网格节点标号的最优化算法*

薛希超　祝丕琦
(三机部六〇一所)

AN OPTIMAL ALGORITHM FOR MESH NODE LABEL

Xue Xi-chao　Zhu Pi-qi
(601 Institute, Third Ministry of Machine-Building Industry)

Abstract

To reduce the computer storage and time in solution of the systems of large linear algebraic equations, we have developed an improved algorithm of matrix bandwidth, storage or wave front minimization, which can increase convergence rate, and make possible the convergence that can not be obtained by using some algorithms in some cases.

一、引　言

无论是变带宽法或是波前法，主元次序的优化对于更有效地利用矩阵的稀疏性来减缩计算机的计算量和存贮量，都起着关键性的作用．按照优化好了的消元次序，在消元过程中，系数矩阵的带宽、存贮或波前，相对来说都是最小的．在事先采用 LU 分解的高斯消去法中，求解方程组所需要的时间是和带宽的平方成比例的．当进行了节点标号优化以后，如果带宽减缩 50%，就意味着解题时间减少 75%．而当采用波前法以便小机器解大题目时，所解题目的大小，完全取决于波前的大小．如果波前减缩 50%，就意味着解题规模大体上可以扩大三倍．所以，减缩带宽或波前的意义，就很显然了．

对于简单的问题，经过对网格的仔细观察，用手工编号也能够得到最小带宽或波前．然而，对于包含复杂拓扑的大型问题，手工编号是很困难的．因此，在过去的十多年中，有许多研究者致力于这方面的工作．如在 [1—5] 中，为简化稀疏矩阵的带宽，分别采用了行置换、整线性规划、图的理论及迭代等技术．1972 年，H. R. Grooms[4] 提出了当时看来是最有效的算法．但，它是经验性的，需要凭经验确定两个公式中的常数．常数的好坏，直接影响方法的质量．1976 年，G. Akhras[6] 根据网格问题带状矩阵的各种拓扑性质，选出"跨度"、"均值"按点号次序单调上升和"和"在关联点个数相同的点号中按点号次序单调上升的性质，作为推导再标号算法的准则．所得算法抛开了经验数字，优化结果较

* 1979 年 9 月 8 日收到．

之 Grooms 结果好得多，也快得多．但是，Akhras 的均值公式对某些情况，方法是不收敛的．此外，“和”准则与均值准则也有些重复．所以，Akhras 算法有时在跨度或均值相等的时候，发生徘徊情况．例如，表 3 中的后两例，便不能收敛．

本算法对 Akhras 算法进行了改进，抛弃了“和”准则，修改了均值定义，增加一个“动态度”判断，于是上述问题就可解决了．应用本算法计算了大量例题，通过与 Akhras 算法比较，事实表明，原来不收敛的题目，现在收敛了；原来已经收敛了的题目，现在又提高了收敛速度．此外，本文还将该算法推广到波前法中去，并可对存贮最小进行优化．对算法的原理，也做了简单直观地分析．

应该指出，所有的准则，单独拿出一项，对一般性的稀疏矩阵的最佳带宽结构可能未必都是满足的．然而，假若网格是遵照上述准则编号的，则相应的矩阵将具有较小带宽．

本文提出的方法的收敛性没有数学证明．但是，方法的效率和可靠性，经过了大量例题的数字对比和考验．

二、方法的导出思想

为了说明方法的基本导出思想，首先我们来观察一个由八个三角形构成的具有十个节点的最佳带宽为 3 的网络，如图 1 所示．这个网络的拓扑性质在表 1 中给出．表 1 的第 2 列为节点标号向量 l，第 3 列为节点关联矩阵 M，即以每个节点的关联点号(不包括该节点本身)为行所构成的矩阵．表 1 的第 4 列为节点关联点号(除掉该节点本身)的“和”向量 SI，第 5 列为节点关联点（除掉该节点本身)个数向量 NI，第 6 列为节点均值向量

