1. 右端项消去

   (a) 令 $v_1 = r_1$

   (b) 对 i 从 2 到 n 循环，计算 $v_i = r_i - \sum_{j=m_i}^{i-1} l_{ji} v_j$

   (c) 对 i 从 1 到 n 循环，计算 $\bar{v}_i = v_i / d_{ii}$

2. 回代求解

   (a) 令 $a_n = \bar{v}_n$

   (b) 对 j 从 n 到 2 循环

   - 对 i 从 m_j 到 $j-1$ 循环，计算 $a_i = \bar{v}_i - l_{ij} a_j$

类似地，程序中也无需显式地执行 $v_1 = r_1$ 和 $a_n = \bar{v}_n$ 操作。活动列求解法的 matlab 程序参见示例程序 colsol.m。

瑞士 Olaf Schenk 教授和德国 Klaus Gartner 博士研发了一个具备线程安全性的高性能大规模稀疏对称和非对称线性方程组求解器 PARDISO 软件包 (http://www.pardiso-project.org)，可运行于共享内存体系结构计算机和分布式内存体系结构计算机，具有很好的可扩展性（使用 8 个处理器可以获得接近于 7 的加速比）。Intel 的数学核心函数库 (Math Kernel Library，MKL) (https://software.intel.com/en-us/intel-mkl/) 集成了 PARDISO 求解器，并对其进行了优化，同时支持内存求解方式和外存求解方式。

在进行 LDLT 分解时，轮廓内的零元素将会变为非零元素，称为填充元 (fill-in)。有限元法的总体刚度矩阵是高度稀疏矩阵，存在大量的填充元。矩阵 $\boldsymbol{K}$ 和矩阵 $\boldsymbol{L}$ 具有完全相同的轮廓，因此采用一维变带宽存储时，LDLT 分解无需再分配额外的内存，分解后的 $\boldsymbol{L}$ 矩阵和 $\boldsymbol{D}$ 矩阵将占用矩阵 $\boldsymbol{K}$ 的存储区域。但是，列压缩等存储方案只存储了非零元素，在 LDLT 分解时需要为填充元分配内存。对于大规模问题，填充元有可能导致内存溢出，因此利用带宽优化算法尽可能降低带宽是必要的。PARDISO 自动会进行带宽优化，以尽可能减少填充元。

2.5 有限元法的程序实现

与其他分析方法相比，有限元法具有很强的通用性，可以分析几乎任何具有复杂边界和加载条件的连续介质问题。有限元程序的效率除了取决于程序所使用单元和算法外，还取决于所采用的程序设计方法。

现有有限元商用软件大部分都是采用面向过程的 FORTRAN 语言编写的。C++ 语言是一种面向对象的语言，支持数据封装和数据隐藏，支持继承和重

用，支持多态性。C++ 程序具有很好的可维护性、易修改性和复用性，因此近年来许多大型科学计算软件都采用或者改用 C++ 语言。例如，开源计算流体力学软件 OpenFOAM[3] 是用 C++ 语言开发的，而分子动力学模拟开源程序包 LAMMPS[4] 早期先后采用 Fortran 77 和 Fortran 90 语言开发，2004 年后改用 C++ 开发维护。

STAP 程序是 K. J. Bathe 用 FORTRAN IV 编写的教学示例程序[5]，其程序结构与 SAP 和 ADINA 的程序结构类似。STAP90 程序是张雄用 FORTRAN 90 语言对 STAP 程序改写而成的，作为《计算动力学》教材[6,7] 的教学程序。本书采用 C++ 语言介绍有限元法的程序实现方法，并提供了用 C++ 语言编写的基于面向对象思想设计的有限元法程序。为了便于学生在进行课程编程训练时自主选择编程语言，该程序采用了与 STAP90 程序完全相同的输入和输出文件格式，因此被称为 STAP++。STAP++ 程序的最新发行版本可以在 GitHub 上获取，仓库地址为 git@github.com:xzhang66/stappp.git。

本节以 STAP++ 程序为例论述基于面向对象思想的有限元程序实现方法。该程序的目的是为了说明基于面向对象思想的有限元程序实现方法，因此只提供了轴力杆单元，学生可在本程序的基础上进行扩充完善，完成课程编程训练。基于 FORTRAN 语言的有限元程序设计方法可以参考《计算动力学》教材[6,7]。

STAP++ 程序只涉及到有限元法五个步骤中的中间三步，即单元分析、总刚组装和方程求解，而前处理和后处理则需要借助其它专用软件进行。有限元商用软件都具有功能很强大的前后处理系统，另外也有其它独立的前后处理软件，如 Gmsh[8]、HyperMesh[9]、GiD[10]、Tecplot[11] 和 ParaView[12] 等。学生可以扩充 STAP++ 程序，将其它前处理软件生成的有限元模型数据文件转换成可供 STAP++ 程序读入的输入数据文件，并将其计算结果按照 Tecplot 或 ParaView 规定的格式保存为文件，然后利用 Tecplot 或 ParaView 进行结果可视化处理。《计算动力学 (第 2 版)》[7] 的附录 B 和 C 分别介绍了如何利用 Tecplot 和 ParaView 进行有限元后处理。

2.5.1 有限元模型数据

有限元模型由结点、单元、截面/材料属性和载荷等组成。STAP++ 程序用 CDomain 类对有限元模型数据进行封装，用 CNode 类封装结点数据，用抽象类 CElement 封装单元数据，用抽象类 CMaterial 封装截面/材料属性数据，用 CLoadCaseData 类封装载荷数据，用 CSkylineMatrix 封装按一维变带宽方式存储的总体刚度矩阵。另外，为了方便，STAP++ 程序将所有单元分为若干个单元组，每个单元组只允许具有一种类型的单元。单元组数据用 CElementGroup

