第 34 卷 第 2 期 力 学 学 报 Vol. 34, No. 2
2002 年 3 月 ACTA MECHANICA SINICA March, 2002

有限元分析快速解法 1)

陈 璞 孙树立 袁明武
(北京大学力学与工程科学系, 北京 100871)

摘要 基于结构分析有限元方程组的特征, 提出了在刚度矩阵及其因子的超方程概念下的细胞稀疏索引存贮方案. 与传统的稀疏索引存贮方案相比, 它可以减少磁盘空间和内存的占用量约 30%. 同时, 这一存贮方案也可以减少关于索引的操作. 结合双向循环展开技术, 发展了一种适合于多维有限元分析的快速稀疏直接静力求解方法. 工程算例表明, 所建议的方案在存贮量和速度方面显著地改进了直接求解法的效率.

关键词 高性能计算, 稀疏矩阵, 有限元分析, 静力求解器

引 言

在科学和工程计算领域, 椭圆型微分方程的有限元或有限差分离散经常产生大型或巨型的稀疏线性方程组 $\boldsymbol{Ax}=\boldsymbol{f}$. 其中 $neq \times neq$ 的对称正定的系数矩阵 $\boldsymbol{A}$ 通常称为总体刚度矩阵, $\boldsymbol{f}$ 和 $\boldsymbol{x}$ 分别是已知的荷载向量和待求的位移向量. 由于工程有限元法问题的复杂性和解题规模迅速增加, 线性方程组的未知量数目常常达到 10^4 甚至 10^6. 这给采用分块一维活动列存贮方案或波前法的传统直接求解器带来了巨大的困难, 它主要反映在求解时间和总体刚度矩阵的大小上.

有限元快速解法是指在非并行计算机上比传统的分块一维变列高解法或波前法快数倍至数十倍的有限元线性代数方程组的解法. 尽管计算机的速度、内存、外存容量等在提高, 但随着工程实际问题复杂程度的增加和分析要求的提高, 计算机性能的提高并不能完全满足大规模计算的需要. 另一方面, 更快更节省存贮空间的算法也一直是计算科学所追求的目标之一. 有限元分析的快速解法是计算力学的核心技术之一, 也是计算力学和计算数学领域中的一个非常重要的课题. 它的进步将为工程与科学计算带来发展的动力. 从实践和研究两方面的意义上看, 有限元的快速解法一直是计算力学的前沿课题之一 [15].

众所周知, 总体刚度矩阵的三角分解 $\boldsymbol{A}=\boldsymbol{LU}=\boldsymbol{LDL}^{\mathrm{T}}$ 是求解正定稀疏线性方程组 $\boldsymbol{Ax}=\boldsymbol{f}$ 中计算量最大的部分. 其中, 因子 $\boldsymbol{L}=(l_{ij})$ 和 $\boldsymbol{D}=\mathrm{diag}(d_i)$ 分别是带单位对角线的下三角矩阵和对角矩阵. 因此, 本文将着重讨论三角分解的策略. 在过去 30 年, 不少有限元软件采用了分块一维活动列存贮方案的解法 [1,2]. 最近几年, 总体刚度矩阵的存贮逐渐变为稀疏存贮法. 国际上一些著名的有限元分析软件已经将稀疏存贮的三角分解作为主要求解方案.

与传统的方案比较, 稀疏存贮解法仅仅需要为刚度矩阵及其因子的非零元分配存贮空间, 矩阵的运算都在索引形式下进行, 只有非零元才被处理. 稀疏存贮的解法比传统的分块一维活动列存贮更快速, 需要更少的磁盘空间. 当然, 稀疏解法的编程远比传统的解法困难.

2000–03–20 收到第一稿, 2001–03–16 收到修改稿.
1) 国家教育部留学回国人员科研启动基金和国家自然科学基金 (10172005) 资助项目.

