25

The Assembly Process

TABLE OF CONTENTS

§25.1. Introduction

Chapters 20, 23 and 24 dealt with element level operations. Sandwiched between element processing and solution there is the *assembly* process that constructs the master stiffness equations. Assembler examples for special models were given as recipes in the complete programs of Chapters 21 and 22. In the present chapter assembly will be studied with more generality.

The position of the assembler in a DSM-based code for static analysis is sketched in Figure 25.1. In most codes the assembler is implemented as a *element library loop driver*. This means that instead of forming all elements first and then assembling, the assembler constructs one element at a time in a loop, and immediately merges it into the master equations. This is the *merge loop*.

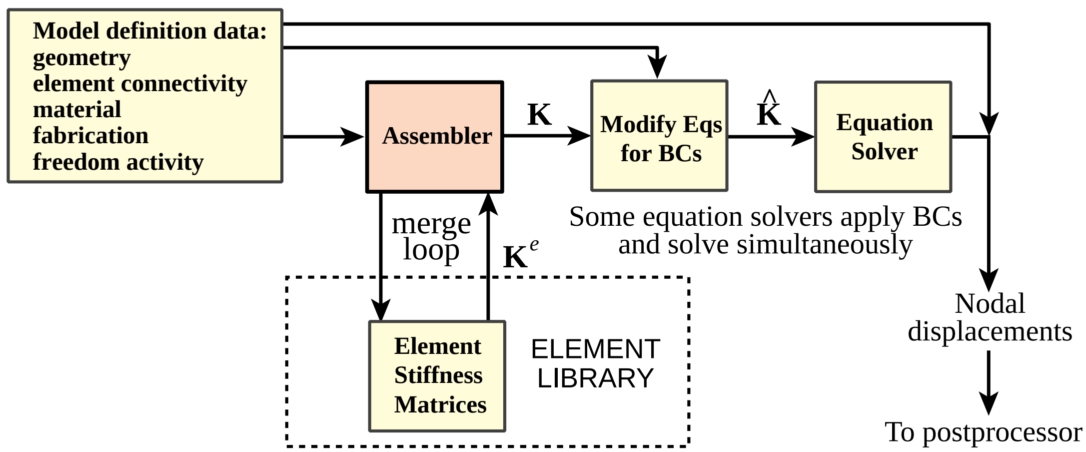

FIGURE 25.1. Role of assembler in FEM program.

Assembly is the most complicated stage of a production finite element program in terms of data flow. The complexity is not due to mathematical operations, which merely involve matrix addition, but interfacing with a large element library. Forming an element requires access to geometry, connectivity, material, and fabrication data. (In nonlinear analysis, to the state as well.) Merge requires access to freedom activity data, as well as knowledge of the sparse matrix format in which **K** is stored. As illustrated in Figure 25.1, the assembler sits at the crossroads of this data exchange.

The coverage of the assembly process for a general FEM implementation is beyond the scope of an introductory course. Instead this Chapter takes advantage of assumptions that lead to coding simplifications. This allows the basic aspects to be covered without excessive delay.

§25.2. Simplifications

The assembly process is considerably simplified if the FEM implementation has these properties:

- All elements are of the same type. For example: all elements are 2-node plane bars.
- The number and configuration of degrees of freedom at each node is the same.
- There are no "gaps" in the node numbering sequence.
- There are no multifreedom constraints treated by master-slave or Lagrange multiplier methods.
- The master stiffness matrix is stored as a full symmetric matrix.

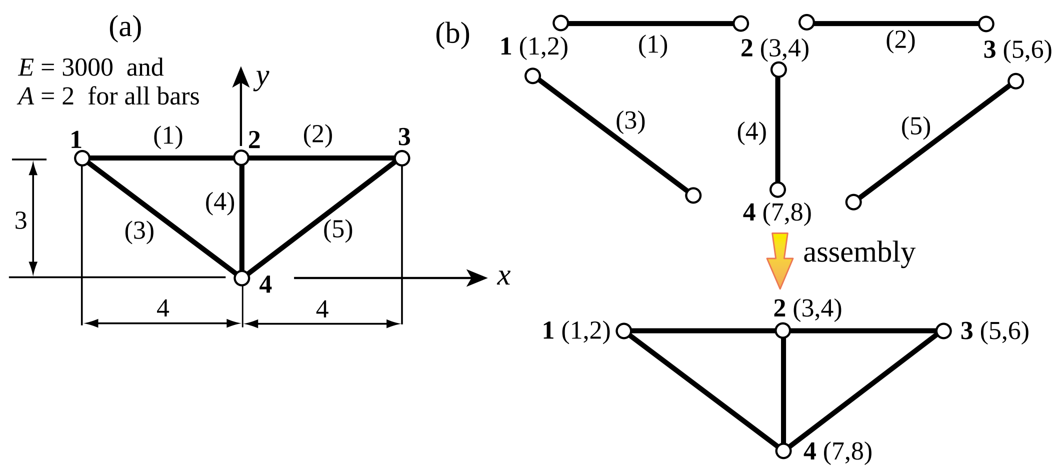

FIGURE 25.2. Left: Example plane truss structure. (a): model definition; (b) disconnection → assembly process. Numbers in parentheses written after node numbers are global DOF numbers.

If the first four conditions are met the implementation is simpler because the element freedom table described below can be constructed "on the fly" from the element node numbers. The last condition simplifies the indexing to access entries of **K**.

§25.3. Simplified Assemblers

In this section all simplifying assumptions listed in §25.2 are assumed to hold. This allows us to focus on *local-to-global DOF mapping*. The process is illustrated on a plane truss structure.

§25.3.1. A Plane Truss Example Structure

The plane truss shown in Figure 25.2(a) will be used to illustrate the details of the assembly process. The structure has 5 elements, 4 nodes and 8 degrees of freedom. The disconnection-to-assembly process is pictured in Figure 25.2(b).

Begin by clearing all entries of the 8×8 master stiffness matrix **K** to zero, so that we effectively start with the null matrix:

$$\mathbf{K} = \begin{array}{c} \left[\begin{array}{cccccccc} 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{array}\right] \end{array} \begin{array}{c} \mathit{1}\\ \mathit{2}\\ \mathit{3}\\ \mathit{4}\\ \mathit{5}\\ \mathit{6}\\ \mathit{7}\\ \mathit{8} \end{array} \tag{25.1}$$

The numbers written after each row of **K** are the *global DOF numbers*.