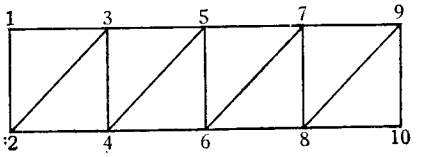

图　1

表　1

i	$l[i]$	$M[i, j]$				SI	NI	SP	SS
1	2	3				4	5	6	7
1	1	2	3			5	2	2.5	4
2	2	1	3	4		8	3	2.66	5
3	3	1	2	4	5	12	4	3	6
4	4	2	3	5	6	16	4	4	8
5	5	3	4	6	7	20	4	5	10
6	6	4	5	7	8	24	4	6	12
7	7	5	6	8	9	28	4	7	14
8	8	6	7	9	10	32	4	8	16
9	9	7	8	10		25	3	8.33	17
10	10	8	9			17	2	8.5	18

SP，其分量的值等于向量 *SI* 的对应分量除以向量 *NI* 的对应分量（当 $NI<2$ 时，*SI* 和 *NI* 要计及该节点本身）. 第 7 列为节点跨度向量 *SS*，即以每个节点的最小关联点号和最大关联点号（要包括该节点本身）之和为分量所构成的向量. 从表 1 的 6，7 两列可看出，节点的均值和跨度都是按着单调上升的次序排列. 这种性质，直观上可以这样去分析：一个网络所对应的系数矩阵的带宽等于这个网络最大的相邻节点号之差加 1. 因此，一个具有最小带宽的矩阵所对应的网络最大相邻节点号之差，应当保持最小. 也就是说，在这种情况下，每个节点的相邻节点都比较相对地接近于该节点号之值，因而每个节点的相邻节点号的平均值（即均值）也就比较相对地接近该节点号之值. 那么每个节点的均值自然应该按其节点号次序单调上升. 对于跨度可做类似地分析. 以上述网络节点的均值和跨度的性质作为准则，便可导出下述使网格矩阵带宽，存贮或波前极小化的一个自动再标号算法. 整个算法以迭代的方式使用了上述两个准则，并且分成两个步骤.

1. 分别根据跨度和均值准则重复执行节点再编号过程，直到所得的带宽、存贮或波前不能再被减缩为止.

2. 在步骤 1 以后，单独按照均值准则重复执行节点再编号过程，直到所获得的带宽、存贮或波前不能进一步再被减缩为止. 步骤 2 被执行完后，有时还要回到步骤 1，以便尽可能地减缩带宽、存贮或波前. 对于一般问题，通常在步骤 1 就可以获得带宽、存贮或波前的极小化.

三、算法说明

1. 建立节点关联矩阵

根据网格的节点关联信息，程序自动形成关联矩阵 $M[i,j]$ 和度向量 $NI[i]$，其中

$$i=1,2,\cdots,NP \tag{1}$$

是网格的任意初始标号向量，*NP* 是网格的节点总数.

$$j=1,2,\cdots,NI[i] \tag{2}$$

代表 i 点的关联点序号.

2. 运用跨度准则（子块 *KD*）

针对当前节点标号向量 l：

$$l=l[1],l[2],\cdots,l[i],\cdots,l[NP] \tag{3}$$

计算跨度向量 *SS*，其第 i 个分量为

$$SS[i]=\max_j(l[M[i,j]],l[i])+\min_j(l[M[i,j]],l[i]). \tag{4}$$

根据分量数值的大小重新排序，得到一新的节点标号向量 L：

$$L=L[1],L[2],\cdots,L[i],\cdots,L[NP], \tag{5}$$

使得

$$SS[L[i]]\leqslant SS[L[i+1]] \tag{6}$$

成立. 其中等号成立的条件是

$$NT[L[i]]\leqslant NT[L[i+1]], \tag{7}$$

式中

$$NT[L[i]] = NI[L[i]] - \sum_{j=1}^{NI[L[i]]} \sum_{k=1}^{L[i]-1} \delta_{k,M[L[i],j]}, \tag{8}$$

$$\delta_{k,M[L[i],j]} = \begin{cases} 1, & \text{当 } k = M[L[i],j], \\ 0, & \text{当 } k \neq M[L[i],j]. \end{cases}$$

