有限元线性代数方程组新的存储格式

李 洪，何文明

（温州大学数学与信息科学学院，浙江温州 325035）

摘 要：根据有限元总刚度矩阵大型稀疏、对称正定、带状的特点，给出了一种紧致存储格式，并给出了在此格式下代数方程组的求解方法和有关的 Matlab 程序.

关键词：有限元法；刚度矩阵；紧致存储；代数方程组

中图分类号：O242 **文献标志码**：A **文章编号**：1674-3563(2009)01-0009-07

DOI：10.3875/j.issn.1674-3563.2009.01.03 本文的 PDF 文件可以从 xuebao.wzu.edu.cn 获得

有限元方法是随着电子计算机的发展而出现的一门新的微分方程的数值解法，它在科学与工程计算中有着广泛的应用，这种方法最后归结为求解高阶线性代数方程组．在结构复杂和精度要求高的情况下，得到的刚度矩阵容量很大，往往存在机器内存容量不足的问题．有限元离散后的线性代数方程组的刚度矩阵具有稀疏、对称、正定、带状的特点，为了节省微机的存储空间，达到用小机器解决大问题的目的，可以利用这些特点采用紧致存储方式来处理这类问题，最后用共轭斜量法迭代求解方程，该法的优点是充分利用刚度矩阵的非零元信息，需要的存储量少编制程序简单．

1 刚度矩阵的紧致存储

有限元离散后的代数方程组的刚度矩阵（系数矩阵）中，非零元素的分布是有规律的，刚度矩阵每行的元素其实是各自由度对列主元自由度的贡献系数．根据有限元离散的特点，列主元自由度所对应结点的影响结点位置上的元素才可能是非零元素．当结点的影响结点的数目相差不大时，可用等带宽存储方法[1]存储每行非零元素，不足带宽处以零补齐．但一般情况下，各结点的影响结点数是不等的，这种情况下可以使用变带宽存储法．

1.1 一维数组的变带宽存储法

常用的变带宽存储使用一维数组来实现[2]．为了区分每行半带宽以内的元素在一维压缩存储数组（不妨设为 *KA*）中的位置，必须另外设置一个数组 *KD*，其长度为 *n*+1，*KD* 用于存放总刚度矩阵的对角元素在一维排列中的地址（首地址为 0）．这样当矩阵中的每行半带宽内的元素相继存入一维排列数组 *KA* 中时，*KD* 数组可以区分每行元素的地址．这种存储方法的空间需求依赖于结点编号，当结点编号很糟糕时，刚度矩阵的每行半带宽可能很大，存储量仍然相当可观．例如，当对单位正方形采用类似图 1 的均匀剖分（总的剖分点数为 node）时此法所需存储量大致为 $(\sqrt{\text{node}}+1)*\text{node}+\text{node}$.

收稿日期：2008-05-29

作者简介：李洪(1982-)，男，四川邻水人，硕士研究生，研究方向：有限元方法

1.2 二维数组变带宽存储法

根据有限元离散后的代数方程组刚度矩阵非零元素的分布规律，本文提出了利用 Matlab[3-4] 二维结构数组实现变带宽存储[5]的一个新算法. 为了方便讨论，定义 *EM* 为 Matlab 中的结构数组，它存储每行带宽内的元素. 定义 *IM* 为 Matlab 中与 *EM* 同大小的结构数组，存储每行带宽内的元素的地址. *IM* 中第 *i* 个数组 *IM*{*i*}的元素实际上是由与结点 *i* 相关联的结点序号按从小到大排列而成. 这个结构数组在后面形成总刚时不再变化. 网格剖分时用 Matlab 结构数组 *ED* 记录每个单元所属结点的编号，用 *PX*、*PY* 分别记录结点横坐标、纵坐标，用 *EI*、*EO* 两个一维数组分别记录内部结点编号、边界结点编号. 例如在图 1 中，本文采用上述的数据结构有：

$$IM\{1\}=[1,2,6,7],\ \ IM\{5\}=[1,2,6,7,8,12,13];$$

$$EM\{1\}=[a_{11},a_{12},a_{16},a_{17}],\ \ EM\{7\}=[a_{71},a_{72},a_{76},a_{77},a_{78},a_{7,12},a_{7,13}].$$

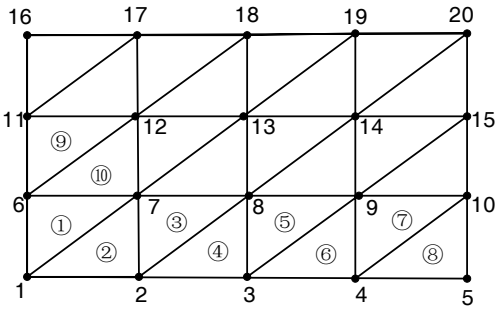

图 1　矩形网格均匀剖分图

Fig 1　Rectangular Grid Uniform Subdivision

这种存储格式只与网格剖分有关，与结点编号无关. 而通常按序号存储的变带宽、定带宽存储法与结点编号有很大关系，当影响结点序号相差很大时，将大大地增加存储量.