优异的数值方法能在计算机上大大地增加求解效率，但它不是影响计算效率的唯一因素．近年来，矩阵运算的循环展开 (unrolling) 技术已广泛地用于并行和串行计算中，它可以有效地减少寄存器 (register) 和随机存储器 (RAM) 之间不必要的数据传送，从而提高计算效率．关于循环展开的介绍可见文献 [5]．细胞稀疏存贮方案为在稀疏存贮下采用循环展开技术提供了良好的数据结构．本文首先在细胞稀疏存贮的条件下引入了双向循环展开．它同时考虑已完成消元部分和被消元部分的循环展开，具有很高的运算效率．

本文提出了细胞稀疏存贮方案，对于三维有限元分析这一方案的总体刚度矩阵大小比一般的稀疏存贮方案减少了约 30%．

1 超方程和刚度矩阵的存贮方案

1.1 传统的稀疏索引存贮方案

假定刚度矩阵已经使用了修正最小度算法 (modified minimum degree) 或多重剖分法 (nested di-section)[6,7] 对矩阵进行了优化．为了更清楚地讨论问题，取一个 6×6 的总体刚度矩阵如下

$$\boldsymbol{A}=\begin{bmatrix} 11 & & a & b & & c \\ & 22 & & & d & \\ & & 33 & e & f & \\ & & & 44 & g & \\ & \text{SYM} & & & 55 & h \\ & & & & & 66 \end{bmatrix}=\begin{bmatrix} (11) & & (a\ b) & & (c) \\ & (22) & & (d) & \\ & & \begin{pmatrix} 33 & e \\ & 44 \end{pmatrix} & \begin{pmatrix} f \\ g \end{pmatrix} & \\ & \text{SYM} & & (55) & (h) \\ & & & & (66) \end{bmatrix} \quad (1)$$

其中 $a,b,\cdots,h$ 是非零实数．由于矩阵是对称的，仅需考虑它的上三角部分．我们首先讨论式 (1) 中两个表示法的第 1 个．

矩阵在稀疏存贮方案下被视作紧凑行向量的顺序组合，共需要 3 个数组 $ICN(1:neq)$，$JCN(1:nzr)$ 和 $PCN(1:nzr)$ 来表示矩阵 $\boldsymbol{A}$ 的上三角部分 [7]，其中 nzr 是非零元的个数．下表是式 (1) 中第 1 个矩阵表示的存贮方案．

表 1 传统的稀疏存贮方案

Table 1 Traditional sparse storage scheme

Eq.	1				2		3			4		5		6
ICN				4		6			9		11		13	14
JCN	1	3	4	6	2	5	3	4	5	4	5	5	6	6
PCN	11	a	b	c	22	d	33	e	f	44	g	55	h	66

数组 $ICN(1:neq=6)$ 是非零元的列指标数组，$JCN(1:nzr=14)$ 以及它的数值数组 $PCN(1:nzr=14)$ 的行索引．第 k 行的非零元个数为 $ICN(k)-ICN(k-1)$(设 $ICN(0)=0$)．对于 $\boldsymbol{A}$ 的因子 $\boldsymbol{U}$ 或 $\boldsymbol{L}^{\mathrm{T}}$(简记为 $\boldsymbol{U}/\boldsymbol{L}^{\mathrm{T}}$) 可以采用相同的方案，但要注意它们的非零元个数与 $\boldsymbol{A}$ 的上三角部分不同．

在矩阵的三角分解过程中将产生许多新的非零元，一般称之为填充元 (fill-in's)．许多文献注意到，填充元的产生将使得 $\boldsymbol{U}/\boldsymbol{L}^{\mathrm{T}}$ 中的不少相邻的行的对角块是满上三角矩阵，其余的部分

具有完全相同的列指标集，或者说这些行具有相同的稀疏性 (sparsity). 这些的相邻行所对应的方程被定义为超方程 [3].

1.2 细胞稀疏索引存贮方案

文献 [3] 等定义的超方程是从一般的稀疏矩阵出发的，没有考虑结构有限元分析中一个节点具有多个自由度这一特征 (注意每个节点的自由度是连续编号的). 这个特征使有限元总体刚度矩阵 $\boldsymbol{A}$ 可以按节点形成分块矩阵. 为方便起见，我们重新定义超方程为同一节点所对应的全体方程，与超方程划分相应的子矩阵为细胞. 式 (1) 中 $\boldsymbol{A}$ 的第 2 个表达式就是按细胞方式写的. 在这一表示法之下，有限元的总体刚度矩阵被视作超行向量的顺序组合. 根据超方程的定义，式 (1) 中有 5 个超方程. 记 $mneq$ 为超方程数，我们用 5 个数组来描述式 (1) 中的第 2 个矩阵.