Element (1) joins nodes 1 and 2. The global DOFs of those nodes are $2 \times 1 - 1 = 1$, $2 \times 1 = 1$, $2 \times 2 - 1 = 3$ and $2 \times 2 = 4$. See Figure 25.2(b). Those four numbers are collected into an array called the *element freedom table*, or EFT for short. The element stiffness matrix $\mathbf{K}^{(1)}$ is listed on the left below with the EFT entries annotated after the matrix rows. On the right is **K** upon merging the element stiffness:

$$
\left[\begin{array}{cccc} 1500 & 0 & -1500 & 0 \\ 0 & 0 & 0 & 0 \\ -1500 & 0 & 1500 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]
\begin{array}{l} \mathit{1} \\ \mathit{2} \\ \mathit{3} \\ \mathit{4} \end{array}
\qquad
\left[\begin{array}{cccccccc} 1500 & 0 & -1500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1500 & 0 & 1500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]
\begin{array}{l} \mathit{1} \\ \mathit{2} \\ \mathit{3} \\ \mathit{4} \\ \\ \\ \\ \\ \end{array}
\tag{25.2}
$$

Element (2) joins nodes 2 and 3. The global DOFs of those nodes are $2 \times 2 - 1 = 3$, $2 \times 2 = 4$, $2 \times 3 - 1 = 5$ and $2 \times 3 = 6$; thus the EFT is { `3,4,5,6` }. Matrices $\mathbf{K}^{(2)}$ and **K** upon merge are

$$
\left[\begin{array}{cccc} 1500 & 0 & -1500 & 0 \\ 0 & 0 & 0 & 0 \\ -1500 & 0 & 1500 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]
\begin{array}{l} \mathit{3} \\ \mathit{4} \\ \mathit{5} \\ \mathit{6} \end{array}
\qquad
\left[\begin{array}{cccccccc} 1500 & 0 & -1500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1500 & 0 & 3000 & 0 & -1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1500 & 0 & 1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]
\begin{array}{l} \\ \\ \mathit{3} \\ \mathit{4} \\ \mathit{5} \\ \mathit{6} \\ \\ \\ \end{array}
\tag{25.3}
$$

Element (3) joins nodes 1 and 4. Its EFT is { `1,2,7,8` }. Matrices $\mathbf{K}^{(3)}$ and **K** upon merge are

$$
\left[\begin{array}{cccc} 768 & -576 & -768 & 576 \\ -576 & 432 & 576 & -432 \\ -768 & 576 & 768 & -576 \\ 576 & -432 & -576 & 432 \end{array}\right]
\begin{array}{l} \mathit{1} \\ \mathit{2} \\ \mathit{7} \\ \mathit{8} \end{array}
\qquad
\left[\begin{array}{cccccccc} 2268 & -576 & -1500 & 0 & 0 & 0 & -768 & 576 \\ -576 & 432 & 0 & 0 & 0 & 0 & 576 & -432 \\ -1500 & 0 & 3000 & 0 & -1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1500 & 0 & 1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -768 & 576 & 0 & 0 & 0 & 0 & 768 & -576 \\ 576 & -432 & 0 & 0 & 0 & 0 & -576 & 432 \end{array}\right]
\begin{array}{l} \mathit{1} \\ \mathit{2} \\ \\ \\ \\ \\ \mathit{7} \\ \mathit{8} \end{array}
\tag{25.4}
$$

Element (4) joins nodes 2 and 4. Its EFT is { 3,4,7,8 }. Matrices $\mathbf{K}^{(4)}$ and **K** upon merge are

$$
\left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 2000 & 0 & -2000 \\ 0 & 0 & 0 & 0 \\ 0 & -2000 & 0 & 2000 \end{array}\right]
\begin{array}{l} \mathit{3} \\ \mathit{4} \\ \mathit{7} \\ \mathit{8} \end{array}
\qquad
\left[\begin{array}{cccccccc} 2268 & -576 & -1500 & 0 & 0 & 0 & -768 & 576 \\ -576 & 432 & 0 & 0 & 0 & 0 & 576 & -432 \\ -1500 & 0 & 3000 & 0 & -1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2000 & 0 & 0 & 0 & -2000 \\ 0 & 0 & -1500 & 0 & 1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -768 & 576 & 0 & 0 & 0 & 0 & 768 & -576 \\ 576 & -432 & 0 & -2000 & 0 & 0 & -576 & 2432 \end{array}\right]
\begin{array}{l} \\ \\ \mathit{3} \\ \mathit{4} \\ \\ \\ \mathit{7} \\ \mathit{8} \end{array}
\tag{25.5}
$$

Finally, element (5) joins nodes 3 and 4. Its EFT is { 5,6,7,8 }. Matrices $\mathbf{K}^{(5)}$ and $\mathbf{K}$ upon merge are

$$
\begin{array}{cc}
\begin{bmatrix} 768 & 576 & -768 & -576 \\ 576 & 432 & -576 & -432 \\ -768 & -576 & 768 & 576 \\ -576 & -432 & 576 & 432 \end{bmatrix} & \begin{array}{c} 5 \\ 6 \\ 7 \\ 8 \end{array}
\end{array}
\quad
\begin{array}{cc}
\begin{bmatrix} 2268 & -576 & -1500 & 0 & 0 & 0 & -768 & 576 \\ -576 & 432 & 0 & 0 & 0 & 0 & 576 & -432 \\ -1500 & 0 & 3000 & 0 & -1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2000 & 0 & 0 & 0 & -2000 \\ 0 & 0 & -1500 & 0 & 2268 & 576 & -768 & -576 \\ 0 & 0 & 0 & 0 & 576 & 432 & -576 & -432 \\ -768 & 576 & 0 & 0 & -768 & -576 & 1536 & 0 \\ 576 & -432 & 0 & -2000 & -576 & -432 & 0 & 2864 \end{bmatrix} & \begin{array}{c} \\ \\ \\ \\ 5 \\ 6 \\ 7 \\ 8 \end{array}
\end{array}
\tag{25.6}
$$

Since all elements have been processed, (25.6) is the master stiffness before application of boundary conditions.

```
PlaneTrussMasterStiffness[nodxyz_,elenod_,elemat_,elefab_,
  eleopt_]:=Module[{numele=Length[elenod],numnod=Length[nodxyz],
  e,ni,nj,eft,i,j,ii,jj,ncoor,Em,A,options,Ke,K},
  K=Table[0,{2*numnod},{2*numnod}];
  For [e=1, e<=numele, e++, {ni,nj}=elenod[[e]];
       eft={2*ni-1,2*ni,2*nj-1,2*nj};
       ncoor={nodxyz[[ni]],nodxyz[[nj]]};
       Em=elemat[[e]]; A=elefab[[e]]; options=eleopt;
       Ke=PlaneBar2Stiffness[ncoor,Em,A,options];
       For [i=1, i<=4, i++, ii=eft[[i]];
           For [j=i, j<=4, j++, jj=eft[[j]];
               K[[jj,ii]]=K[[ii,jj]]+=Ke[[i,j]] ];
           ];
      ];  Return[K]
  ];
```

