

# 用环形链表实现整刚矩阵的动态存贮

主承前 王全凤

(土木工程系)

**摘要** 用环形链表和 C 语言,实现有限元整刚矩阵的动态存贮,以及子结构凝聚和设计不设数组的解方程程序,因而避免传统存贮方法数组大小与元素多少不匹配的矛盾,实现了内存空间的按需分配。

**关键词** 环形链表,动态存贮,整刚矩阵

## 0 引言

我们在从事有限元整刚矩阵的存贮过程中,深感目前广泛使用的静态存贮方法的不足。例如,用一维变带宽存贮如果需要子结构凝聚,则需将交界面变量的元素集中在一起,将会彻底打乱各行主对角元素在一维变带宽存贮数组中的地址,恢复起来需做大量计算。在凝聚过程中,新生成的非零元素无法插入原来的数组中,必须为之另辟存贮空间,破坏了数据存贮的完整性。象二维等带宽、索引、超矩阵等存贮方法亦有类似问题。这种通病是在存贮元素之前,必须预先估计最多可能有多少元素,据此确定数组的大小,这就大大降低了程序的通用性。所述矛盾促使寻找一个新的存贮方法,既能不存零元素以节约内存,又可在凝聚子结构的过程中,方便地换行换列以集中交界面变量,同时在凝聚过程中新生成的非零元素可以方便地插入原存贮空间中。为此,我们用数据结构中的环形链表,用 C 语言实现了整刚矩阵的动态存贮,并运用于一维杆系有限元和二维等参有限元中。这种新的存贮方法不仅具有上述特点,而且存取数据不需计算,不存任何零元素,运行速度快。

本文的环形链表存贮用 C 语言加以实现,为方便数据存取和子结构的凝聚,设计了7个精巧而实用的 C 函数,包括制表头函数、元素插入函数、删除函数、引入支座条件函数、换行换列函数、凝聚函数以及解方程函数。这7个函数不依赖于任何一种单元形式。有了这7个函数,即使不懂 C 语言和动态存贮的人,只要弄懂函数中每个参量的意义,就可以很方便地实现动态存

本文1992-04-21收到。

贮和凝聚。

## 1 基本原理

链表是数据结构中的概念,其结点一般有两部分组成.一部分表示数据,另一部分表示指向下一个结点的指针.将结点用指针连接起来即构成一个链表,环形链表是指链表最后一个结点的指针指向表头结点的地址.C语言本身具有指针数据类型,结点可以用C语言的结构加以描述,因而用C语言描述环形链表是很方便的.用环形链表存贮整刚矩阵时,对矩阵中的每一行非零元素,构成一个行环形链表,对每一列中的非零元素,亦构成一个列环形链表.这样需要设置两个链场,一个链指向同行中的下一个非零元素,一个链指向同列的下一个非零元素.每一个结点表示的信息如下所示.

结点信息表

row	col	value
down		right

其中 row, col 分别表示元素所在的行和列,value 表示元素值.right, down 分别表示指向同一行和同一列中下一个非零元素地址的指针.

为了查找元素的方便,在各行各列之首建立表头结点,列表头的 down 指向下一行的表头,right 指向同行中第一个非零元素的地址.该行最后一个结点的 right 指向行表头地址.列表头的 right 指向下一列的表头结点,列表头的 down 指向同列中第一个非零元素的地址.该列最后一个元素的 down 指向该列表头结点.所有行表头和列表头的结点的 value 均为零.为适应环形链表的特点,须在行表头和列表头之首设公共表头结点,其 value 为零,其 down 指向第一行表头,最后一行表头结点的 down 指向公共表头.公共表头的 right 指向第一列表头,最后一列表头的 right 指向公共表头.零矩阵由公共表头和行表头以及列表头形成的环形链表表示.在存贮整刚矩阵时,根据整刚矩阵的阶,采用制表头函数,形成一个同阶的零矩阵,然后调用插入函数将形成的整刚矩阵元素插入已形成的矩阵中.对于引入支座条件,集中交界面变量,以及非交界面变量的凝聚,都可以编制C语言的函数加以实现,以得到子结构刚度矩阵.在本程序设计中,广泛使用了指针的概念,对于指针的正确理解是程序设计的关键.

## 2 程序设计

首先要定义一个能表示如上所示结点信息表的数据类型.用C语言的结构数据类型加以表示,即

```
typedef struct matnode {int row, col; struct matnode * down, * right; float value} MATNODE; 其中,down 和 right 分别是指向 matnode 结构的指针.
```

在各子程序(函数)之前,要有一段如下预处理程序,即: #define getnode(type)(type \*) malloc (sizeof(type)). 这是一个宏定义,它调用C语言的标准I/O库函数 malloc( ) 分配一个新增结点的地址,并使 getnode( ) 函数值为指向该结点的指针.实际上正是这个函数实现了动态内存分配,每当增加一个元素时,就可用 getnode( ) 函数为其分配一个地址.每当减少一个元素时,可调用C语言的I/O库函数予以删除.则该元素将不占内存.具体实施需要下

面的函数加以实现.