封装。

首先考虑与结点相关的数据。假定程序允许每个结点最多可以有 CNode :: NDF = 6 个可能的自由度，包括 3 个平动自由度和 3 个转动自由度，如图2.11所示。每一个结点必须明确各自由度边界条件类型，即哪些自由度是自由的（将进入到总体自由度中），哪些自由度是约束的（不进入到总体自由度中）。例如对于位于 xy 坐标面内的平面桁架，每个结点最多只有两个自由度，即只需使用各结点的第 1 个和第 2 个自由度。这可以通过为每个结点定义边界条件标示数组 bcode[NDF] 来实现。如果 bcode[i] = 0，则该结点的第 i 个方向自由，该自由度将出现在总体自由度中。如果 bcode[i] = 1，则该结点的第 i 个方向固定，该自由度不出现在总体自由度中。使用标示数组 bcode[i] 后，每个结点的自由度数是可以变化的，如在同一个结构中，薄膜上的结点只有 2 个结点自由度，板上的结点有 3 个自由度，而壳上的结点则有 5 或 6 个自由度。

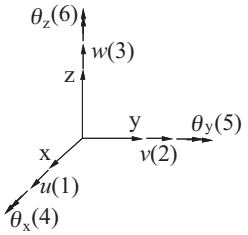

图 2.11: 结点自由度

例如，图2.12所示的平面桁架位于 xy 平面内，每个结点只有 x 和 y 方向的位移，即各结点只有 x 和 y 方向的自由度。结点 1 和 6 是固定的，它们的位移为零，这 2 个结点自由度不进入到总体自由度中，故该结构共有 22 个自由度，总体位移向量 $\boldsymbol{d}$ 为

$$\boldsymbol{d} = [\ u_2 \quad v_2 \quad u_3 \quad \cdots \quad v_5 \quad u_7 \quad v_7 \quad \cdots \quad u_{12} \quad v_{12}\ ]^{\mathrm{T}} \tag{2.61}$$

STAP++ 目前没有梁板壳单元，因此取 CNode :: NDF = 3。各结点的标识数组（每列对应于一个结点）为

$$\text{bcode} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.62}$$

数组 bcode 的内容给定后，可以依次扫描每个结点的各自由度（即按列扫描 bcode），将 bcode 的每一个零元素替换为相应的方程号，同时把 bcode 的其

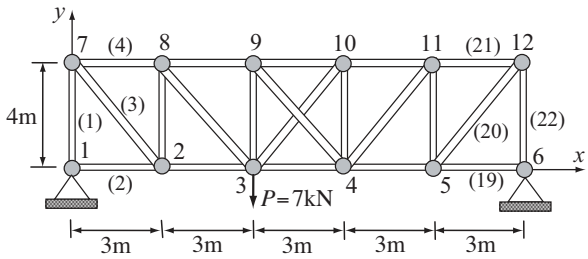

图 2.12: 平面桁架

他元素置为零，就可以得到各结点自由度对应的总体自由度（方程）号。相应的代码见 CDomain 类的成员函数 void CalculateEquationNumber()。执行该函数后，bcode 数组变为

$$\text{bcode} = \begin{bmatrix} 0 & 1 & 3 & 5 & 7 & 0 & 9 & 11 & 13 & 15 & 17 & 19 \\ 0 & 2 & 4 & 6 & 8 & 0 & 10 & 12 & 14 & 16 & 18 & 20 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.63}$$

除了定义 bcode 数组外，还需要读入各结点的 x、y、z 坐标。对图2.12所示的系统，坐标数组 XYZ （每列对应于一个结点）为

$$\text{XYZ} = \begin{bmatrix} 0 & 3 & 6 & 9 & 12 & 15 & 0 & 3 & 6 & 9 & 12 & 15 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 4 & 4 & 4 & 4 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

已知所有结点信息后，程序需要读入单元信息。描述一个单元所需的数据取决于单元的具体类型。描述一个单元所需的数据包括单元的结点编号、单元的截面/材料属性以及施加在该单元上的面力或体力等。一般情况下，多个单元具有相同的单元截面/材料属性和单元载荷，因此可以先定义单元截面/材料属性组和单元载荷组，而对每一个单元只要给出其截面/材料属性组号和载荷组号即可。

对于如图2.12所示的系统，所有杆的截面面积均为 $A = 10^{-2}\text{m}^{-2}$，弹性模量均为 $E = 1.5 \times 10^{11}\text{Pa}$，因此只需定义一个截面/材料属性组。各单元的定义

为

单元号	左结点号	右结点号	截面/材料属性组号
1	1	7	1
2	1	2	1
3	2	7	1
⋮	⋮	⋮	⋮
22	6	12	1

在有限元法中，载荷既可以作用在单元上，也可以直接作用在结点上，但最终都会转化为结点载荷。STAP++ 目前只允许输入结点载荷，作用在单元上的载荷需要由用户自己转化为结点载荷。本问题只有一个载荷工况，且集中载荷作用在结点 3 的 y 方向，因此载荷组可以定义为

受载荷作用的结点号	载荷作用方向	载荷值
3	2	−7000

2.5.2 STAP++ 程序的输入数据文件格式

STAP++ 的输入数据文件包括以下五部分（需严格按照下面给出的先后顺序填写）：

1. 标题行，用于对所求解问题进行简单的描述；

2. 控制行，依次填写 NUMNP（结点总数）、NUMEG（单元组总数，每个单元组只能包含相同类型的单元）、NLCASE（载荷工况数）、MODEX（求解模式，等于 0 时只做数据检查，等于 1 时进行求解）；