表 2 细胞稀疏存贮方案

Table 2 Cell sparse storage scheme

Super eq.	1			2		3		4		5
$SUPER$	1			2		4		5		6
IA			3		5		7		9	10
JA	1	3	5	2	4	3	4	5	6	6
LA			4		6		11		13	14
PA	11	$(a\ b)$	c	22	d	$\begin{pmatrix} 33 & e \\ & 44 \end{pmatrix}$	$\begin{pmatrix} f \\ g \end{pmatrix}$	55	h	66

数组 $SUPER(1:mneq=5)$ 是超方程编号到原方程编号的映射. 数组 IA 是非零细胞的超列指标 (或称细胞指标)JA 的索引，LA 是非零细胞数值数组 PA 的索引.

一般的稀疏存贮方案与本文建议的存贮方案之间的差异在于列指标的数量. 本文建议的存贮方案的列指标比传统存贮方案大大减少. 对于有许多 3×3 细胞的三维实体分析，采用本文建议的存贮方案，细胞指标数仅仅是传统方案列指标数的约 1/9. 以 4 字节整数和 8 字节实数计，总体刚度矩阵的大小可缩减约 1/3. 除此之外，对三维实体分析数组 IA,LA 和 $SUPER$ 的长度之和大致也等于 ICN 的长度. 对于有许多 6×6 细胞的三维壳体分析，则指标数相对更少.

2 细胞存贮方案与循环展开技术的稀疏矩阵分解

稀疏矩阵的三角分解可以分为两步：符号分解与数值分解. 其中，符号分解用于确定填充元的位置以及因子 $\boldsymbol{U}/\boldsymbol{L}^{\mathrm{T}}$ 的非零元分布. 正定矩阵的符号分解一般先于数值分解进行. 稀疏矩阵的符号分解可以在细胞意义下进行，它比通常意义下的符号分解方式节省了时间和存贮空间，对不超过 20 万自由度的三维有限元模型，细胞意义下的符号分解可以在 10MB 的内存里完成.

正定矩阵的三角分解 $\boldsymbol{LDL}^{\mathrm{T}}$ 的绝大部分工作量在于三重循环

$$A_{ji} = A_{ji} - L_{ik}L_{jk}D_{kk} \tag{2}$$

与上三角行存贮方案相适应的是所谓 JKI 分解. 根据矩阵代数的理论，按细胞分块矩阵的 JKI 分解与普通矩阵的 JKI 分解可以写成相似的形式 [16]

算法 1: 上三角行 $\boldsymbol{LDL}^{\mathrm{T}}$ 分解

```
DO J = 1, neq
    DO K = 1, J - 1
        RowTask(K, J)
    END DO
    RowTask(J, J)
END DO
```

算法 2: 上三角超行 $\boldsymbol{LDL}^{\mathrm{T}}$ 分解

```
DO MJ = 1, mneq
    DO MK = 1, MJ - 1
        SuperRowTask(MK, MJ)
    END DO
    SupeRowTask(MJ, MJ)
END DO
```

这里 $RowTask(K,J)$ 中包含的 I- 循环消去第 J 行第 K 列的矩阵元素，而 $SuperRowTask(MK,MJ)$ 中的 I- 循环包含了数个 $RowTask$ 的内容，它消去第 MJ 超行第 MK 超列的细胞. 当然，稀疏算法使用了链接技巧，限于篇幅不作叙述，有兴趣的读者可参阅文献 [3,4,7]. 算法 1 与算法 2 的差异在于最内层关于 I 的循环. 算法 1 中 $RowTask(K,J)$ 的 I- 循环仅包含式 (2) 一个语句. 而 $SuperRowTask$ 被写成了双向循环展开的形式，它的 I- 循环包含多个如下的语句

$$A_{ji}=A_{ji}-L_{i,mk}L_{j,mk}D_{mk,mk}-\cdots-L_{i,mk+n-1}L_{j,mk+n-1}D_{mk+n-1,mk+n-1} \tag{3}$$

其中 n 是超方程 mk 的子方程个数. 如果一个超方程包含的最大方程个数为 3，那么为了实现各种细胞尺度的 $SuperRowTask(MK,MJ)$，一共需要 9 种不同的具体消元格式，每种格式都对应于一个子程序. 考虑到自消元 $SuperRowTask(MJ,MJ)$ 的 3 个子程序，为了实施带循环展开技术的三角分解方程总共需要 12 个子程序. 如果对多个右端项的处理也用循环展开，那么还要增加 18 个子程序. 如果一个超方程包含的最大方程个数为 6，子程序的总数则为 78. 经过测定，这些子程序在求解过程可以使运算加快 3 至 5 倍. 在带有强大的编译程序的计算机上，例如 HP 工作站，可以不使用子程序而代之以嵌入式代码，其加速比可能更大 [9].