FIGURE 25.3. Plane truss assembler module.

```
nodxyz={{-4,3},{0,3},{4,3},{0,0}};
elenod= {{1,2},{2,3},{1,4},{2,4},{3,4}};
elemat= Table[3000,{5}]; elefab= Table[2,{5}]; eleopt= {True};
K=PlaneTrussMasterStiffness[nodxyz,elenod,elemat,elefab,eleopt];
Print["Master Stiffness of Plane Truss of Fig 25.2:"];
K=Chop[K]; Print[K//MatrixForm];
Print["Eigs of K=",Chop[Eigenvalues[N[K]]]];
```

Master Stiffness of Plane Truss of Fig 25.2:

$$
\begin{pmatrix} 2268. & -576. & -1500. & 0 & 0 & 0 & -768. & 576. \\ -576. & 432. & 0 & 0 & 0 & 0 & 576. & -432. \\ -1500. & 0 & 3000. & 0 & -1500. & 0 & 0 & 0 \\ 0 & 0 & 0 & 2000. & 0 & 0 & 0 & -2000. \\ 0 & 0 & -1500. & 0 & 2268. & 576. & -768. & -576. \\ 0 & 0 & 0 & 0 & 576. & 432. & -576. & -432. \\ -768. & 576. & 0 & 0 & -768. & -576. & 1536. & 0 \\ 576. & -432. & 0 & -2000. & -576. & -432. & 0 & 2864. \end{pmatrix}
$$

Eigs of K={5007.22, 4743.46, 2356.84, 2228.78, 463.703, 0, 0, 0}

FIGURE 25.4. Plane truss assembler tester and output results.

§25.3.2. Simplified Assembler Implementation

Figure 25.3 shows a a *Mathematica* implementation of the assembly process just described. This assembler calls the element stiffness module `PlaneBar2Stiffness` of Figure 20.2. The assembler is invoked by

```
K = SpaceTrussMasterStiffness[nodxyz,elenod,elemat,elefab,prcopt]
```
(25.7)

The five arguments: `nodxyz`, `elenod`, `elemat`, `elefab`, and `prcopt` have the same function as those described in §21.1.3 for the `SpaceTrussMasterStiffness` assembler. In fact, comparing the code in Figure 25.3 to that of Figure 21.1, the similarities are obvious.

Running the script listed in the top of Figure 25.4 produces the output shown in the bottom of that figure. The master stiffness (25.6) is reproduced. The eigenvalue analysis verifies that the assembled stiffness has the correct rank of $8 - 3 = 5$.

§25.4. MET Assemblers

The next complication occurs when elements of different types are to be assembled, while the number and configuration of degrees of freedom at each node remains the same. This scenario is common in special-purpose programs for analysis of specific problems: plane stress, plate bending, solids. This will be called a *multiple-element-type assembler*, or MET assembler for short.

The only incremental change with respect to the simplest assemblers is that an array of element types, denoted here by `eletyp`, has to be provided.

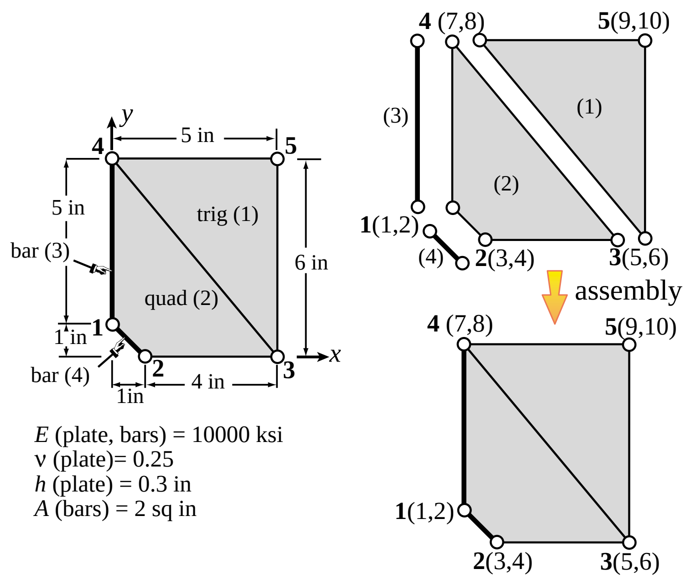

FIGURE 25.5. A plane stress structure used to test a MET assembler.

The assembler calls the appropriate element stiffness module according to type, and builds the local to global DOF mapping accordingly.

§25.4.1. A Plane Stress Assembler

This kind of assembler will be illustrated using the module `PlaneStressMasterStiffness`. As its name indicates, this is suitable for use in plane stress analysis programs. We make the following assumptions:

1 All nodes in the FEM mesh have 2 DOFs, namely the $\{u_x, u_y\}$ displacements. There are no node gaps or MFCs. The master stiffness is stored as a full matrix.

2 Element types can be: 2-node bars, 3-node linear triangles, ot 4-node bilinear quadrilaterals. These elements can be freely intermixed. Element types are identified with character strings `"Bar2"`, `"Trig3"` and `"Quad4"`, respectively.

As noted above, the chief modification over the simplest assembler is that a different stiffness module is invoked as per type. Returned stiffness matrices are generally of different order; in our case 4×4, 6×6 and 8×8 for the bar, triangle and quadrilateral, respectively. But the construction of the EFT is immediate, since node n maps to global freedoms $2n - 1$ and $2n$. The technique is illustrated with the 4-element, 5-node plane stress structure shown in Figure 25.5. It consists of one triangle, one quadrilateral, and two bars. The assembly process goes as follows.