$NT[L[i]]$ 代表 $L[i]$ 点的关联点 $M[L[i],j]$ (j 取 (2) 式)中尚未编号的个数，称为动态度. (7) 式表示在跨度相等的时候，动态度小的节点先编号. (7) 式等号成立的条件是

$$L[i] < L[i+1], \tag{9}$$

这意味着在动态度也相等的时候，保持原编号 l 的顺序. 最后做代换

$$l[L[i]] = i, \tag{10}$$

作为下一次迭代的当前节点编号向量，于是关联矩阵 $M[i,j]$ 不用重新形成. (10) 式的实际意义是，当 $l[1] = 5$，表示初始 $i = 1$ 的第 1 号节点现在变为第 5 号节点.

3. 运用均值准则(子块 *MD*)

做法完全同于跨度准则，只是用 SP 代替 SS. SP 的第 i 个分量为

$$SP[i] = SI[i]/(NI[i] + \delta_{NI[i],1}), \tag{11}$$

式中

$$SI[i] = \sum_{j=1}^{NI[i]} l[M[i,j]] + l[i] \cdot \delta_{NI[i],1}. \tag{12}$$

δ 记号的意义同前.

4. 求最小(子块 *MIN*)

在每运用一次跨度或均值准则时，都会得到一个新的标号向量 l，因此也就相应地得到一个新的最大带宽、存贮或最大波前的数值 B. 优化的目标是捕获对应于较小 B 值(记作 BB) 的标号向量 LP，其余的标号向量 l 都可随算随弃. B 的计算公式为：

对于带宽优化

$$B = \max_i (l[i] - \min_j L[M[i,j]] + 1). \tag{13}$$

对于存贮优化

$$B = \sum_{i=1}^{NP} (l[i] - \min_j L[M[i,j]]) + NP. \tag{14}$$

对于波前优化

$$B = \max_i \sum_{k=i+1}^{NP} (1 - \delta_{P[k],0}) + 1, \tag{15}$$

式中

$$P[L[M[l[i],j]]] = L[M[l[i],j]], \quad \text{仅当 } L[M[l[i],j]] > i, \tag{16}$$

$$i = 1, 2, \cdots, NP-1, \quad j = 1, 2, \cdots, NI[l[i]].$$

P 向量初值置零.

5. 检验和控制(子块 *JY*, *KZ*)

为了避免程序进入无限的循环，要随时检查 B 和向量 l 是否一直在变化. 如果发现连续三次没有变化，或者连续三次等幅振荡，则应将程序转到下一个步骤.

自动节点再标号算法粗框图示于图 2 中.

