简单或具有某种特性的一些矩阵的乘积 ,在矩阵理论的研究与应用中都显得十分重要. 文献[1]对通用的基本线性代数子程序库(basic linear algebra subprograms ,BLAS) ,从 20 世纪 70 年代的 Level-1 的 BLAS 开始到 Level-3 BLAS ,LAPACK(linear algebra package) 的发展过程进行了详细介绍 ,并对矩阵的 Cholesky 分解进行研究和改进 ,给出递归算法及 Fortran90 程序. 文献[2]对 m 行 n 列($m > n$) 矩阵的 LU 分解进行了研究;文献[3]给出了 Crout 方法在解线性方程组中的应用. 文献[4]给出了 Crout 递归分解算法及实现. 文中对任意矩阵用分块的思想进行 Doolittle 分解的研究 ,给出更简单的递归分解算法及程序实现.

收稿日期:2006-04-27.

1 矩阵 Doolittle 分解的定义

定义 设 $A \in C^{n\times n}$. 如果 A 可分解成 $A = LR$,其中 L 是对角元素为 1 的下三角矩阵(称为单位下三角矩阵) , R 是上三角矩阵 ,则称之为 A 的 Doolittle 分解[5].

根据定义获得的计算公式

$$r_{1j} = a_{1j}\ (j = 1,2,\cdots,n)\ ,$$

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

$$r_{kj} = a_{kj} - \sum_{t=1}^{k-1} l_{kt} * r_{tj}\ (j = k, k+1,\cdots,n; k = 2,$$

3, …, n),

$$l_{ik} = 1/r_{kk} * \left(a_{ik} - \sum_{t=1}^{k-1} l_{it} * r_{tk} \right) \quad (i = k+1, \cdots, n; k = 2, 3, \cdots, n).$$

由定义知,Doolittle 分解是在 n 行 n 列的矩阵上进行的,由公式知道当 i, j, k 较大时,l_{ik}, r_{kj} 的计算量是比较大的,当矩阵的行数与列数不相同时是无法进行 Doolittle 分解的,为克服这些缺陷,从分块的角度出发研究 Doolittle 分解. 使 Doolittle 分解能够适合可分解的任何矩阵.

2　*Doolittle* 分解递归算法的推导

设 A 是 m 行 n 列的矩阵,L 是 m 行 p 列矩阵,U 是 p 行 n 列矩阵,可以进行 Doolittle 分解,则

$$A_{m\times n} = L_{m\times p} U_{p\times n}, \quad p = \min(m, n).$$

$$\begin{pmatrix} A_{11\,d\times d} & A_{12\,d\times(n-d)} \\ A_{21\,(m-d)\times d} & A_{22\,(m-d)\times(n-d)} \end{pmatrix} = \begin{pmatrix} L_{11\,d\times d} & O_{12\,d\times(n-d)} \\ L_{21\,(m-d)\times d} & L_{22\,(m-d)\times(n-d)} \end{pmatrix} \cdot \begin{pmatrix} U_{11\,d\times d} & U_{12\,d\times(n-d)} \\ O_{21\,(m-d)\times d} & U_{22\,(m-d)\times(n-d)} \end{pmatrix} \cdot$$

$$\begin{cases} A_{11} = L_{11}U_{11}, & (1) \\ A_{21} = L_{21}U_{11}, & (2) \\ A_{12} = L_{11}U_{12}, & (3) \\ A_{22} = L_{21}U_{12} + L_{22}U_{22}. & (4) \end{cases}$$

由式(1)、(2) $\Rightarrow \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} (U_{11})$,即对 $\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$ 进行 Doolittle 分解

1) 可解出 L_{11}, L_{21}, U_{11}.

2) 由式(3) $\Rightarrow U_{12} = L_{11}^{1} A_{12}$.

3) 由式(4) 令 $A_{22} = A_{22} - L_{21}U_{12}$,

$$A_{22} = L_{22}U_{22}.$$

即对 A_{22} 进行 Doolittle 分解

3　算法求精

由上述推导知,若矩阵可进行 Doolittle 分解,使用一次分解就可求出 L 的前 d 行与 U 的前 d 行,接下来对 $(m-d)$ 行 $(n-d)$ 列矩阵 A'_{22} 进行 Doolittle 分解即可. 在求 U_{12} 的过程中要计算 L_{11} 的逆矩阵,为简化计算取 $d = 1$,可得到比较简单的计算公式:

$$l_{kk} = 1 \quad (k = 1, 2, \cdots, \min(m, n)),$$

$$u_{kj} = a_{kj} \ (j = k, k+1, \cdots, n; k = 1, 2, \cdots, \min(m, n)),$$

$$l_{ik} = a_{ik}/a_{kk} \ (i = k+1, \cdots, m; k = 1, 2, \cdots, \min(m, n)),$$

$$a_{ij} = a_{ij} - l_{ik}u_{kj} \ (i = k+1, \cdots, m; j = k+1, \cdots, n; k = 1, 2, \cdots, \min(m, n)).$$