细胞稀疏方案的编程不仅比一维变列高方案复杂，而且也比传统稀疏矩阵复杂，整个解法 (包括总刚叠加，优化，细胞三角分解，向前消元与回代的 78 个子程序) 共约 22000 条 Fortran 源程序，其中的 19 000 条是各类 $SuperRowTask$ 的编码.

在文献 [3] 中，由于采用了传统的稀疏存贮方案，超方程是在 $\boldsymbol{U}/\boldsymbol{L}^{\mathrm{T}}$ 意义下定义的，只对已完成分解的因子 $\boldsymbol{U}/\boldsymbol{L}^{\mathrm{T}}$ 才能使用，即只能对算法 1 的循环 K 使用超方程. 所以，尽管文献 [3] 也用了循环展开，但它是算法 1 与算法 2 的杂交，即 J- 循环不展开，但 K- 循环展开，我们称之为单向循环展开，它的 I- 循环仅包含一个式 (3) 那样的语句. 数值实验表明，无论是满矩阵还是稀疏矩阵，我们采用的双向循环展开比文献 [3] 的单向循环展开效率更高.

由于总体刚度矩阵的大小往往超过内存的限度，与一维变列高存贮方案一样，细胞稀疏存贮方案可以采用分块外存技术. 按细胞存贮的稀疏矩阵的分块原则与一维活动列存贮没有太大的差别，只是要注意在考虑稀疏矩阵时，矩阵 $\boldsymbol{A}$ 和其因子 $\boldsymbol{U}/\boldsymbol{L}^{\mathrm{T}}$ 的分块不一样，并且 $\boldsymbol{U}/\boldsymbol{L}^{\mathrm{T}}$ 也不覆盖 $\boldsymbol{A}$，因为它们的非零元个数大不相同. 另外，还要考虑符号矩阵的分块.

3　数值试验

大量的实际的工程问题被用来检验本文提出的求解器方案，限于篇幅仅列出 8 个. 所有的

数值试验都在一台带 IDE 磁盘，操作系统为中文 Windows 98 的 Pentium III 550 机器上进行，编译器选用的是 Compaq Visual Fortran 6.1a. 代码采用了标准的 Fortran 90 语言，我们没有使用任何数值分析的库程序，它能容易地被移植到不同的计算机平台上. 本文测试的各种解法均在结构分析程序 SAP84 中实现 [13]. 解法 B 由文献 [9] 建议，并移植到了国家九五攻关项目《机械 CAE 系统的产业化开发》中.

数值试验采用的解法见表 3.

表 3 数值试验中使用的解法

Table 3 Solution approaches in the numerical tests

Solver	Storage scheme	Major operation	Unrolling	Optimization	Reference
A	column	dot-product	*none*	minimum front	[1,2,11,8]
B	skyline		two-way		[1,2,11,8,5,9]
C	row-sparse	DAXPY*	*none*	modified	[3]
D			one-way	minimum	[6,7]
E	cell sparse		two-way	degree	this paper

* DAXPY 是双精度 $Ax+y$ 的缩写.

* DAXPY is the shorting of double precision matrix operation $Ax+y$.

下表是各种解法的总体刚度及其因子的大小与求解时间的比较. 其中, 在考虑总刚大小时, 解法 A 与 B 的因子矩阵 $\boldsymbol{U}/\boldsymbol{L}^{\mathrm{T}}$ 覆盖了矩阵 $\boldsymbol{A}$ ，它们只记一次. 而解法 C, D 与 E 的大小是矩阵 $\boldsymbol{A}$ 与 $\boldsymbol{U}/\boldsymbol{L}^{\mathrm{T}}$ 之和.

表 4 几种解法的比较

Table 4 Comparison of these Solution Approaches

Example	Number of equation	File size of global stiffness matrix (MB)			Factorization time (reduction time) (s)				
		A,B	C,D	E	A	B	C	D	E
botanical hall[14]	22 044	117	38	25	105.86 (1.89)	61.89 (1.80)	10.48 (0.62)	5.74 (0.37)	2.59 (0.31)
group container 1	33 960	189	84	56	280.54 (5.61)	155.87 (4.68)	68.54 (2.97)	36.98 (1.68)	14.64 (1.52)
group container 2	63 336	574	137	91	1164.92 (17.13)	651.81 (14.46)	97.99 (4.67)	51.73 (3.01)	21.45 (2.63)
4-core container	80 676	1 082	278	186	3 813.24 (41.10)	2 175.12 (31.48)	309.14 (11.03)	171.73 (7.43)	66.11 (6.28)
cofferdam	87 804	1 298	647	427	6 590.02 (53.17)	3 803.53 (41.25)	3 754.88 (28.64)	1 899.33 (16.22)	669.83 (14.57)
tall building	55 590	415	190	135	762.63 (11.25)	424.61 (9.71)	487.70 (7.64)	262.60 (4.90)	110.06 (3.97)
twin tower building	94 653	1 173	299	201	6 189.38 (49.03)	3 509.86 (40.08)	396.74 (6.81)	230.42 (4.69)	117.29 (5.64)
machine element	94 893	1 635	591	394	8 078.32 (35.71)	5 118.22 (34.93)	2 417.32 (15.37)	968.14 (11.00)	455.99 (9.56)