当求解区域采用均匀剖分（每行结点数相同）时，二维数组变带宽存储法又可以作如下的改进：对于刚度矩阵的非零元素仍用一个二维结构数组 *EM* 来存储，但对于带宽内元素的地址我们用一个单变量 k_0 来实现，舍去二维结构数组 *IM*. k_0 存储区域每行结点数. 对图 1 的网格，本文以 *i*=7 号结点为例实现上述的存储方案.

$$(k_0=7):\quad EM\{7\}=[a_{7,k_1},a_{7,k_2},a_{7,k_3},a_{7,k_4},a_{7,k_5},a_{7,k_6},a_{7,k_7}],$$

其中，第 *i*=7 个结点（这里只考虑内部结点）的影响结点号计算公式如下：

$$k_1=i-k_0-1,\ \ k_2=i-k_0,\ \ k_3=i-1,\ \ k_4=i,\ \ k_5=i+1,\ \ k_6=i+k_0,\ \ k_7=i+k_0+1.$$

改进的二维变带宽优化存储方案，由于减少了一个标记地址的二维数组，只增加一个存储每行结点数的单变量，节省存储空间的效果是相当明显的. 实际上对于单位正方形的剖分，本法所需要的存储量约为 $7*\mathrm{node}$.

当然对求解区域更加一般化，网格剖分产生的每行结点数不再相同了，根据上述存储格式的思想可以增加一个一维数组来存储每行结点数信息，只不过这时寻找带宽内元素的地址就会变得复杂一些.

2 算法的实现

考虑用有限元法解椭圆型边值问题[6]：

$$(P)\begin{cases}-\Delta u(x,y)=f(x,y) & (x,y)\in \Omega \\ u(x,y)=g(x,y) & (x,y)\in \partial\Omega\end{cases}$$

设求解区域为单位正方形：$\Omega=[0,1]\times[0,1]$，其边界为Γ，Ω_h为有限元子空间，Γ_h为相应的近似边界．通过单元分析与总体合成，得到$n\times n$阶的总刚度矩阵 *A* 和总的荷载列向量 *F*，即转化为求解网格剖分下含离散点$u_1,u_2,...,u_n$的有限元线性方程组：

$$A*u=F \tag{1}$$

其中，$u=[u_1,u_2,...,u_n]^T$，$F=[f_1,f_2,...,f_n]^T$，u ,F 为 *n* 解列向量，*n* 为剖分得到的总的结点数．

下面介绍在约束条件下采用 1.2 所述二维数组变带宽存储法求解线性代数方程组（1）的方法．根据有限元解方程的步骤把程序分成 6 块进行处理．

1）有限元前处理的 *M* 文件（Gridf）：网格剖分得到单元个数 danyuan，结点个数 node，内部结点数组 *EI*，边界结点数组 *EO*，结构数组 *ED*；