Element (1) is a 3-node triangle with nodes 3, 5 and 4, whence the EFT is { `5,6,9,10,7,8` }. The element stiffness is

$$\mathbf{K}^{(1)} = \left[\begin{array}{rrrrrr} 500.0 & 0 & -500.0 & -600.0 & 0 & 600.0 \\ 0 & 1333.3 & -400.0 & -1333.3 & 400.0 & 0 \\ -500.0 & -400.0 & 2420.0 & 1000.0 & -1920.0 & -600.0 \\ -600.0 & -1333.3 & 1000.0 & 2053.3 & -400.0 & -720.0 \\ 0 & 400.0 & -1920.0 & -400.0 & 1920.0 & 0 \\ 600.0 & 0 & -600.0 & -720.0 & 0 & 720.0 \end{array}\right] \begin{array}{l} \mathit{5} \\ \mathit{6} \\ \mathit{9} \\ \mathit{10} \\ \mathit{7} \\ \mathit{8} \end{array} \tag{25.8}$$

Element (2) is a 4-node quad with nodes 2, 3, 4 and 1, whence the EFT is { `3,4,5,6,7,8,1,2` }. The element stiffness computed with a 2×2 Gauss rule is

$$\mathbf{K}^{(2)} = \left[\begin{array}{rrrrrrrr} 2076.4 & 395.55 & -735.28 & -679.11 & -331.78 & -136.71 & -1009.3 & 420.27 \\ 395.55 & 2703.1 & -479.11 & -660.61 & -136.71 & -1191.5 & 220.27 & -850.95 \\ -735.28 & -479.11 & 2067.1 & 215.82 & 386.36 & 427.34 & -1718.1 & -164.05 \\ -679.11 & -660.61 & 215.82 & 852.12 & 627.34 & 358.30 & -164.05 & -549.81 \\ -331.78 & -136.71 & 386.36 & 627.34 & 610.92 & 178.13 & -665.50 & -668.76 \\ -136.71 & -1191.5 & 427.34 & 358.30 & 178.13 & 1433.4 & -468.76 & -600.15 \\ -1009.3 & 220.27 & -1718.1 & -164.05 & -665.50 & -468.76 & 3393.0 & 412.54 \\ 420.27 & -850.95 & -164.05 & -549.81 & -668.76 & -600.15 & 412.54 & 2000.9 \end{array}\right] \begin{array}{l} \mathit{3} \\ \mathit{4} \\ \mathit{5} \\ \mathit{6} \\ \mathit{7} \\ \mathit{8} \\ \mathit{1} \\ \mathit{2} \end{array} \tag{25.9}$$

Element (3) is a 2-node bar with nodes 1 and 4, whence the EFT is { `1,2,7,8` }. The element stiffness is

$$\mathbf{K}^{(3)} = \left[\begin{array}{rrrr} 0 & 0 & 0 & 0 \\ 0 & 4000.0 & 0 & -4000.0 \\ 0 & 0 & 0 & 0 \\ 0 & -4000.0 & 0 & 4000.0 \end{array}\right] \begin{array}{l} \mathit{1} \\ \mathit{2} \\ \mathit{7} \\ \mathit{8} \end{array} \tag{25.10}$$

Element (4) is a 2-node bar with nodes 1 and 2, whence the EFT is { `1,2,3,4` }. The element stiffness is

$$\mathbf{K}^{(4)} = \left[\begin{array}{rrrr} 7071.1 & -7071.1 & -7071.1 & 7071.1 \\ -7071.1 & 7071.1 & 7071.1 & -7071.1 \\ -7071.1 & 7071.1 & 7071.1 & -7071.1 \\ 7071.1 & -7071.1 & -7071.1 & 7071.1 \end{array}\right] \begin{array}{l} \mathit{1} \\ \mathit{2} \\ \mathit{3} \\ \mathit{4} \end{array} \tag{25.11}$$

Upon assembly, the master stiffness of the plane stress structure, printed with one digit after the

```
PlaneStressMasterStiffness[nodxyz_,eletyp_,elenod_,
  elemat_,elefab_,prcopt_]:=Module[{numele=Length[elenod],
  numnod=Length[nodxyz],ncoor,type,e,enl,neldof,
       i,n,ii,jj,eftab,Emat,th,numer,Ke,K},
  K=Table[0,{2*numnod},{2*numnod}];  numer=prcopt[[1]];
  For [e=1,e<=numele,e++, type=eletyp[[e]];
       If [!MemberQ[{"Bar2","Trig3","Quad4"},type], Print["Illegal type",
           " of element e=",e," Assembly interrupted"]; Return[K]];
       enl=elenod[[e]]; n=Length[enl];
       eftab=Flatten[Table[{2*enl[[i]]-1,2*enl[[i]]},{i,1,n}]];
       ncoor=Table[nodxyz[[enl[[i]]]],{i,n}];
       If [type=="Bar2",  Em=elemat[[e]]; A=elefab[[e]];
           Ke=PlaneBar2Stiffness[ncoor,Em,A,{numer}] ];
       If [type=="Trig3", Emat=elemat[[e]]; th=elefab[[e]];
            Ke=Trig3IsoPMembraneStiffness[ncoor,Emat,th,{numer}] ];
       If [type=="Quad4", Emat=elemat[[e]]; th=elefab[[e]];
            Ke=Quad4IsoPMembraneStiffness[ncoor,Emat,th,{numer,2}] ];
       neldof=Length[Ke];
       For [i=1,i<=neldof,i++, ii=eftab[[i]];
           For [j=i,j<=neldof,j++, jj=eftab[[j]];
                K[[jj,ii]]=K[[ii,jj]]+=Ke[[i,j]] ];
       ];
     ];  Return[K];
];
```

FIGURE 25.6. Implementation of MET assembler for plane stress.

decimal point, is

$$\begin{bmatrix} 10464.0 & -6658.5 & -8080.4 & 7291.3 & -1718.1 & -164.1 & -665.5 & -468.8 & 0 & 0 \\ -6658.5 & 13072.0 & 7491.3 & -7922.0 & -164.1 & -549.8 & -668.8 & -4600.2 & 0 & 0 \\ -8080.4 & 7491.3 & 9147.5 & -6675.5 & -735.3 & -679.1 & -331.8 & -136.7 & 0 & 0 \\ 7291.3 & -7922.0 & -6675.5 & 9774.1 & -479.1 & -660.6 & -136.7 & -1191.5 & 0 & 0 \\ -1718.1 & -164.1 & -735.3 & -479.1 & 2567.1 & 215.8 & 386.4 & 1027.3 & -500.0 & -600.0 \\ -164.1 & -549.8 & -679.1 & -660.6 & 215.8 & 2185.5 & 1027.3 & 358.3 & -400.0 & -1333.3 \\ -665.5 & -668.8 & -331.8 & -136.7 & 386.4 & 1027.3 & 2530.9 & 178.1 & -1920.0 & -400.0 \\ -468.8 & -4600.2 & -136.7 & -1191.5 & 1027.3 & 358.3 & 178.1 & 6153.4 & -600.0 & -720.0 \\ 0 & 0 & 0 & 0 & -500.0 & -400.0 & -1920.0 & -600.0 & 2420.0 & 1000.0 \\ 0 & 0 & 0 & 0 & -600.0 & -1333.3 & -400.0 & -720.0 & 1000.0 & 2053.3 \end{bmatrix} \tag{25.12}$$

The eigenvalues of $\mathbf{K}$ are

$$[\,32883. \quad 10517.7 \quad 5439.33 \quad 3914.7 \quad 3228.99 \quad 2253.06 \quad 2131.03 \quad 0 \quad 0 \quad 0\,] \tag{25.13}$$

which displays the correct rank.