1) 由上述公式得到 A 的 Doolittle 分解算法

```
for(k = 0;k < m 且 k < n;k + +)
{for(j = k;j < n;j + +) r_kj ←a_kj;
  l_kk ←1;
  for(i = k + 1;i < m;i + +) l_ik ←a_ik/a_kk;
  for(i = k + 1;i < m;i + +)
    for(j = k + 1;j < n;j + +)
      a_ij ←a_ij - l_ik u_kj;
```

矩阵 L 和 U 的其他元素为 0.

```
}
```

2) 对上述算法优化,得到紧凑的 Doolittle 分解算法

```
for(k = 0;k < m 且 k < n;k + +)
{
  for(i = k + 1;i < m;i + +) l_ik ←a_ik/a_kk;
  for(i = k + 1;i < m;i + +)
    for(j = k + 1;j < n;j + +)
      a_ij ←a_ij - l_ik u_kj;
}
```

算法中 $i < j$ 时,a_{ij} 是矩阵 U 的元素,否则为 L 的元素,且 $l_{ii} = 1$.

4　Doolittle 分解递归算法的程序实现

由算法知,数值计算程序相当简单,这里从略.

在程序中用 gcd() 函数求公约数,从而避免了在 2 个数相除时产生的积累误差,使计算结果的精确度得到提高.

Doolittle() 函数完成对矩阵的 Doolittle 分解,在 Doolittle() 函数又调用了 Doolittle() 函数,使得算法的结构更加严谨而容易理解.

main( ) 函数为顺序结构,内容依次为输入要分解的矩阵,调用 Doolittle 分解矩阵,输出分解结果(L,U 矩阵). 关于矩阵是否能进行 LU 分解[6-7]的判断这里从略. 若想添加上该内容,所需行列式的计算函数可参看文献[8].

```
# define N 7
struct fraction {int n,d;};
/ *用结构体存储数据的分子和分母 */
```

```
struct fraction a[N][N];
int gcd(int u,int v)
{
int r,t=v; if(u<0)u=-u;if(v<0)v=-v;
while(v!=0) {r=u%v;u=v;v=r;}
if(t>0) return u; else return -u;
}/*gcd*/
void Doolittle (int k,int m,int n)
/*Doolittle 分解的递归实现*/
{int i,j,g;
  if((k>=m)||(k>=n)) return;
    /*计算*/
  else
  {for(i=k+1;i<m;i++)
    /*aik=aik/akk*/
    {a[i][k].n= a[i][k].n * a[k][k].d;
     a[i][k].d= a[i][k].d * a[k][k].n;
     g=gcd(a[i][k].n, a[i][k].d);
     a[i][k].n= a[i][k].n/g;
     a[i][k].d= a[i][k].d/g;
  }
  for(i=k+1;i<m;i++)
  /*A22'=A22-L21 U12*/
    for(j=k+1;j<n;j++)
    {a[i][j].n=a[i][j].n*a[i][k].d*
a[k][j].d-a[i][j].d*a[i][k].n*a[k][j].n;
     a[i][j].d=a[i][j].d*a[i][k].d*
a[k][j].d;
     g=gcd(a[i][j].n,a[i][j].d);
     a[i][j].n=a[i][j].n/g;
     a[i][j].d=a[i][j].d/g;
  }/*aij=aij-lik ukj*/
Doolittle (k+1,m,n);
    /*对 Doolittle 函数的递归调用*/
  }/*计算*/
}/*Doolittle*/
main ( )
{int i,j,k,m,n;
  printf("m="); scanf("%d",&m);
  printf("n="); scanf("%d",&n);
  for(i=0;i<m;i++) /*输入*/
    for(j=0;j<n;j++)
    {printf("a %d %d=",i+1,j+1);
     scanf("%d",&a[i][j].n);
     a[i][j].d=1;}
printf(A:\n");/*输出*/
for(i=0;i<m;i++)
{
  for(j=0;j<n;j++)
  printf("%d\t",a[i][j].n);
  printf("\n");
}
Doolittle (0,m,n);
/*调用 Doolittle 函数*/
printf("L:\n");/*输出 L 矩阵*/
for(i=0;i<m;i++)
{for(j=0;(j<=i)&&(j<n);j++)
    if(j==i) printf("1");
    else if(a[i][j].d==1)
printf("%d\t",a[i][j].n);
    else
    printf("%d/%d\t",a[i][j].n,
a[i][j].d);
    printf("\n");
  }
  printf("U:\n");/*输出 U 矩阵*/
  for(i=0;i<m;i++)
  {
    for(j=0;j<i;j++) printf("\t");
      for(j=i;j<n;j++)
      if(a[i][j].d==1)
      printf("%d\t",a[i][j].n);
    else printf("%d/%d\t",a[i][j].n,
a[i][j].d);
    printf("\n");
  }
  printf("A=LU\n");
}/*main*/
```