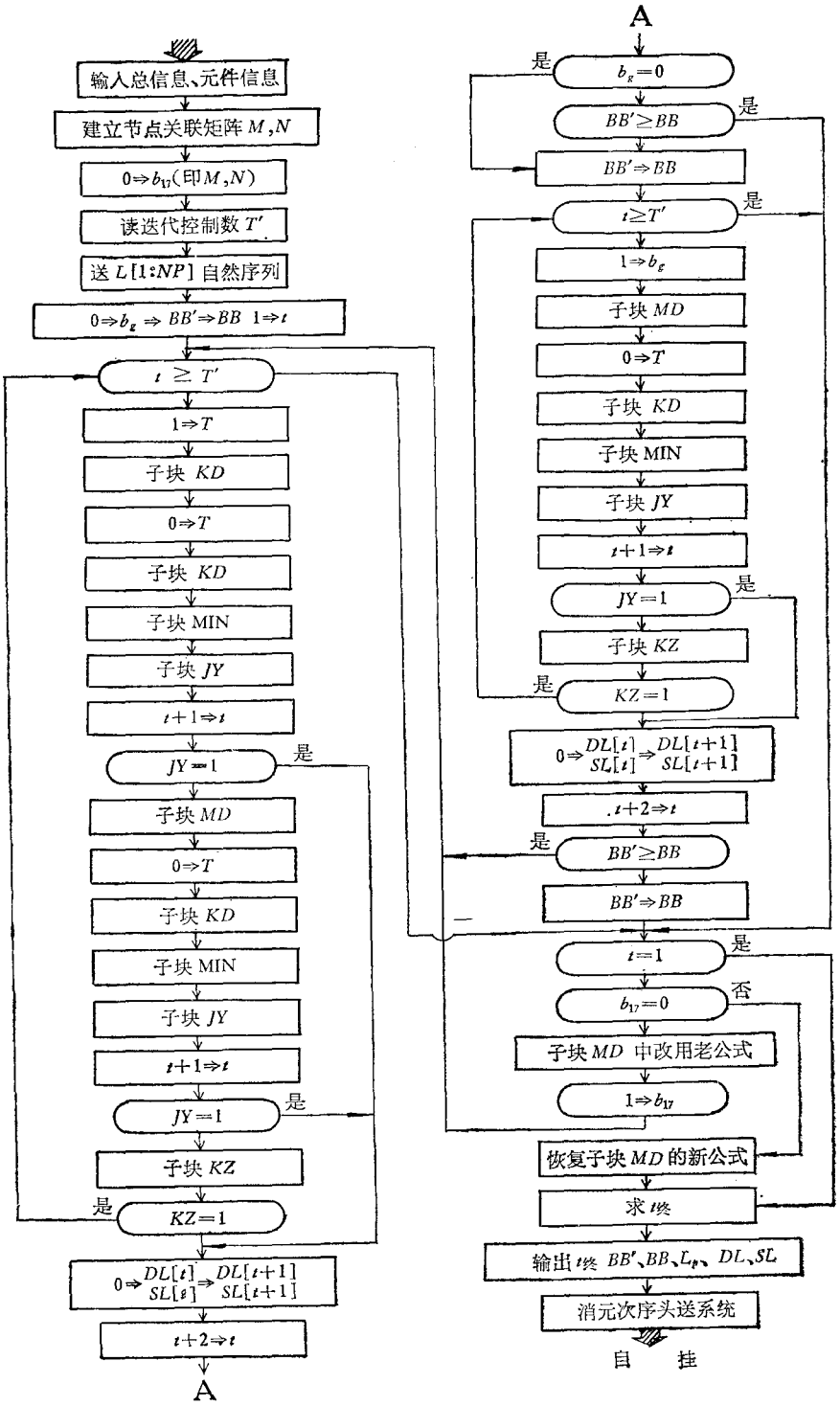

图 2 自动节点再标号算法粗框图

下面，对算法的改进之处，作一简单的剖析. 为此，用 Akhras 算法和本算法同时对例 4 (图 6)进行优化运算，其优化的第一步结果列于表 2 中.表 2 中的 3—5 列，是 Akhras 的跨度、均值和"和"的结果. 不难看出，尽管图 6 的节点编号并不是最优的，但却完全满

表 2

点　号	*NI*	Akhras			本　算　法		
		SS	*SP*	*SI*	*SS*	*SP*	*NT*
1	2	3	4	5	6	7	8
1	3	7	3.5	14	7	4.33	3
2	5	8	4	24	8	4.4	4
3	5	10	5	30	10	5.4	4
4	3	11	5.5	22	11	6	2
5	5	11	5.5	33	11	5.6	3
6	8	12	6	54	12	6	4
7	8	14	7	63	14	7	4
8	5	15	7.5	45	15	7.4	2
9	3	15	7.5	30	15	7	1
10	5	16	8	48	16	7.6	1
11	5	18	9	54	18	8.6	1
12	3	19	9.5	38	19	8.67	0