§25.4.2. MET Assembler Implementation

An implementation of the MET plane stress assembler as a *Mathematica* module called `PlaneStressMasterStiffness` is shown in Figure 25.6. The assembler is invoked by

```
K = PlaneStressMasterStiffness[nodxyz,eletyp,elenod,elemat,elefab,prcopt]
```
(25.14)

The only additional argument is `eletyp`, which is a list of element types.

```
Em=10000; ν=1/4; h=3/10; A=2;
Emat=Em/(1-ν^2)*{{1,ν,0},{ν,1,0},{0,0,(1-ν)/2}};
nodxyz= {{0,1},{1,0},{5,0},{0,6},{5,6}};
elenod= {{3,5,4},{2,3,4,1},{1,4},{1,2}};
eletyp= {"Trig3","Quad4","Bar2","Bar2"};
elemat= {Emat,Emat,Em,Em};
elefab= {h,h,A,A}; prcopt={True};
K=PlaneStressMasterStiffness[nodxyz,eletyp,elenod,elemat,elefab,prcopt];
Print["Master Stiffness Matrix of Plane Stress Structure:"];
Print[SetPrecision[Chop[K],5]//MatrixForm];
Print["Eigs of K:",Chop[Eigenvalues[N[K]]]];
```

Master Stiffness Matrix of Plane Stress Structure:

$$
\begin{pmatrix}
10464. & -6658.5 & -8080.4 & 7291.3 & -1718.1 & -164.05 & -665.50 & -468.76 & 0 & 0 \\
-6658.5 & 13072. & 7491.3 & -7922.0 & -164.05 & -549.81 & -668.76 & -4600.2 & 0 & 0 \\
-8080.4 & 7491.3 & 9147.5 & -6675.5 & -735.28 & -679.11 & -331.78 & -136.71 & 0 & 0 \\
7291.3 & -7922.0 & -6675.5 & 9774.1 & -479.11 & -660.61 & -136.71 & -1191.5 & 0 & 0 \\
-1718.1 & -164.05 & -735.28 & -479.11 & 2567.1 & 215.82 & 386.36 & 1027.3 & -500.00 & -600.00 \\
-164.05 & -549.81 & -679.11 & -660.61 & 215.82 & 2185.5 & 1027.3 & 358.30 & -400.00 & -1333.3 \\
-665.50 & -668.76 & -331.78 & -136.71 & 386.36 & 1027.3 & 2530.9 & 178.13 & -1920.0 & -400.00 \\
-468.76 & -4600.2 & -136.71 & -1191.5 & 1027.3 & 358.30 & 178.13 & 6153.4 & -600.00 & -720.00 \\
0 & 0 & 0 & 0 & -500.00 & -400.00 & -1920.0 & -600.00 & 2420.0 & 1000.0 \\
0 & 0 & 0 & 0 & -600.00 & -1333.3 & -400.00 & -720.00 & 1000.0 & 2053.3
\end{pmatrix}
$$

Eigs of K: {32883., 10517.7, 5439.33, 3914.7, 3228.99, 2253.06, 2131.03, 0, 0, 0}

FIGURE 25.7. Script for assembling the plane stress model of Figure 25.5 and output results.

The script listed in the top of Figure 25.7 runs module `PlaneStressMasterStiffness` with the inputs appropriate to the problem defined in Figure 25.5. The output shown in the bottom of the figure reproduces the master stiffness matrix (25.12) and the eigenvalues (25.13). .

§25.5. MET-VFC Assemblers

The next complication beyond multiple element types is a major one: allowing nodes to have different freedom configurations. An assembler that handles this feature will be called a *MET-VFC assembler*, where VFC is an acronym for Variable Freedom Configuration.

A MET-VFC assembler gets closer to what is actually implemented in production FEM programs. A assembler of this kind can handle orientation nodes as well as node "gaps" automatically. Only two advanced attributes remain: allowing treatment of MFCs by Lagrange multipliers, and handling a sparse matrix storage scheme. The former can be done by an extension of the concept of element. The latter requires major programming modifications, however, and is not considered in this Chapter.

The implementation of a MET-VFC assembler requires the definition of additional data structures pertaining to freedoms, at both node and element levels. These are explained in the following subsections before giving a concrete application example.

§25.5.1. Node Freedom Arrangement

A key decision made by the implementor of a FEM program for structural analysis is: what freedoms will be allocated to nodes, and how are they arranged? While many answers are possible, we focus

```
PackNodeFreedomSignature[s_]:=FromDigits[s,10];
UnpackNodeFreedomSignature[p_,m_]:=IntegerDigits[p,10,m];
NodeFreedomCount[p_]:=DigitCount[p,10,1];
```

FIGURE 25.8. Node freedom manipulation utility functions.

here on the most common arrangement for a linear three-dimensional FEM program.[1] At each node n three displacements $\{u_{xn}, u_{yn}, u_{zn}\}$ and three infinitesimal rotations $\{\theta_{xn}, \theta_{yn}, \theta_{zn}\}$ are chosen as freedoms and ordered as follows:

$$u_{xn}, u_{yn}, u_{zn}, \theta_{xn}, \theta_{yn}, \theta_{zn} \tag{25.15}$$

This is a *Node Freedom Arrangement* or NFA. Once the decision is made on a NFA, the *position of a freedom never changes*.[2] Thus if (25.15) is adopted, position #2 is forever associated to u_{yn}.

The length of the Node Freedom Arrangement is called the NFA length and often denoted (as a variable) by `nfalen`. For the common choice (25.15) of 3D structural codes, `nfalen` is 6.

§25.5.2. Node Freedom Signature

Having picked a NFA such as (25.15) does not mean that all such freedoms need to be present at a node. Take for example a model, such as a space truss, that only has translational degrees of freedom. Then only $\{u_{xn}, u_{yn}, u_{zn}\}$ will be used, and it becomes unnecessary to carry rotations in the master equations. There are also cases where allocations may vary from node to node.