3. 结点数据，依次填写结点号、x-平动方向边界条件代码（0 为自由，1 为固定）、y-平动方向边界条件代码、z-平动方向边界条件代码、x-坐标、y-坐标和 z-坐标。结点数据必须从 1 到 NUMNP 按结点编号顺序填写；

4 载荷数据，共输入 NLCASE 组载荷数据，每组包括：

(1) 载荷数据控制行，依次填写载荷工况号和本工况中集中载荷的个数；

(2) 各工况载荷数据，依次填写集中载荷作用的结点号、载荷作用方向（1 –x 方向，2 –y 方向，3 –z 方向）和载荷值；

5. 单元组数据，输入共 NUMEG 组单元，每组包括：

(1) 单元组控制数据，依次填写单元类型 (1 –轴力杆单元)、本单元组中的单元总数 NUME 和截面/材料属性组数 NUMMAT；

(2) 截面/材料属性数据，共读入 NUMMAT 行，依次填写截面/材料属性组号、杨氏模量和截面面积；

(3) 单元数据，共填写 NUME 行，依次填写单元号、单元左端结点号、单元右端结点号和本单元的截面/材料属性组号；

STAP90 对各数据项所在列有严格限制，例如控制行第 1-5 列填写 NUMNP、第 6-10 列填写 NUMEG、第 11-15 列填写 NLCASE、第 16-20 列填写 MODEX。STAP++ 解除了对数据项所在列的限制，只要数据之间用空格（或制表键）隔开即可，因此 STAP++ 可以直接读入 STAP90 的输入数据文件，但 STAP90 不一定能直接读入 STAP++ 的输入数据文件。

2.5.3 结构总体矩阵的存储

STAP++ 程序从输入数据文件中读入有限元模型数据，计算每个单元的刚度矩阵并组装生成总体刚度矩阵，然后求解总体刚度方程，并将结果保存到数据文件中。STAP++ 程序的流程如图2.13所示。

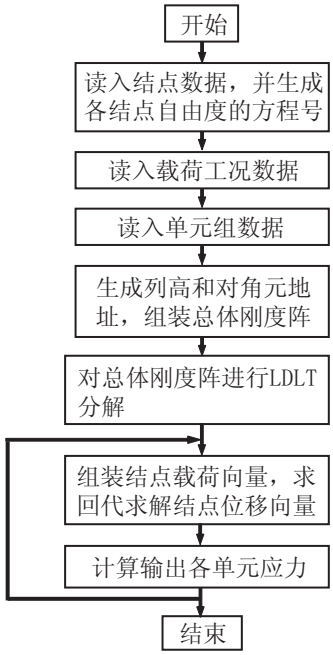

图 2.13: STAP++ 程序流程图

在组装总体刚度矩阵时，需要利用每个单元自由度所对应的总体自由度（方程）号。STAP++ 程序在 CElement 类中定义了单元位置数组 LocationMatrix 来存储单元各自由度所对应的总体自由度号。例如，对于如图2.12所示的系统，

单元 3 的 1 号和 2 号结点分别对应结构总体的 2 号和 7 号结点，由 bcode 数组可以得到单元各自由度所对应的总体自由度号，因此该单元的位置数组为

$$\text{LocationMatrix} = [\ 1 \quad 2 \quad 0 \quad 9 \quad 10 \quad 0\ ]$$

生成位置数组 LocationMatrix 的代码见 CElement 类的成员函数 void GenerateLocationMatrix()。

在有限元法中，每个单元只与很少几个结点相连，因此结构的总体刚度矩阵是高度稀疏的，存在大量的零元素 (如图2.14所示)。在有限元程序中，为了提高计算效率和计算规模，一般都采用特殊的存储格式来存储结构总体刚度矩阵，如一维变带宽存储格式 (Skyline Storage Format)、行压缩格式 (Compressed Sparse Row Format，CSR)、列压缩格式 (Compressed Sparse Column Format，CSC) 和坐标存储格式 (Coordinate Format、COO) 等。下面只介绍一维变带宽存储格式和行压缩格式。

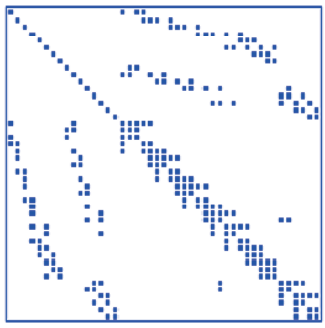

图 2.14: 典型的稀疏刚度矩阵

2.5.3.1 一维变带宽存储格式

图2.15(a) 所示为一个典型的结构总体刚度矩阵 $\boldsymbol{K}$。对称矩阵一维变带宽存储方案是用一个一维数组 A 按列 (或行) 依次存储结构总体刚度矩阵的上三角阵，每列从主对角元素 k_{ii} 开始直到最高的非零元素 $k_{m_i i}$，即该列中行号最小的非零元素为止。图2.15(b) 给出了矩阵 $\boldsymbol{K}$ 的各元素在数组 A 中的具体位置。STAP++ 采用 CSkylineMatrix 类封装以一维变带宽方式存储的总体刚度矩阵。

用 m_i 表示矩阵 $\boldsymbol{K}$ 第 i 列第 1 个非零元素的行号，它给出了矩阵 $\boldsymbol{K}$ 的轮廓线。$(i - m_i)$ 为矩阵 $\boldsymbol{K}$ 第 i 列的列高，各列的列高是不同的。在对矩阵 $\boldsymbol{K}$ 进行 LDLT 分解时，轮廓线外的零元素在分解过程中仍然为零，但轮廓线内的