对于二维问题，本文建议的求解方案也显著地改进了直接求解法的效率，但由于二维问题本身的计算工作量比三维问题小许多，限于篇幅就不举例了.

4 结　论

对于工程有限元分析，本文提出了细胞稀疏存贮方案．它的外存和内存占用量是一维变列高存贮的 1/5 到 1/3，是传统的稀疏存贮方案的 7/10．即使与 Sherman 的紧凑存贮方案[12] 比较，本文建议的存贮方案需要内外存空间要小一些，而且更容易操作．细胞存贮方案非常适合在稀疏矩阵的三角分解中将 J- 循环和 K- 循环展开．前人的工作中，由于仅对 $\boldsymbol{U}/\boldsymbol{L}^{\mathrm{T}}$ 引入了超方程的概念，只有 K- 循环被展开了．以求解时间与磁盘占用衡量，建议的稀疏技术和循环展开技术结合在一起达到了非常高的效率．与广泛采用的一维变列高方案比较，求解速度提高了数倍到数十倍．与传统的稀疏存贮方案比，求解速度也提高了四、五倍．与只采用单向 K- 循环展开的算法相比，求解速度提高了约两倍．

求解速度的提高来源于以下几个方面： 1) 采用了稀疏存贮方案的三角分解的计算量大大小于一维变列高方案的计算量； 2) 循环展开减少了 RAM 与 Cache 之间的通讯，提高了数据的中靶率，即调入 Cache 的数据的使用次数； 3) 细胞存贮方案减少了索引的数目，也减少了索引操作的数量； 4) 细胞存贮方案下矩阵的文件较小，使得 I/O 减少．

本文提出的快速有限元解法可以在各类特征值算法与非线性解法中替代传统的一维变列高直接解法．虽然本文的方法已达到了相当高的求解速度，但我们认为仍有改进的余地．本文方法的进一步研究必将促进有限元分析的效率的提高．

致谢　第一作者对崔俊芝院士的鼓励表示感谢．

参　考　文　献

1 Bathe KJ. Finite Element Procedures, Prentice-Hall, Englewood Cliffs, New Jersey, 1996

2 Zienkiewicz OC, Taylor RL. The Finite Element Method (4^{th} ed.). McGraw-Hill, 1989/1990, 1/2

3 Nguyen DT, Qin J, Chang TYP, Tong P. Efficient sparse equation solver with unrolling strategies for computational mechanics. Proceeding of the ICES'97 Conference, Costa Rica, May 4~9, 1997. 676~681

4 Chen P, Runesha H, Nguyen DT, Tong P, Chang TYP. Sparse algorithms for indefinite systems of linear equations. *Computational Mechanics Journal*, 2000, 25(1): 33~42

5 张林波．利用 m4 宏语言进行 Fortran 77 循环展开．数值计算与计算机应用， 1998, 19(1): 49~63. (Zhang LB. Fortran do loop unrolling using m4 macro-language. *Journal of Numerical Methods and Computer Application*, 1998, 19(1): 49~63 (in Chinese))

6 George A, Liu WH. Computer Solution of Large Sparse Positive Definite Systems. Prentice-hall, Englewood Cliffs, NJ, 1981

7 Duff IS, Erisman AM, Reid JK. Direct Methods for Sparse Matrices. Clarendin Press, Oxford, 1989

8 Hoit M, Wilson EL. An equation numbering algorithm based on a minimum front criteria. *Computers and Structures*, 1983, 16(1-4): 225~239

9 郑东．私人通信． 1995 (Zheng D. private communication. 1995)

10 Lewis GJ, Simon HD. The impact of hardware gather/scatter on sparse gaussian elimination. Proceedings of the 1986 International Conference on Parallel Processing (Hwang K, Jacobs S, Swartzlander E eds.), IEEE Computer Society, Los Angeles, 1986