足了 Akhras 的准则，因此优化工作无法进行下去. 分析其原因，主要是 Akhras 的均值定义相当于本算法公式 (11)，(12) 中取 $\delta_{NI[i],1}=1$. 在再标号过程中，本来 i 点的编号是待求的，其编号大小应取决于周围关联点号的均值大小. 但若在每步计算均值时，都将 i 点的老编号计算在内，当老编号和最优编号相差较大时，就会起到很坏的作用，因此影响了算法的收敛速度，甚至使某些情况不收敛. 而"和"准则又和均值准则相似，对于 4，5 两点及 8，9 两点的 *SS*，*SP* 相等情况，没有帮助，所以 Akhras 算法在这个例题面前就无能为力了. 例 5 也有类似的情况. 表 2 中的 6—8 列，是本算法给出的跨度、均值及动态度的结果. 从 *SS* 数据中看出，虽然 8，9 两点的数值也相等，但 *NT* 不等，故可以优化. 由此可见，动态度的作用. 从 *SP* 数据中看出，4，5 两点及 8，9 两点都有差别，故也可以优化，由此可见，改进后的新均值公式的作用. 例 5 也得到了同样的解决. 但另一方面，当节点编号优化到接近最优编号时，考虑到在均值公式中加进老编号，会起到好的作用，所以这时再用老均值公式优化几次，也就达到最优了. 本算法有这一步. 实际上，绝大多数题目，在新均值公式下就已经收敛到最优了.

四、计 算 实 例

为了与 Grooms 和 Akhras 的算法进行比较，选择了他们共同算过的有网格图形的八个例题[4]. 此外还给出了一个机身结构最佳化设计的大型实用例题，这个例题的初始编号已经进行了精心的人工优选. 最后还有两个例题，是 Akhras 算法所不能收敛的.

问题的简单描述列在表 3 中，例 7—11 的网格图形示于图 3—7 中，计算结果及其对

表 3

例 题	节点数	线条数	三角形数	四边形数	初始带宽	最后带宽	初始存贮	最后存贮	初始波前	最后波前
1	16	16	0	0	16	4	45	45	3	3
2	45	85	0	0	37	6	620	249	19	6
3	19	34	0	0	19	5	69	69	4	4
4	42	81	0	0	38	9	263	246	7	7
5	56	4	0	54	53	20	867	764	19	19
6	20	9	0	8	17	8	116	96	9	6
7	101	0	151	0	37	20	947	946	13	12
8	62	78	0	0	49	10	486	272	12	7
9	227	0	21	227	50	50	7566	4915	42	42
10	12	0	0	6	6	5	53	47	6	5
11	12	0	12	0	6	4	53	40	6	4