零元素在分解过程中可能变为非零，因此无需存储矩阵 $\boldsymbol{K}$ 轮廓线外的所有零元素，但需存储轮廓线内的零元素。

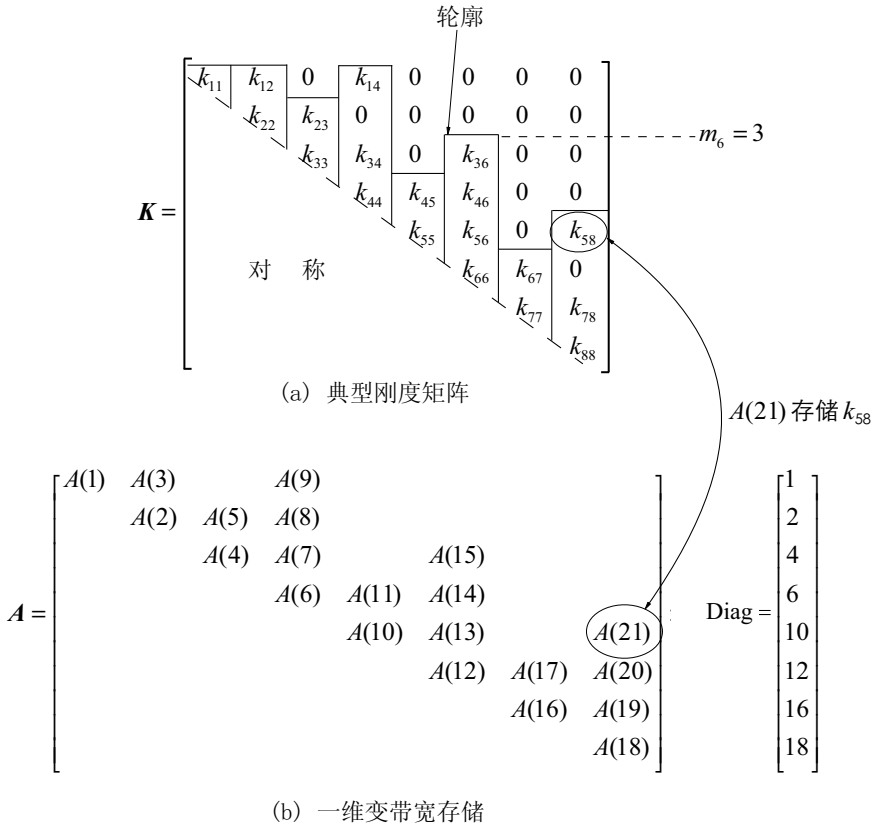

图 2.15: 刚度矩阵的一维变带宽存储方案

列高可以通过单元的位置数组 LocationMatrix 来确定。例如对于如图2.12所示的系统，由各单元的位置数组 LocationMatrix 可知，只有单元 1、3 和 4 与结构的第 10 个自由度有关，而这三个单元的位置数组中最小的自由度号是 1，因此 $m_{10} = 1$，列高为 9。再如，只有单元 20、21 和 22 与结构的第 20 个自由度有关，而这三个单元的位置数组中最小的自由度号是 7，因此 $m_{20} = 7$，列高为 13。同理可得到各列的列高为

$$0, 1, 2, 3, 2, 3, 2, 3, 8, 9, 10, 11, 10, 11, 12, 13, 12, 13, 12, 13$$

计算列高的代码见 CSkylineMatrix 类的成员函数 void CalculateColumn-

Height(unsigned int* LocationMatrix, size_t ND)，其中 ND 为单元自由度总数。

为了快速找出矩阵 $\boldsymbol{K}$ 的各元素在数组 A 中的位置，可以定义一个数组 Diag 来存放矩阵 $\boldsymbol{K}$ 的各对角元在数组 A 中的位置，即矩阵 $\boldsymbol{K}$ 的对角元 k_{ii} 在数组 A 中的地址为 Diag(i)。由图2.15(b) 可以看出，Diag(i) 等于矩阵 $\boldsymbol{K}$ 的前 $(i-1)$ 列的列高的总和再加上 i，因此矩阵 $\boldsymbol{K}$ 中第 i 列非零元素的个数等于 Diag$(i+1)-$Diag(i)，这些元素在数组 A 中的地址分别为 Diag(i)，Diag$(i)+1$，Diag$(i)+2$，$\cdots$，Diag$(i+1)-1$。

对于如图2.12所示的系统，刚度矩阵 $\boldsymbol{K}$ 各对角元在数组 A 中的地址分别为

$$1,2,4,7,11,14,18,21,25,34,44,55,67,78,90,103,114,130,144,157,171$$

计算刚度矩阵对角元地址的代码见 CSkylineMatrix 类的成员函数 void CalculateDiagnoalAddress()。组装结构总体刚度矩阵的代码见 CSkylineMatrix 类的成员函数 void Assembly(double* Matrix, unsigned int* LocationMatrix, size_t ND)。