To handle such scenarios in general terms a *Node Freedom Signature* or NFS, is introduced. The NFS is a sequence of zero-one integers. If the $\mathtt{j}^{th}$ entry is one, it means that the $\mathtt{j}^{th}$ freedom in the NFA is allocated; whereas if zero that freedom is not in use. For example, suppose that the underlying NFA is (25.15) and that node 5 uses the three freedoms $\{u_{x5}, u_{y5}, \theta_{z5}\}$. Then its NFS is

$$\{\,1,1,0,0,0,1\,\} \tag{25.16}$$

If a node has no allocated freedoms (e.g, an orientation node), or has never been defined, its NFS contains only zeros.

It is convenient to decimally pack signatures into integers to save storage because allocating a full integer to hold just 0 or 1 is wasteful. The packed NFS (25.16) would be 110001. This technique requires utilities for packing and unpacking. *Mathematica* functions that provide those services are listed in Figure 25.8. All of them use built-in functions. A brief description follows.

`p = PackNodeFreedomSignature[s]`	Packs the NFS list `s` into integer `p`.
`s = UnpackNodeFreedomSignature[p,m]`	Unpacks the NFS `p` into a list `s` of length `m`, where `m` is the NFA length. (The second argument is necessary in case the NFS has leading zeros.)
`k = NodeFreedomCount[p]`	Returns count of ones in `p`.

[1] This is the scheme used by most commercial codes. In a geometrically nonlinear FEM code, however, finite rotation measures must be used. Consequently the infinitesimal node rotations $\{\theta_x, \theta_y, \theta_z\}$ must be replaced by something else.

[2] NFA positions #7 and beyond are available for additional freedom types, such as Lagrange multipliers, temperatures, pressures, fluxes, etc. It is also possible to reserve more NFA positions for structural freedoms.

```
NodeFreedomMapTable[nodfat_]:=Module[{numnod=Length[nodfat],
  i,nodfmt}, nodfmt=Table[0,{numnod}];
  For [i=1,i<=numnod-1,i++, nodfmt[[i+1]]=nodfmt[[i]]+
        DigitCount[nodfat[[i]],10,1]  ];
  Return[nodfmt]];

TotalFreedomCount[nodfat_]:=Sum[DigitCount[nodfat[[i]],10,1],
                               {i,Length[nodfat]}];
```

FIGURE 25.9. Modules to construct the NFMT from the NFAT, and to compute the total number of freedoms.

Example 25.1. `PackNodeFreedomSignature[{ 1,1,0,0,0,1 }]` returns `110001`. The inverse is `UnpackNodeFreedomSignature[110001,6]`, which returns `{ 1,1,0,0,0,1 }`.
`NodeFreedomCount[110001]` returns 3.

§25.5.3. The Node Freedom Allocation Table

The Node Freedom Allocation Table, or NFAT, is a node by node list of packed node freedom signatures. In *Mathematica* the list containing this table is internally identified as `nodfat`. The configuration of the NFAT for the plane stress example structure of Figure 25.5 is

$$\texttt{nodfat = \{ 110000,110000,110000,110000,110000 \} = Table[110000,\{ 5 \}].} \tag{25.17}$$

This says that only two freedoms: $\{u_x, u_y\}$, are used at each of the five nodes.

A zero entry in the NFAT flags a node with no allocated freedoms. This is either an orientation node, or an undefined one. The latter case happens when there is a node numbering gap, which is a common occurence when certain mesh generators are used. In the case of a MET-VFC assembler such gaps are not considered errors.

§25.5.4. The Node Freedom Map Table

The Node Freedom Map Table, or NFMT, is an array with one entry per node. Suppose that node n has $k \geq 0$ allocated freedoms. These have global equation numbers $i + j$, $j = 1, \ldots, k$. This i is called the *base equation index*: it represent the global equation number *before* the first equation to which node n contributes. Obviously $i = 0$ if $n = 1$. Base equation indices for all nodes are recorded in the NFMT.[3]

In *Mathematica* code the table is internally identified as `nodfmt`. Figure 25.9 lists a module that constructs the NFMT given the NFAT. It is invoked as

$$\texttt{nodfmt = NodeFreedomMapTable[nodfat]} \tag{25.18}$$

The only argument is the NFAT. The module builds the NFMT by simple incrementation, and returns it as function value.

[3] The "map" qualifier comes from the use of this array in computing element-to-global equation mapping.

```
ElementFreedomTable[enl_,efs_,nodfat_,nodfmt_,m_]:=Module[
  {nelnod=Length[enl],eft,i,j,k=0,ix,n,s,es,sx},
  eft=Table[0,{Sum[DigitCount[efs[[i]],10,1],{i,1,nelnod}]}];
  For [i=1,i<=nelnod,i++, n=enl[[i]];
       s=IntegerDigits[nodfat[[n]],10,m]; ix=0;
       sx=Table[0,{m}]; Do [If[s[[j]]>0,sx[[j]]=++ix],{j,1,m}];
       es=IntegerDigits[efs[[i]],10,m];
       For [j=1,j<=m,j++, If [es[[j]]>0, k++;
            If [s[[j]]>0, eft[[k]]=nodfmt[[n]]+sx[[j]] ]]];
       ];
  Return[eft]];
```

FIGURE 25.10. Module to construct the Element Freedom Table for a MET-VFC assembler.

Example 25.2. The NFAT for the plane truss-frame structure of Figure 25.11 is `nodfat = { 110001,110000, 110001,000000,110001 }`. The node freedom counts are `{ 3,2,3,0,3 }`. The call

$$\texttt{nodfmt = NodeFreedomMapTable[nodfat]} \tag{25.19}$$

returns `{ 0,3,5,8,8 }` in `nodfmt`. This can be easily verified visually: `0+3=3`, `0+3+2=5`, etc.

Figure 25.9 also lists a function `TotalFreedomCount` that returns the total number of freedoms in the master equations, given the NFAT as argument.

§25.5.5. The Element Freedom Signature

The Element Freedom Signature or EFS is a data structure that shares similarities with the NFAT, but provides freedom data at the element rather than global level. More specifically, given an element, it tells to which degrees of freedom it constributes.