2）剖分信息组装总刚度矩阵和右端向量的 *M* 文件（Stiffmatf）；

3）总刚度矩阵约束处理的 *M* 文件（Adealf）；

4）右端向量约束处理的 *M* 文件（Fdealf）；

5）共轭斜量法解有限元线性代数方程的 *M* 文件（Solvef）：根据本文的二维变带宽存储方法迭代求解方程；

6）有限元后处理的 *M* 文件（Postf）：完成一些误差估计．

步骤 1），6）两步有专门的通用程序予以处理，这里只讨论 2）至 5）步，并给出了 Matlab 函数的头文件说明．详细算法见文后附录．

2.1 剖分信息组装总刚度矩阵和右端向量的 *M* 头文件（Stiffmatf）

Function[*EM*,*IM*,*F*]= Stiffmatf（*ED*, *NB*, *DNX*, DNY, *WG*, *fxy*, *dn*）

%-*DNX*，*DNY* 为结点基函数对自然坐标的 *X*，*Y* 导数在高斯积分点上的数值；

%-Matlab 结构数组 *ED* 存储每个单元所属结点号，*NB* 为结点基函数在高斯积分点上的数值；

%-*dn* 为单元结点数，*WG* 为高斯积分权重，*fxy* 为函数 *f*(*x*,*y*)的高斯积分点上的数值；

%-*EM*，*IM* 分别存储每行带宽内的元素与地址，*F* 存储右端向量．

2.2 总刚度矩阵约束处理的 *M* 头文件（Adealf）

删除由边界结点构成的那些行列，得到经约束处理后刚度矩阵的结构数组 *ENN*．实际上 *ENN* 存储内部结点构成每行带宽内的元素．

Function[*EMM*，*IMM*]=Adealf（*EI*，*IM*，*EC*）

%-*EI* 为内部结点数组，*IM*，*EM* 分别为存储每行带宽内元素的数值地址；

%-*EMM*，*IMM* 分别为总刚度约束处理后的存储结构．

2.3 右端向量约束处理的 *M* 头文件（Fdealf）

Function *FF* = Fdealf（*EI*, *EO, JPN*, *IM*, *EM*）

%-*JPN* 为结点的影响结点构成的 Matlab 结构数组，这些结点号按从小到大排序；

%-*FF* 为返回经约束处理的右端向量．

2.4 共轭斜量法解有限元线性代数方程的 *M* 头文件（Solvef）

本算法要多次用到 *M* 文件 Axdot，此函数文件主要完成矩阵 *A* 与向量 x_0 的乘法运算．输入

参数为矩阵 A 的二维存储结构 EM、IM 和向量 x_0．下面是共轭斜量法解方程的 M 头文件．

function x = Solvef（EMM，IMM，x_0，FF）

%-EMM，IMM，FF 分别是约束处理后刚度矩阵右端向量的信息；

%-x_0 为迭代的初始向量，x 为最终的解向量．

3 数值算例

考虑一类椭圆型边值问题[6]：

$$(P)\begin{cases} L\varepsilon u^{\varepsilon} \equiv \dfrac{\partial}{\partial x_i}\left(a_{ij}\left(\dfrac{x}{\varepsilon}\right)\dfrac{\partial u^{\varepsilon}}{\partial x_j}\right) = f(x), & in\ \Omega \\ u^{\varepsilon} = g(x), & on\ \partial\Omega \end{cases}$$

$$f(x)=e^{x_1+x_2},\ g(x)=2\sin(x_1)+4\cos(x_2),$$

$$a_{11}(x)=0.3+2x_1(1-x_1),\ a_{22}(x)=0.1+2x_2(1-x_2),$$

$$a_{12}(x)=a_{21}(x)=x_1x_2(1-x_1)(1-x_2).$$

区域 $\Omega=[0,1]\times[0,1]$，用线性有限元方法解此问题（P），记 K 为剖分的内部分割线数．记未经压缩的直接存储法为算法 1；一维变带宽压缩存储法为算法 2；本文二维变带宽存储法为算法 3；在此基础上的改进方法为算法 4．以上几种算法程序所需时间与程序所需存储空间的情况见表 1．

表 1 几种算法比较

Table 1 The Comparison of Several Algorithms

算 法		存储空间	储存约束时间	解方程时间	程序运行时间
算法 1	K=40	320.1KB	14.6362s	1.9854s	16.6216s
	K=100	5.9M	88.4144s	192.2077s	280.6221s
算法 2	K=40	86.1KB	4.9714s	0.80461s	5.77601s
	K=100	645.2KB	4755.5608s	65.4650s	4821.0258s
算法 3	K=40	119.1KB	7.4220s	4.0112s	11.4332s
	K=100	783.6KB	129.7702s	113.5361s	243.3063s
算法 4	K=40	67.6KB	9.9667s	12.2326s	22.1993s
	K=100	580.2KB	110.9929s	87.1978s	198.1907s

上述结果是在 Sun-小型机上得到的．从表 1 可见，当剖分细度减小（K 增大）时，从存储空间消耗上比较，算法 2-4 比算法 1 大大的节省了内存空间．从消耗时间上比较；当内存足够大的情况下，算法 3 与算法 4 所需时间较少．所以当剖分的尺度越来越小时，无论从存储空间上考虑还是消耗时间上考虑算法 3 和算法 4 都是较好的选择．

4 结束语

本文提出了一个刚度矩阵压缩存储的改进算法．本算法充分利用网格剖分信息和有限元离散后代数方程组刚度矩阵非零元素的分布特点，用 Matlab 二维结构数组实现了变带宽存储，获得了一种新的高效的存储方案．

参考文献

[1] 周建华, 顾宏中. 有限元线性代数方程组的紧致存储格式及解法[J]. 上海交通大学学报, 1996, (12): 36-40.

[2] 李开泰, 黄艾香, 黄庆怀. 有限元方法及其应用[M]. 北京: 科学出版社, 2006: 63-66.

[3] 任玉杰. 数值分析极其 Matlab 实现[M]. 北京: 高等教育出版社, 2007: 173-175.

[4] 陈杰. MATLAB 宝典[M]. 北京: 电子工业出版社, 2007: 80-100.