许多有限元程序均使用了上面讨论的一维变带宽存储方案。使用一维变带宽存储方案后，求解线性代数方程组 $\boldsymbol{Ka}=\boldsymbol{R}$ 时大约需要 $\frac{1}{2}nm_k^2$ 次运算，其中 n 为矩阵 $\boldsymbol{K}$ 的阶数，m_k 为平均半带宽，即列高 $(i-m_i)(i=1,2,\cdots,n)$ 的平均值。因此，减小平均半带宽不但可以减少矩阵 $\boldsymbol{K}$ 的存储量，而且可以大幅减少求解方程组所需的运算次数。对某些简单问题可以凭经验确定结点的合理编号方式，以尽可能减小矩阵 $\boldsymbol{K}$ 的半带宽。但在复杂问题中，人工确定结点编号的最优方式是很困难的，此时可使用一些半带宽优化算法来自动对结点进行编号。例如，图2.16所示为用 6 个 8 结点四边形单元离散的悬臂梁，采用了两种不同的结点编号方式。第一种编号方式的刚度矩阵 $\boldsymbol{K}$ 的最大列高为 45，而第二种编号方式的最大列高只有 15，即第二种编号方式的平均半带宽大约只有第一种编号方式的 1/3，因此第二种编号方式求解方程的计算量大约只有第一种编号方式的 1/9 左右。

采用一维变带宽存储方案可以极大地降低有限元法的内存使用量和计算量。例如，完全存储（双精度）$n=10^6$ 阶的刚度阵需要 8TB（$8n^2$）的内存，但如果采用一维变带宽存储方案，假设平均带宽 $m_k=10^3$，则只需 8GB（$8nm_k$）内存，求解时间也只有原来的 2×10^5（$n^3/(\frac{1}{2}10nm_k^2)$）分之一左右（式(2.49)中的系数 C 取为原来的 10 倍）。因此，原来需要求解 30 年的方程组，采用一维变带宽存储方案后只需要 1.3 小时左右。

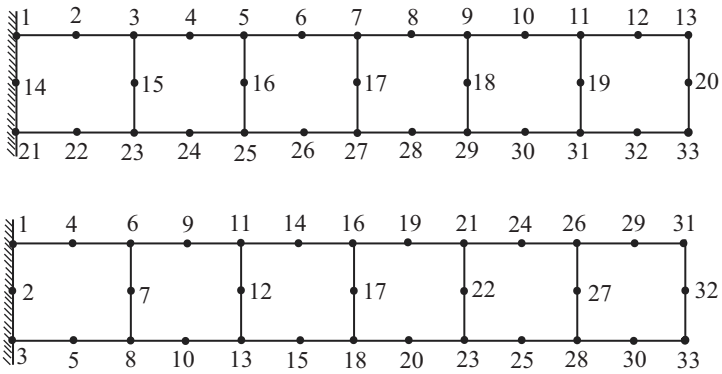

图 2.16: 两种典型的结点编号方式

2.5.3.2　行压缩格式

行压缩格式 (CSR) 有两种形式：4 数组形式和 3 数组形式。3 数组形式的行压缩格式采用 values、columns 和 rowIndex 3 个数组来存储稀疏矩阵，其中 values 按行依次存储刚度矩阵上三角部分的所有非零元素，columns 存储 values 的各元素在刚度矩阵中的列号，rowIndex 存储刚度阵各行的第一个非零元素 (即刚度阵的各对角元) 在 values 中的序号。数组 values 和 columns 的长度相同，而数组 rowIndex 的长度等于刚度矩阵的阶数 n 加 1，其中 rowIndex$(n+1)$ 等于刚度矩阵上三角部分非零元素总数加 1。rowIndex$(I+1)$ 和 rowIndex(I) 分别给出了刚度矩阵上三角部分的第 $I+1$ 行和第 I 行第一个非零元素在 values 中的地址，因此 rowIndex$(I+1)$ − rowIndex(I) 为第 I 行非零元素的个数。对于图2.15(a) 中的矩阵，行压缩格式的 3 个数组的内容分别为

	1	2	3	4	5	6	7	8	9	⋯	18	19
values	k_{11}	k_{12}	k_{14}	k_{22}	k_{23}	k_{33}	k_{34}	k_{36}	k_{44}	⋯	k_{78}	k_{88}
columns	1	2	4	2	3	3	4	6	4	⋯	8	8
rowIndex	1	4	6	9	12	15	17	19	20			

列压缩格式 (CSC) 与行压缩格式类似，它按列依次存放刚度阵的上三角部分。坐标存储格式 (COO) 将刚度阵上三角部分的所有非零元素存储在数组 rows 中，并用数组 rows 和 columns 分别存放各非零元素的行号和列号。

2.5.4 STAP++ 程序常用类说明

STAP++ 程序采用面向对象的设计思想，用 CDomain 类封装有限元模型数据，用 CNode 类封装结点数据，用抽象类 CElement 封装单元数据，用抽象类 CMaterial 类封装截面/材料属性数据，用 ElementGroup 封装单元组数据，用 CLoadCaseData 类封装载荷数据，用 CSkylineMatrix 封装按一维变带宽方式存储的总体刚度矩阵，用 CLDLTSolver 类封装 LDLT 求解器，并用 COutputter 类封装有限元模型和结果输出功能。下面对这些类分别进行介绍。

2.5.4.1 CDomain 类

CDomain 类封装有限元模型数据，其协作图如图2.17所示，其中紫色虚线箭头表示在类中申明了箭头所指向类的对象，对象名称标注在虚线箭头旁边。例如，CDomain 类中申明了 CNode 类的对象 NodeList、CElementGroup 类的对象 EleGrpList、CLoadCaseData 类的对象 LoadCases、CSkylineMatrix<double> 类的对象 StiffnessMatrix 和指向自身的对象 _instance，而 CElement 类中则申明了 CNode 类的对象 nodes_ 和 CMaterial 类的对象 ElementMaterial_。