5 程序运行示例

$$1)\begin{bmatrix}4&2\\3&1\\4&6\\8&1\end{bmatrix}=\begin{bmatrix}1&0\\3/4&1\\1&-8\\2&6\end{bmatrix}\cdot\begin{bmatrix}4&2\\0&-1/2\end{bmatrix}.$$

$$2)\begin{bmatrix}2\\45\\8\\6\end{bmatrix}=2\begin{bmatrix}1\\45/2\\4\\3\end{bmatrix}.$$

$$3)\begin{bmatrix}6&2&1&-1\\2&4&1&0\\1&1&4&-1\\-1&0&-1&3\end{bmatrix}=$$

$$\begin{bmatrix}1&&&\\1/3&1&&\\1/6&1/5&1&\\-1/6&1/10&-9/37&1\end{bmatrix}\cdot\begin{bmatrix}6&2&1&-1\\&10/3&2/3&1/3\\&&37/10&-9/10\\&&&191/74\end{bmatrix}.$$

6　结束语

文中是将矩阵分解由数值运算[9]推广到符号运算,希望将来能推广到复数矩阵的分解.

参考文献:

[1]陈建平,JERZY W. Cholesky 分解递归算法与改进[J]. 计算机研究与发展,2001,38(8):923 - 926.
CHEN Jianping, JERZY W. Recursive algorithm and improvement for cholesky factorization [J]. Journal of Computer Research and Development, 2001,38(8):923 - 926.

[2]陈建平. 矩阵三角分解的递归算法[J]. 南通工学院学报(自然科学版),2003,2(4):1 - 3.
CHEN Jianping. Recursive algorithm for matrix triangular factorization [J]. Journal of Nantong Institute of Technology (natural science),2003, 2(4):1 - 3.

[3]LIU Mode. The structured programming of Crout's method solving linear equation [J]. Journal of Sanming College,2003,20(2):62 - 66.

[4]智慧来,张礼达. 矩阵的 Crout 递归分解算法及程序设计[J]. 西华大学学报(自然科学版),2006,25(3):48 - 50.
ZHI Huilai, ZHANG Lida. Recursive algorithm and programming for matrix Crout's factorization [J]. Journal of Xihua University (natural science),2006,25(3):48 - 50.

[5]徐　仲,张凯院,陆　全,等. 矩阵论简明教程[M]. 北京:科学出版社,2002.

[6]RICHARD L, BURDEN J, FAIRES D. Numerical analysis :7th Edition[M]. Thomson Learning,2001.

[7]JAIN S K, GUNAWARDENA A D. Linear algebra: an interactive approach [M]. Beijing: China Machine Press,2003.

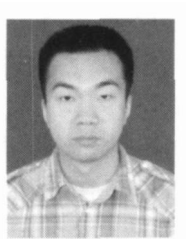
[8]智东杰. 按定义计算行列式及余子式的 C 语言递归程序设计[J]. 中国教育教学研究杂志,2004,3(12): 79 - 80.
ZHI Dongjie. C language recursive programming of computing determinant and complement minor according to their definitions [J]. The Research Magazine of Education and Teach in China,2004,3(12):79 - 80.

[9]郑桂水. Mathcad 2000 实用教程[M]. 北京:国防工业出版社,2000.

作者简介:

智东杰,男,1952 年生,中国科技研究交流中心研究员,主要研究方向为人工智能、符号计算.

Emai: zhidongjie @126. com.



智慧来,男,1981 年生,硕士研究生,主要研究方向为计算智能在水电站经济运行中的应用.