### 2.1 制表头函数 makehead(nuvd)

该函数形成  $nuvd \times nuvd$  阶的零矩阵,即只含有行和列的表头以及公共表头的环形链表,并返回公共表头的指针  $mat$ ,其中  $nuvd$  是该函数的参量,它表示整刚矩阵的阶.在制表头函数  $makehead()$  中,首先定义  $makehead()$  函数的返回值为指向公共表头的指针,然后形成公共表头,再形成行表头环形链表和列表头环形链表.在形成整刚矩阵以前,该表是空表,占内存很少.

### 2.2 插入函数 insert(mat, i, j, val)

在这个函数中,参量  $mat$  表示公共表头的地址,有了它,就可以查找整个环形链表,以下  $mat$  所代表的意义与此相同. $i, j$  表示元素在矩阵中所在的行和列,  $val$  表示元素的值.  $insert()$  函数调用  $malloc()$ ,其功能是,如果环形链表在该位置没有元素,则插入此元素;如果已有元素,则将其值迭加在一起,这符合形成整刚矩阵的原则.

### 2.3 删除函数 kill(mat, i, j)

该函数将第  $i$  行第  $j$  列元素删除,并保持环形链表的完整.若第  $i$  行第  $j$  列没有元素,则直接返回.若有此元素,则寻找离该元素最近的上下左右4个结点,并使其上面结点的  $down$  指向其下面的结点,其左面结点的  $right$  指向其右面结点.最后调用 C 语言的 I/O 库函数  $free()$  将该元素删除.

### 2.4 引入支座条件函数 incon(mat, i, nuvd)

其中  $nuvd$  表示整刚矩阵的阶,  $i$  表示支座条件所代表的位移在整刚矩阵的行和列.该函数仅用于引入固定支座条件,引入的办法是将整刚矩阵中与支座条件相关的行和列的主对角元素置1,非主对角元素置零,并调用  $kill()$  函数,将零元素从链表中清除.

### 2.5 换行换列函数 exchang(mat, i, j, nuvd)

为了凝聚的方便,须将子结构整刚矩阵的界面变量集中到整刚矩阵的上部或一部.为此需要进行行列的对调,函数  $exchange()$  实现将  $i$  行与  $j$  行,  $i$  列与  $j$  列的对调,方法是恰当地引用插入和删除函数.

### 2.6 凝聚函数 condense(mat, nc, nuvd)

该函数假定将界面变量所对应的行和列置于整刚矩阵的右下部,  $nc$  表示界面变量的个数.又假定非界面变量所在结点不受外力,即右端项为零.本函数采用高斯消元法进行凝聚.

### 2.7 解方程函数 eq(mat, nuvd, f)

若已知作用于各结点力,则可解刚度矩阵方程,  $nuvd$  为方程的阶数,  $f(nuvd+1)$  为存放外力的数组,  $f(0)$  未赋值,  $f(i)$  为对应于第  $i$  位移的结点力 ( $i=1, 2, \dots, nuvd$ ). 函数  $eq()$  采用高斯消元法解方程.

上述7个函数所实现的环形链表存贮和整刚矩阵的凝聚以及方程式求解,已被作者应用于平面杆系有限元和平面等参有限元中.同传统存贮方法相比,该方法节约内存,运行速度快.其计算结果也是可靠的.

## 参 考 文 献

- [1] 王广芳、曹兰斌、黄考慈,数据结构,国防科技大学出版社,(1982).  
[2] 王勖成、邵敏,有限单元法—基本原理与数值方法,清华大学出版社,(1988).  
[3] 姜文清,数据结构与算法—C 语言程序设计,上海交通大学出版社,(1988).  
[4] 刘甲耀、严桂兰,PAD 编程方法与 C 语言程序设计,电子工业出版社,(1989).

## Dynamic Storage of Global Stiffness Matrix Implemented by Applying a Circular List

Zhu Chengqian Wang Quanfeng

*(Department of Civil Engineering)*

**Abstract** By applying a circular list in the data structure, the dynamic storage of global stiffness matrix with finite elements is implemented with C language. Based on this storage scheme, the condensation of minor structures and the program for solving the equations are designed. In view of the failing of traditional storage scheme which failed in the improper coordination between the size of array and the number of elements, no array is offered in this storage scheme. Consequently, the memory space can thus be allocated according to demand.

**Key words** circular list, dynamic storage, global stiffness matrix