The EFS is best described through an example. Suppose that in a general FEM program all defined structural nodes have the signature 111111, meaning the arrangement (25.15). The program includes 2-node space bar elements, which contribute only to translational degrees of freedom $\{u_x, u_y, u_z\}$. The EFS of any such element will be

$$\texttt{efs = \{ 111000,111000 \}} \tag{25.20}$$

For a 2-node space beam element, which contributes to all six nodal freedoms, the EFS will be

$$\texttt{efs = \{ 111111,111111 \}} \tag{25.21}$$

For a 3-node space beam element in which the third node is an orientation node:

$$\texttt{efs = \{ 111111,111111,000000 \}} \tag{25.22}$$

This information, along with the NFAT and NFMT, is used to build Element Freedom Tables.

§25.5.6. The Element Freedom Table

The Element Freedom Table or EFT has been encountered in all previous assembler examples. It is a one dimensional list of length equal to the number of element DOF. If the $\mathtt{i}^{th}$ element DOF maps to the $\mathtt{k}^{th}$ global DOF, then the $\mathtt{i}^{th}$ entry of EFT contains `k`. This allows to write the merge loop compactly. In the simpler assemblers discussed in previous sections, the EFT can be built "on the fly" simply from element node numbers. For a MET-VFC assembler that is no longer the case. To take care of the VFC feature, the construction of the EFT is best done by a separate module. A *Mathematica* implementation is shown in Figure 25.10. Discussion of the logic, which is not trivial, is relegated to an Exercise, and we simply describe here the interface. The module is invoked as

```
eft = ElementFreedomTable[enl,efs,nodfat,nodfmt,m]
```
(25.23)

The arguments are

`enl`	The element node list.
`efs`	The Element Freedom Signature (EFS) list.
`nodfat`	The Node Freedom Allocation Table (NFAT) described in 25.5.3
`nodfmt`	The Node Freedom Map Table (NFMT) described in 25.5.4
`m`	The NFA length, often 6.

The module returns the Element Freedom Table as function value. Examples of use of this module are provided in the plane trussed frame example below.

Remark 25.1. The EFT constructed by `ElementFreedomTable` is guaranteed to contain only positive entries if every freedom declared in the EFS matches a globally allocated freedom in the NFAT. But it is possible for the returned EFT to contain zero entries. This can only happen if an element freedom does not match a globally allocated one. For example suppose that a 2-node space beam element with the EFS (25.21) is placed into a program that does not accept rotational freedoms and thus has a NFS of 111000 at all structural nodes. Then six of the EFT entries, namely those pertaining to the rotational freedoms, would be zero.

The occurrence of a zero entry in a EFT normally flags a logic or input data error. If detected, the assembler should print an appropriate error message and abort. The module of Figure 25.10 does not make that decision because it lacks certain information, such as the element number, that should be placed into the message.

§25.5.7. A Plane Trussed Frame Structure

To illustrate the workings of a MET-VFC assembler we will follow the assembly process of the structure pictured in Figure 25.11(a). The finite element discretization, element disconnection and assembly process are illustrated in Figure 25.11(b,c,d). Although the structure is chosen to be plane to facilitate visualization, it will be considered within the context of a general purpose 3D program with the freedom arrangement (25.15) at each node.

The structure is a trussed frame. It uses two element types: plane (Bernoulli-Euler) beam-columns and plane bars. Geometric, material and fabrication data are given in Figure 25.11(a). The FEM idealization has 4 nodes and 5 elements. Nodes numbers are 1, 2, 3 and 5 as shown in Figure 25.11(b). Node 4 is purposely left out to illustrate handling of numbering gaps. There are 11 DOFs, three

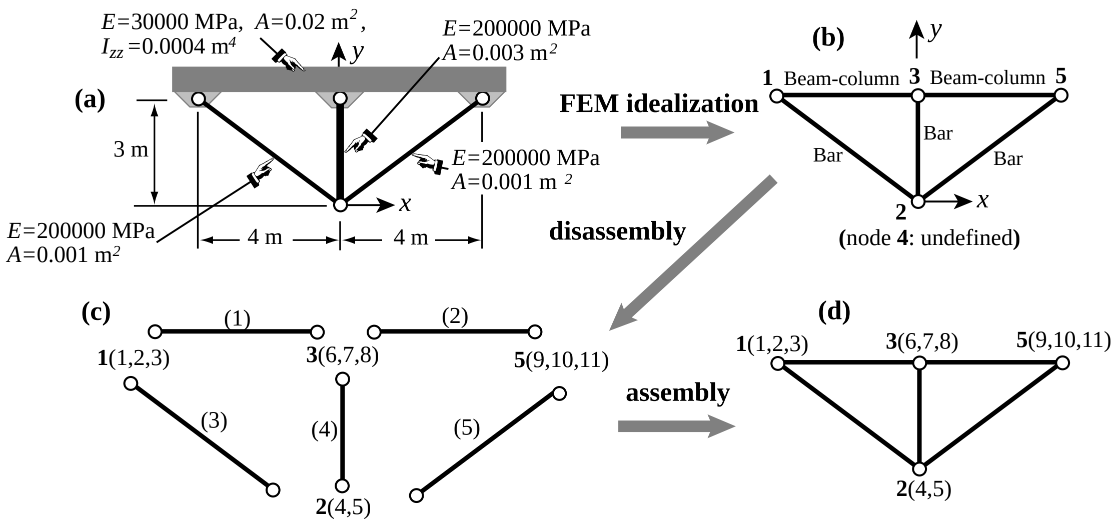

FIGURE 25.11. Trussed frame structure to illustrate a MET-VFC assembler: (a) original structure showing dimensions, material and fabrication properties; (b) finite element idealization with bars and beam column elements; (c) conceptual disassembly; (d) assembly. Numbers written in parentheses after a node number in (c,d) are the global freedom (equation) numbers allocated to that node.

at nodes 1, 3 and 5, and two at node 2. They are ordered as

Global DOF #:	1	2	3	4	5	6	7	8	9	10	11
DOF:	u_{x1}	u_{y1}	θ_{z1}	u_{x2}	u_{y2}	u_{x3}	u_{y3}	θ_{z3}	u_{x5}	u_{y5}	θ_{z5}
Node #:	1	1	1	2	2	3	3	3	5	5	5

(25.24)

The NFAT and NFMT can be constructed by inspection of (25.24) to be

```
nodfat = { 110001,110000,110001,000000,110001 }
nodfmt = { 0,3,5,8,8 }
```

(25.25)

The element freedom data structures can be also constructed by inspection:

Elem	Type	Nodes	EFS	EFT
(1)	Beam-column	`{ 1,3 }`	`{ 110001,110001 }`	`{ 1,2,3,6,7,8 }`
(2)	Beam-column	`{ 3,5 }`	`{ 110001,110001 }`	`{ 6,7,8,9,10,11 }`
(3)	Bar	`{ 1,2 }`	`{ 110000,110000 }`	`{ 1,2,4,5 }`
(4)	Bar	`{ 2,3 }`	`{ 110000,110000 }`	`{ 4,5,6,7 }`
(5)	Bar	`{ 2,5 }`	`{ 110000,110000 }`	`{ 4,5,9,10 }`

(25.26)

Next we list the element stiffness matrices, computed with the modules discussed in Chapter 20. The EFT entries are annotated as usual.