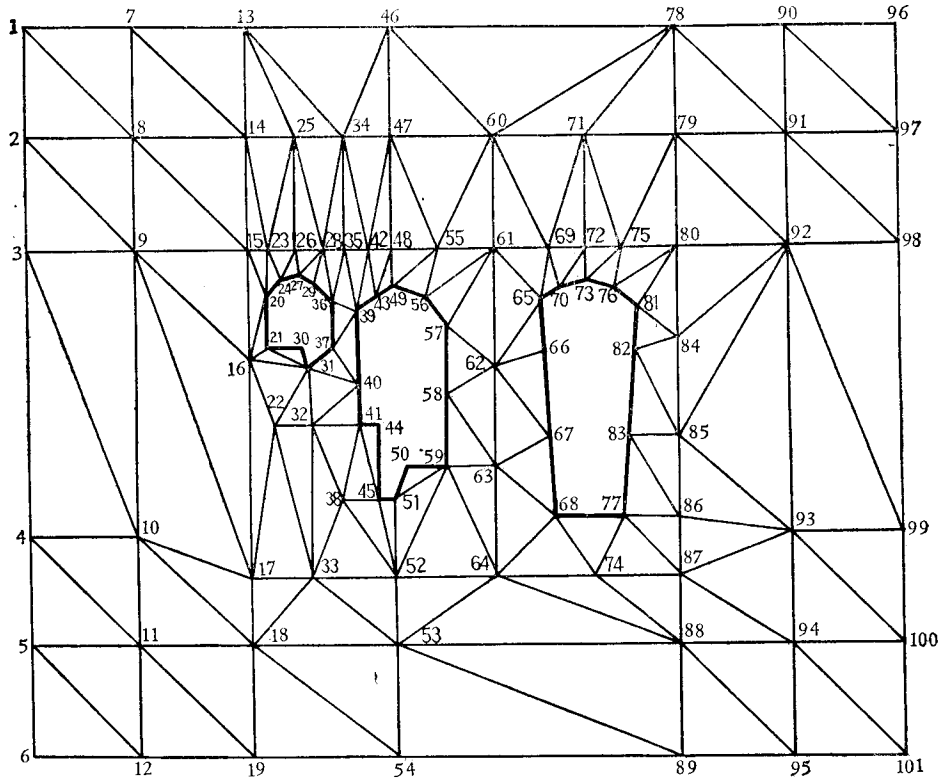

图 3 例 7

比列在表 4 中. 为了对比方便，Grooms 算法的结果取自[6]，为 Akhras 所重做. 表 4 中的迭代次数，是指使用准则的次数. 计算时间，Akhras 用的是 APLSV 语言程序，在 IBM 370/158 机器上实现. 本文用的是手编程序，在 DJS-15 机器上实现. 本文的时间是包括动用外部设备在内的实用时间，用来说明本算法是十分迅速的. 带宽数值，是按照本文的带宽定义列出的. 存贮是针对存贮极小化算得的. 存贮数值及波前数值，均只在本算法

中有结果．由表 3 可以看出，带宽、存贮及波前的优化，效果是显著的．由表 4 可以看出，本算法对各种类型的例题都收敛．迭代次数清楚地表明了本算法收敛速度得到了提高，最后带宽表明本算法结果为佳．

表 4

例题	Grooms			Akhras			本算法		
	迭代次数	时间秒	最后带宽	迭代次数	时间秒	最后带宽	迭代次数	时间秒[1]	最后带宽
1	12	2.05	4	9	1.20	4	3	4	4
2	101	46.48	8	22	8.53	6	10	6	6
3	5	1.15	6	9	1.63	5	4	5	5
4	118	66.45	9	30	13.31	10	14	7	9
5	200	117.50	24	33	25.05	20	17	13	20
6	20	3.03	9	16	3.5	8	9	5	8
7	75	149.60	21	44	74.22	19[2]	30	21	20
8	124	111.43	12	40	30.30	11	39	22	10
9	—	—	—	—	—	—	133	226	50
10	—	—	—	不收敛			5	4	5
11	—	—	—	不收敛			18	5	4