[5] 李辉. 大型结构有限元分析中刚度矩阵二维变带宽优化存储和方程组的解法[J]. 四川建筑科学研究, 1995, (1): 2-6.

[6] He M, Cui J. A Finite Element Method for Elliptic Problems with Rapidly Oscillating Coefficients [J]. BIT Numerical Mathematics, 2007, 47: 77-102.

New Storage Format and FEM Linear Algebraic Equations

LI Hong, HE Wenming

(School of Mathematics and Information Science, Wenzhou University, Wenzhou, China 325035)

Abstract: According to the finite element stiffness matrix large, sparse, symmetry, positive, and belt-shaped features, this article gives an improved compact storage format and solving method. Besides, the related procedures are given.

Key words: Finite element method (FEM); Stiffness matrix; Compact storage; Algebraic equations

附录：算法 3 的主要程序清单

1 剖分信息组装总刚度矩阵和右端向量的文件

```
Function [EM, IM, F] = Stiffmatf( ED, NB, DNX, DNY, WG, fxy, dn)
EM = cell(node,1); IM = cell(node,1);
for j0 = 1: danyuan
    for i = 1: dn
        ip = ED{j0}(i);
        ND = JPN{ip}; %-计算结点 ip 的影响结点
        F(ED{j0}(i)) = F(ED{j0}(i)) + 2*S*fxy*WG(i)*NB(i);
         for j = 1: dn
              %-寻找 jp 在影响结点数组 ND 中的位置
           jp = ED{j0}(j); jp0 = PFind(jp, ND); IM{ip}(jp0) = jp;
           EM{ip}(jp0) = EM{ip}(jp0) + 2*S*(WG(i)*(DNX(i)*DNX(j) + DNY(i)*DNY(j)));
         end
    end
end
```

2 总刚度矩阵约束处理文件

```
Function  [EMM, IMM] = Adealf(EI, IM, EM)
EMM = cell(length(EI),1); IMM = cell(length(EI),1);
for i = 1:length(EI)
    ip = EI(i); mn = length(IM{i}); kp = 0;
   for j = 1:mn
       jp0 = IM{ip}(j);
          %-寻找 jp0 在内部结点数组中的位置
          NP = PPFind(jp0, EI);
          %- jp0 在数组中的位置为 NP，并记录非零元个数
          if (NP ~= 0)
              kp = kp + 1;
              EMM{i}(kp) = EM{ip}(j); IMM{i}(kp) = NP;
          end
    end
end
```

3 右端向量约束处理文件

```
Function  FF = Fdealf(EI, EO, JPN, IM, EM)
FF = zeros(1, length(EI));
for i = 1:length(EI)
    sum1 = 0; ip = EI(i);
    mn = length(JPN(ip));  %-计算 ip 行带宽
    for j = 1:mn
        jp0 = IM{ip}(j);
       %-寻找 jp0 在边界点数组中的位置
       if (NP ~= 0)
          U0(jp0) = U0xy(PX(jp0), PY(jp0));
          sum1 = sum1 + EM{ip}(j) * U0(jp0);
       end
       FF(i) = F(EI(i)) - sum1;
    end
end
```

4 共轭斜量法解有限元线性代数方程文件 *Axdot* 与 *Solvef*

```
Function  b = Axdot(EM, IM, x0)
```

```
m = size(IM,1);  %-结构数组的行数
for  i = 1 : m
     b(i) = 0;
   n = length(IM{i}); %-结构数组的每行带宽
   for  j = 1 : n
      b(i) = b(i) + EM{i}(j) * x0(IM{i}(j));  %-取出 IM{i} 元素
   end
end

Function x = Solvef (EMM, IMM, x0, FF)
b0 = Axdot(EMM, IMM, x0);
r1 = b - b0;  p1 = r1;
r11 = Axdot(EMM, IMM, r1);
d = dot(r1, r1) / dot(r1, r11);  %-取出 Matlab 中的系统点积函数 dot
x = x0 + d * p1;
p11 = Acdot(EMM, IMM, p1);
r2 = r1 - d * p11;
f = dot(r2, r2) / dot(r1, r1);
p2 = r2 + f * p1;
n = 1;  temp = norm(r2);
while temp >= 0.000000001  &  n < size(EMM,1)
      x0 = x;    p1 = p2;    r1 = r2;
      p11 = Axdot(EMM, IMM, p1);
      d = dot(r1, r1) / dot(p1, p11);
     x = x0 + d * p1;    r2 = r1 - d * p11;
      f = dot(r2, r2) / dot(r1, r1);   p2 = r2 + f * p1;
      n = n + 1;  temp = norm(r2);
 end
 p22 = Axdot(EMM, IMM, p2);
 d = dot(r2, r2) / dot(p2, p22);
     x = x + d * p2;
```

（编辑：王一芳）