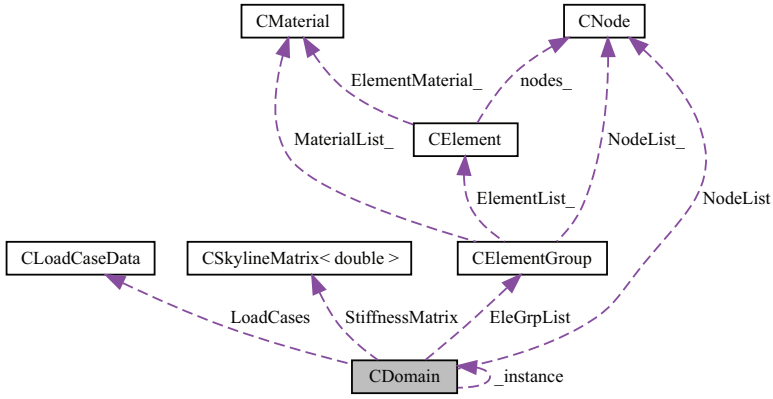

图 2.17: CDomain 类协作图

CDomain 类是一个单例类，即只能有一个 CDomain 类的实例。CDomain 类含有一个指向该类单例的静态私有指针，只提供私有构造函数，并提供了一个静态公有函数用于创建该单例或获取指向该单例的指针。CDomain 类具有以下私有成员变量：

1. static CDomain* _instance：指向 CDomain 类实例的静态私有指针；

2. ifstream Input：输入文件流，CDomain 将从该流中读取有限元模型数据；

3. char Title[256]：有限元模型的标题，用于简单描述该模型；

4. unsigned int MODEX：求解模式，0 表示只检查数据但不求解，1 表示求解；

5. unsigned int NUMNP：有限元模型的结点总数；

6. CNode* NodeList：结点列表数组；

7. unsigned int NUMEG：单元组总数，每个单元组中所有单元的类型必须相同；

8. CElementGroup* EleGrpList：单元组列表数组；

9. unsigned int NLCASE：载荷工况总数；

10. CLoadCaseData* LoadCases：载荷工况列表数组；

11. unsigned int* NLOAD：各载荷工况中的集中载荷总数；

12. unsigned int NEQ：总体自由度总数（总体刚度方程组的方程数）；

13. CSkylineMatrix<double>* StiffnessMatrix：总体刚度矩阵（以一维变带宽方式存放）；

14. double* Force：总体刚度方程右端项（求解刚度方程前为结点载荷向量，求解后为结点位移向量）。

CDomian 类提供的公有成员函数有：

1. static CDomain* GetInstance()：静态公有函数，用于创建 CDomain 类的单例或获取指向该单例的指针；

2. bool ReadData(string FileName, string OutFile)：从文件 FileName 中读取有限元模型数据，并将结果输出到文件 OutFile 中；

3. bool ReadNodalPoints()：读入结点数据；

4. bool ReadLoadCases()：读入载荷工况数据；

5. bool ReadElements()：读入单元数据；

6. void CalculateEquationNumber()：计算各结点各自由度所对应的总体自由度号（方程号）；

7. void CalculateColumnHeights()：计算总体刚度矩阵各列的列高；

8. void AllocateMatrices()：分配结点力数组 Force、总刚列高数组 ColumnHeights、对角元地址数组 DiagonalAddress 和总刚数组 StiffnessMatrix，并计算总刚各列列高和对角元地址；

9. void AssembleStiffnessMatrix()：组装总体刚度矩阵；

10. bool AssembleForce(unsigned int LoadCase)：组装工况 LoadCase 的结点力向量；

为了保护有限元模型数据不被其它代码随意修改，CDomain 类将其成员变量申明为私有的，并提供了以下公有操作来访问这些数据：

1. unsigned int GetMODEX()：返回求解模式 MODEX；
2. string GetTitle：返回有限元模型的标题 Title；
3. unsigned int GetNEQ：返回总体自由度总数 NEQ；
4. unsigned int GetNUMNP：返回结点总数 NUMNP；
5. CNode* GetNodeList()：返回结点列表数组 NodeList；
6. unsigned int GetNUMEG()：返回单元组总数 NUMEG；
7. CElementGroup* GetEleGrpList()：返回单元组列表数组 EleGrpList；
8. double* GetForce()：返回当前工况的结点载荷向量 Force；
9. double* GetDisplacement()：返回当前工况的位移向量；
10. unsigned int GetNLCASE()：返回载荷工况总数 NLCASE；
11. unsigned int* GetNLOAD()：返回各工况中的结点载荷总数 NLOAD；
12. CLoadCaseData* GetLoadCases()：返回载荷工况列表数组 LoadCases；
13. CSkylineMatrix<double>* GetStiffnessMatrix：返回总体刚度矩阵；

2.5.4.2 CNode 类

CNode 类封装结点数据，具有以下公有数据成员：

1. unsigned int NodeNumber：结点编号（从 1 开始编号）；
2. double XYZ [3]：结点的 x、y 和 z 坐标；
3. unsigned int bcode [NDF]：结点边界条件代码，其中静态常量 NDF 表示每个结点的自由度数，目前定义为 3。对于梁、板和壳问题，NDF 可能为 5 或 6。

CNode 类提供了以下公有成员函数：

1. bool Read (ifstream &Input)：从输入文件流 Input 中读入结点数据；
2. void Write (COutputter &output)：输出结点数据；
3. void WriteEquationNo (COutputter &OutputFile)：输出结点各自由度对应的总体自由度号；
4. void WriteNodalDisplacement (COutputter &OutputFile, double *Displacement)：输出当前工况的结点位移。