1) 所用机器不同，且包括部分打印时间．

2) 此例原文[6]有误．

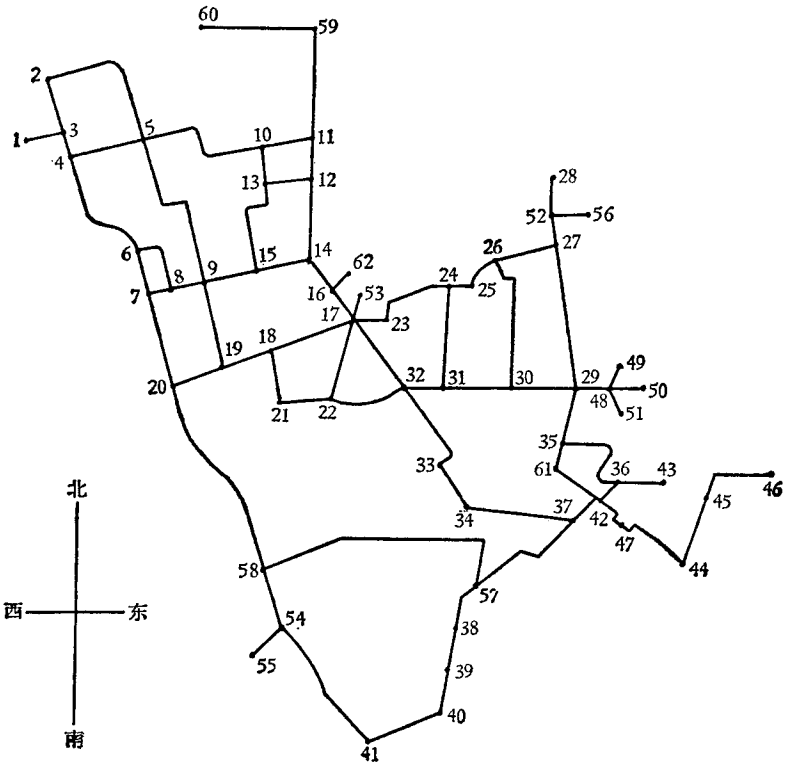

图 4　例 8

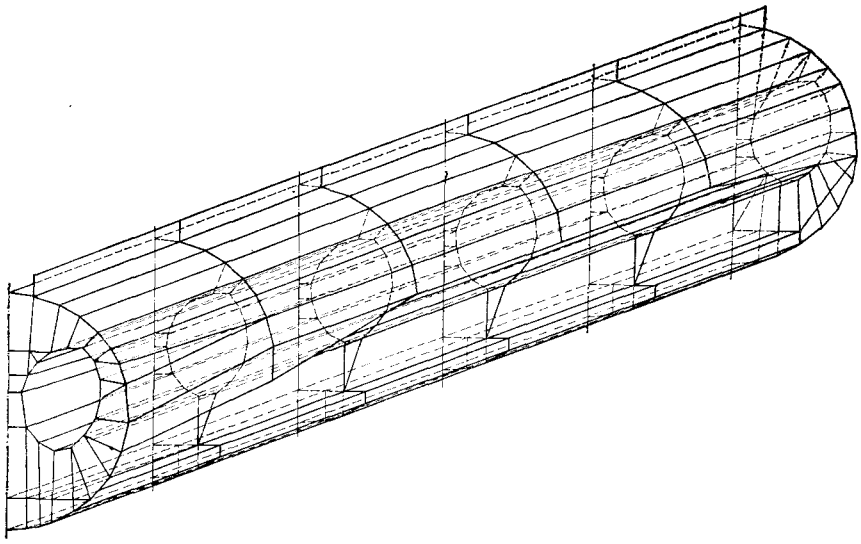

图5 例9

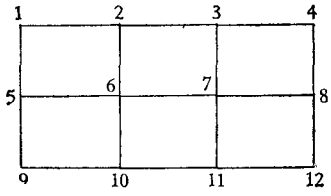

图6 例10

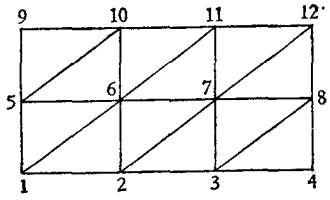

图7 例11

五、结 论

通过减缩对称稀疏关联矩阵的带宽、存贮或波前，本文实现了一种改进的算法，这种算法减少了求解方程组的机器时间和所需存贮量. 方法能够处理某些其它算法所不收敛的问题，而且是简单和快速的. 通过和其它算法的比较，表明本算法取得了较好的实际效果.

该算法还有节省信息和数据准备工作量的用处.

参 考 文 献

[1] R. Rosen, Matrix Bandwidth Minimization, Proceed. Nat. Cont. ACM, 23th. 1968.
[2] R. P. Tewarson, Comp. J., **10**: 3(1967), 300—305.
[3] F. A. Akyuz, S. Utku, *AIAA. J.*, **7**: 4(1968), 728—730.
[4] H. R. Grooms, Algorithm for Matrix Bandwidth Reduction, J. Struct. Div., ASCE, 98, No. STI, 203—214, Jan. 1972.
(译文: "应用数学与计算数学", 第2期, 1979, 4, 尹文霖).
[5] 曹志浩, 大型线性代数方程集的直接解法, 复旦大学学报(自然科学版), 第一期, 1974.
[6] G. Akhras, G. Dhatt, An Automatic Node Relabeling Scheme for Minimizang a Matrix or Network Bandwidth, Int. J. Num. Meth. Eng., V. **10**: 4(1976).