11 Wilson EL, Bathe KJ, Doherty WP. Direct Solution of Large System of Linear Equations. *Computers and Structures*, 1974, 4: 363~372

12 Sherman AH. On the Efficient Solution of Sparse Systems of Linear and Nonlinear Equations, Report No.46, Dept of Computer Science, Yale University, 1975

13 袁明武等. 微机结构分析程序SAP84 版本 3.2(plus). 北京大学， 1991 (Yuan MW et al. Structural Analysis Program SAP84 Version 3.2(plus). Peking University, 1991 (in Chinese))

14 陈斌，陈璞，孙树立，袁明武，陆承康，盛平．北京植物园展览温室结构的静动力分析与研究．建筑结构， 2000, 30(1): 53~56 (Chen B, Chen P, Sun SL, Yuan MW, Lu CK, Sheng P. Static and dynamic analysis of public plant conservatory at beijing botanical garden. *Building Structure*, 2000, 30(1): 53~56(in Chinese))

15 Felippa CA. Recent Developments in Basic Finite Element Technologies in Computational Mechanics in Structural Engineering——Recent Development. Cheng FY and GU YX eds. Elsevier, 1999

16 北京大学数学系代数与几何教研室. 高等代数. 北京：高等教育出版社， 1978 (Algebra and Geometry Division of Peking University. Advanced Algebra. Beijing: Higher Education Publisher, 1978(in Chinese))

FAST SOLUTION ALGORITHM IN FINITE ELEMENT ANALYSIS[1)]

Chen Pu Sun Shuli Yuan Mingwu

(*Department of Mechanics and Engineering Science, Peking University, Beijing* 100871, *China*)

Abstract In early nineteen seventy's, the skyline active column algorithm is considered as an efficient one for $\boldsymbol{LDL}^{\mathrm{T}}$ or Cholesky factorization of the positive definite sparse linear equations $\boldsymbol{Ax} = \boldsymbol{f}$. But beneath the skyline there are usually many zeros in almost each column. All these zeros occupied the storage allocations and participated operations. These zeros caused also extra I/O between the core and the secondary memory. As the size of linear equations increasing, the skyline approach encounters difficulties of storage shortage and lower efficiency. Comparing to conventional skyline and half-bandwidth storage schemes, the sparse storage scheme needs only allocations for non-zero entries of the stiffness matrix and the whole matrix is manipulated in a very smart way. Therefore sparse solver requires less disk space and is faster than conventional direct solvers with skyline or half-bandwidth storage scheme for large-scale problems.

In reference [3] the super-equation was introduced to sparse matrix computation with the conventional sparse storage scheme for the factor $\boldsymbol{L}^{\mathrm{T}}$ of $\boldsymbol{K}$ in order to perform one-way unrolling. In this paper, we extent the super-equation concept to the global stiffness matrix $\boldsymbol{A}$ of three dimensional FEA and proposes cell sparse storage scheme. The proposed cell sparse storage scheme provides an excellent data structure for two-way loop unrolling, to which we unrolled two outer loops of the triply nested loop in so-called JKI $\boldsymbol{LDL}^{\mathrm{T}}$ factorization. This essential changes led to a big improvement in efficiency and the disk space requirement. The cell sparse storage reduced about 70% of core memory as well as disk space requirement for engineering FEA in comparison with the skyline storage scheme. Even compared to conventional sparse storage scheme, 30% of memory requirement was reduced. The combination of the cell sparse technology and loop unrolling in the new solution strategy achieved very high efficiency in sense of elapsed time and disk space requirement. Eight examples showed that the proposed algorithm was 7~57 times faster than the skyline solver and the disk space requirement reduced to 1/3 for engineering FEA.

The higher speed came from the following aspects: 1) the number of operations of the $\boldsymbol{LDL}^{\mathrm{T}}$ factorization in the sparse storage scheme is much less than that in the skyline scheme; 2) the loop unrolling reduces the data traffic between RAM and cache, and even registers; 3) there are less indices in the cell sparse storage scheme, thus the index manipulations decreased; and 4) smaller disk and core space requirement reduces the I/O.

Key words high performance computing, sparse matrix, finite element analysis, static solver

Received 20 March 2000, revised 16 March 2001.

1) The project supported by the Science Foundation of Returned Overseas Chinese Scholars Ministry of Education and the National Natural Science Foundation of China (10172005).