2.5.4.3 CElement 类

CElement 类是一个单元抽象类，只能作为基类来派生各种单元类，而不能用其直接声明对象。CElement 类的保护成员变量（可以被派生类的成员函数引

用）有：

1. unsigned int NEN_：每个单元的节点数（杆单元为 2，三角形常应变单元为 3）；

2. CNode** nodes_：单元的结点数组，其元素为指向单元各结点的指针；

3. CMaterial* ElementMaterial_：指向该单元截面/材料属性的指针；

4. unsigned int* LocationMatrix_：单元位置数组，存储单元各自由度对应的总体自由度号；

5. unsigned int ND_：单元自由度数（二维杆单元为 4，三维杆单元为 6）。

CElemenet 类的公有成员函数有：

1. bool Read(ifstream& Input, CMaterial* MaterialSets, CNode* NodeList)：纯虚函数，用于从输入流 Input 中读入单元数据，其具体实现在派生类中给出；

2. void Write(COutputter& output)：纯虚函数，用于输出单元数据，其具体实现在派生类中给出；

3. void GenerateLocationMatrix()：纯虚函数，用于生成单元位置数组 LocationMatrix_，其具体实现在派生类中给出；

4. unsigned int SizeOfStiffnessMatrix()：返回按列存放单元刚度矩阵上三角部分的一维数组的大小。

5. void ElementStiffness(double* stiffness)：纯虚函数，用于计算单元刚度矩阵（一维数组，按列存放上三角部分），其具体实现在派生类中给出；

6. void ElementStress(double* stress, double* Displacement)：纯虚函数，计算单元应力 stress，其具体实现在派生类中给出；

CElement 类提供了如下操纵以存取其保护成员变量：

1. unsigned int GetNEN()：返回单元的结点数；

2. CNode** GetNodes：返回指向单元结点数组的指针；

3. CMaterial* GetElementMaterial()：返回指向单元界面/材料属性的指针；

4. unsigned int* GetLocationMatrix()：返回指向单元位置数组的指针；

5. unsigned int GetND()：返回单元自由度数；

2.5.4.4 CBar 类

CBar 类是抽象类 CElement 的派生类，用于封装三维杆单元。CBar 类提供了以下公有成员函数：

1. bool Read(ifstream& Input, CMaterial* MaterialSets, CNode* NodeList)：从输入流 Input 中读入杆单元数据；

2. void Write(COutputter& output)：输出杆单元数据；

3. void GenerateLocationMatrix()：生成杆单元自由度位置数组；

4. void ElementStiffness(double* Matrix)：计算杆单元的单元刚度矩阵；

5. void ElementStress(double* stress, double* Displacement)：计算杆单元的应力；

2.5.4.5 CMaterial 类

CMaterial 类是一个截面/材料属性抽象类，用于作为基类派生各种截面/材料属性类。其公有成员变量有：

1. unsigned int nset：材料组序号；

2. double E：弹性模量。

CMaterial 类的公有成员函数有：

1. bool Read(ifstream& Input)：纯虚函数，用于从输入流 Input 中读入截面/材料属性数据，其具体实现在派生类中给出；

2. void Write(COutputter& output)：纯虚函数，用于输出截面/材料属性数据，其具体实现在派生类中给出。

2.5.4.6 CBarMaterial

CBarMaterial 类是 CMaterial 类的派生类，用于定于杆单元的截面/材料属性。CBarMaterial 类除了继承 CMaterial 类的所有成员外，还新定义了成员变量 double Area，用于定义杆单元的截面面积。

2.5.4.7 CLoadCaseData 类

CLoadCaseData 类封装载荷工况数据，其公有成员变量有：

1. unsigned int nloads：本工况中的集中载荷数；

2. unsigned int* node：受集中载荷作用的结点号数组；

3. unsigned int* dof：集中载荷的作用方向数组；

4. double* load：集中载荷的大小数组。

CLoadCaseData 类的公有成员函数有：

1. void Allocate(unsigned int num)：初始化 nloads，并创建数组 node、dof 和 load；

2. bool Read(ifstream& Input)：从数据流 Input 中读入载荷工况数据；

3. void Write(COutputter& output)：输出载荷工况数据。

2.5.4.8 CElementGroup

CElementGroup 类封装单元组，具有以下私有成员变量：

1. static CNode* NodeList_：指向结构结点列表的指针。该变量为静态变量，保证所有单元组对象均指向结构结点列表；

2. ElementTypes ElementType_：本组单元的单元类型。ElementTypes 为枚举类型，目前可以取值 Bar、Q4、T3、H8、Beam、Plate 和 Shell，但程序中只实现了 Bar 单元；

3. std::size_t ElementSize_：本单元组的单元类型（如 CBar）对象所占的内存字节数；

4. unsigned int NUME_：本单元组的单元总数；

5. CElement* ElementList_：本单元组的单元列表数组；

6. unsigned int NUMMAT_：本单元组的截面/材料属性组数；

7. CMaterial* MaterialList_：本单元组的截面/材料属性列表数组；

8. std::size_t MaterialSize_：本单元组的截面/材料属性类型（如 CBarMaterial）对象所占的内存字节数；

CElementGroup 类提供的公有成员函数有：

1. bool Read(ifstream& Input)：从输入流 Input 中读入单元组数据；

2. void CalculateMemberSize()：计算本单元组的单元类型对象和截面/材料属性类型对象所占的内存字节数，即 ElementSize_ 和 MaterialSize_（在新增加单元派生类时，必须扩充此函数以计算新单元派生类的相关信息）；

3. void AllocateElements(std::size_t size)：创建单元列表数组；

4. void AllocateMaterials(std::size_t size)：创建截面/材料属性列表数组；

5. ElementTypes GetElementType()：返回本单元组的单元类型；

6. unsigned int GetNUME()：返回本单元组的单元总数；

7. CElement& operator[](unsigned int i)：重载操作符“[ ]”，根据单元类型正确存取单元列表数组中的各单元对象；

8. CMaterial& GetMaterial(unsigned int i)：返回本单元组的第 i 组截面/材料属性对象；

9. unsigned int GetNUMMAT()：返回本单元组的截面/材料属性组数。

2.5.4.9 COutputter 类

COutputter 类是一个单例类，按照与 STAP90 相同的格式将有限元模型和计算结果同时输出到输出文件流 OutputFile 和标准输出流 cout 中。COutputter 类具有以下公有成员函数：

1. inline ofstream* GetOutputFile()：返回指向输出文件流 OutputFile 的指针；

2. static COutputter* GetInstance(string FileName)：创建/返回该类的单例，其输出文件流对应的文件名为 FileName。

3. void PrintTime(const struct tm * ptm, COutputter& output)：输出当前日期与时间；

4. void OutputHeading()：输出标题信息；

5. void OutputNodeInfo()：输出结点数据；

6. void OutputEquationNumber()：输出各结点自由度对应的总体自由度号（方程号）；

7. void OutputElementInfo()：输出单元数据；

8: void OutputBarElements(unsigned int EleGrp)：输出杆单元数据（单元组号为 EleGrp）；

9. void OutputLoadInfo()：输出载荷数据；

10. void OutputNodalDisplacement()：输出当前工况的结点位移；

11. void OutputElementStress()：输出当前工况的单元应力；

12. void OutputTotalSystemData()：输出模型总体信息。

2.5.4.10 CSkylineMatrix 模板类

CSkylineMatrix 模板类封装以一维变带宽方式存储的总体刚度矩阵，其私有成员变量有：

1. T_* data_：一维数组，用于存储总体刚度矩阵；

2. unsigned int NEQ_：总刚矩阵的维数；

3. unsigned int MK_：最大变带宽；

4. unsigned int NWK_：一维变带宽存储的总刚数组的大小；

5. unsigned int* ColumnHeights_：列高数组；

6. unsigned int* DiagonalAddress_：对角元数组。

CSkylineMatrix 类提供了以下成员函数：

1. T_& operator()(unsigned int i, unsigned int j)：重载运算符 (i,j)，返回刚度矩阵元素 k_{ij}（i 和 j 从 1 开始）；

2. void Allocate()：为数组 data_ 分配内存；

3. void CalculateColumnHeight(unsigned int* LocationMatrix, size_t ND)：计算列高数组，其中 LocationMatrix 为单元自由度位置数组，ND 为单元自由度数；

4. void CalculateMaximumHalfBandwidth()：计算最大半带宽（= max(ColumnHeights) + 1))；

5. void CalculateDiagnoalAddress()：计算各列的对角元地址（从 1 开始）；

6. void Assembly(double* Matrix, unsigned int* LocationMatrix, size_t ND)：将单元的刚度矩阵 Matrix 组装到总体刚度矩阵中；

7. unsigned int* GetColumnHeights()：返回列高数组 ColumnHeights_；

8. unsigned int GetMaximumHalfBandwidth() const：返回最大半带宽 MK_；

9. unsigned int* GetDiagonalAddress()：返回对角元地址数组 DiagonalAddress_；

10. unsigned int dim() const：返回总刚度矩阵的维数 NEQ_；

11. unsigned int size() const：返回用于存储一维变带宽存储的总刚数组的大小 NWK_。

2.5.4.11　CLDLTSolver 类

CLDLTSolver 类封装了线性代数方程组的 LDLT 求解器，具有私有成员变量 CSkylineMatrix<double>& K，提供以下两个公有成员函数：

1. void LDLT()：对以一维变带宽储存的总体刚度矩阵 K 进行 LDLT 分解；

2. void BackSubstitution(double* Force)：对右端项进行消去与回代求解。

习题

2-1　利用 STAP90++ 程序计算如 2.18 所示桁架各结点的位移和各杆件的内力，并校验结果的正确性。各杆件的截面积均为 $A = 10^{-2}\mathrm{m}^{-2}$，弹性模量均为 $E = 1.5 \times 10^{11}\mathrm{Pa}$。

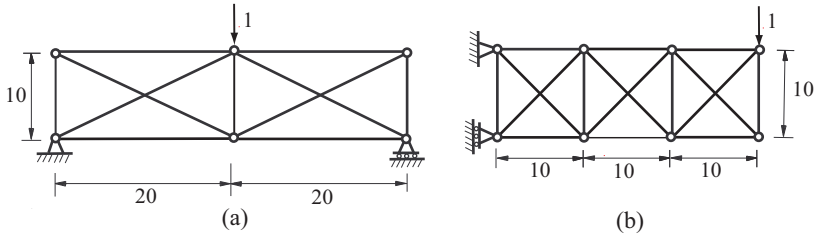

图 2.18: 桁架

2-2 利用 STAP90++ 程序计算如 2.12 所示桁架各结点的位移和各杆件的内力，并校验结果的正确性。各杆件的面积均为 $A = 0.1$，弹性模量均为 $E = 2.0 \times 10^5$。

2-3 扩展 STAP90++ 程序，逐步增加在本课程中学习的单元，如 3T（3 结点三角形单元）、4Q（4 结点四边形等参元）、8H（8 结点六面体等参元）和梁板壳等。