Element (1): This is a plane beam-column element with stiffness

$$\mathbf{K}^{(1)} = \left[\begin{array}{rrrrrr} 150. & 0. & 0. & -150. & 0. & 0. \\ 0. & 22.5 & 45. & 0. & -22.5 & 45. \\ 0. & 45. & 120. & 0. & -45. & 60. \\ -150. & 0. & 0. & 150. & 0. & 0. \\ 0. & -22.5 & -45. & 0. & 22.5 & -45. \\ 0. & 45. & 60. & 0. & -45. & 120. \end{array}\right] \begin{array}{l} \mathit{1} \\ \mathit{2} \\ \mathit{3} \\ \mathit{6} \\ \mathit{7} \\ \mathit{8} \end{array} \tag{25.27}$$

Element (2): This is a plane beam-column with element stiffness identical to the previous one

$$\mathbf{K}^{(2)} = \left[\begin{array}{rrrrrr} 150. & 0. & 0. & -150. & 0. & 0. \\ 0. & 22.5 & 45. & 0. & -22.5 & 45. \\ 0. & 45. & 120. & 0. & -45. & 60. \\ -150. & 0. & 0. & 150. & 0. & 0. \\ 0. & -22.5 & -45. & 0. & 22.5 & -45. \\ 0. & 45. & 60. & 0. & -45. & 120. \end{array}\right] \begin{array}{l} \mathit{6} \\ \mathit{7} \\ \mathit{8} \\ \mathit{9} \\ \mathit{10} \\ \mathit{11} \end{array} \tag{25.28}$$

Element (3): This is a plane bar element with stiffness

$$\mathbf{K}^{(3)} = \left[\begin{array}{rrrr} 25.6 & -19.2 & -25.6 & 19.2 \\ -19.2 & 14.4 & 19.2 & -14.4 \\ -25.6 & 19.2 & 25.6 & -19.2 \\ 19.2 & -14.4 & -19.2 & 14.4 \end{array}\right] \begin{array}{l} \mathit{1} \\ \mathit{2} \\ \mathit{4} \\ \mathit{5} \end{array} \tag{25.29}$$

Element (4): This is a plane bar element with stiffness

$$\mathbf{K}^{(4)} = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 200. & 0 & -200. \\ 0 & 0 & 0 & 0 \\ 0 & -200. & 0 & 200. \end{array}\right] \begin{array}{l} \mathit{4} \\ \mathit{5} \\ \mathit{6} \\ \mathit{7} \end{array} \tag{25.30}$$

Element (5): This is a plane bar element with stiffness

$$\mathbf{K}^{(5)} = \left[\begin{array}{rrrr} 25.6 & 19.2 & -25.6 & -19.2 \\ 19.2 & 14.4 & -19.2 & -14.4 \\ -25.6 & -19.2 & 25.6 & 19.2 \\ -19.2 & -14.4 & 19.2 & 14.4 \end{array}\right] \begin{array}{l} \mathit{4} \\ \mathit{5} \\ \mathit{9} \\ \mathit{10} \end{array} \tag{25.31}$$

Upon merging the 5 elements the master stiffness matrix becomes

$$\mathbf{K} = \left[\begin{array}{rrrrrrrrrrr} 175.6 & -19.2 & 0 & -25.6 & 19.2 & -150. & 0 & 0 & 0 & 0 & 0 \\ -19.2 & 36.9 & 45. & 19.2 & -14.4 & 0 & -22.5 & 45. & 0 & 0 & 0 \\ 0 & 45. & 120. & 0 & 0 & 0 & -45. & 60. & 0 & 0 & 0 \\ -25.6 & 19.2 & 0 & 51.2 & 0 & 0 & 0 & 0 & -25.6 & -19.2 & 0 \\ 19.2 & -14.4 & 0 & 0 & 228.8 & 0 & -200. & 0 & -19.2 & -14.4 & 0 \\ -150. & 0 & 0 & 0 & 0 & 300. & 0 & 0 & -150. & 0 & 0 \\ 0 & -22.5 & -45. & 0 & -200. & 0 & 245. & 0 & 0 & -22.5 & 45. \\ 0 & 45. & 60. & 0 & 0 & 0 & 0 & 240. & 0 & -45. & 60. \\ 0 & 0 & 0 & -25.6 & -19.2 & -150. & 0 & 0 & 175.6 & 19.2 & 0 \\ 0 & 0 & 0 & -19.2 & -14.4 & 0 & -22.5 & -45. & 19.2 & 36.9 & -45. \\ 0 & 0 & 0 & 0 & 0 & 0 & 45. & 60. & 0 & -45. & 120. \end{array}\right] \begin{array}{l} \mathit{1} \\ \mathit{2} \\ \mathit{3} \\ \mathit{4} \\ \mathit{5} \\ \mathit{6} \\ \mathit{7} \\ \mathit{8} \\ \mathit{9} \\ \mathit{10} \\ \mathit{11} \end{array} \tag{25.32}$$

in which global freedom numbers are annotated for convenience. Since all elements have been processed, (25.32) is the master stiffness matrix. It eigenvalues are

$$[\,460.456\;\;445.431\;\;306.321\;\;188.661\;\;146.761\;\;82.7415\;\;74.181\;\;25.4476\;\;0\;\;0\;\;0\,] \tag{25.33}$$

This has the correct rank of $11 - 3 = 8$.

Remark 25.2. For storage as a skyline matrix (Chapter 26) the template for (25.32) would look like

$$\mathbf{K} = \begin{bmatrix} 175.6 & -19.2 & 0 & -25.6 & 19.2 & -150. & & & & & \\ & 36.9 & 45. & 19.2 & -14.4 & 0 & -22.5 & 45. & & & \\ & & 120. & 0 & 0 & 0 & -45. & 60. & & & \\ & & & 51.2 & 0 & 0 & 0 & 0 & -25.6 & -19.2 & \\ & & & & 228.8 & 0 & -200. & 0 & -19.2 & -14.4 & \\ & & & & & 300. & 0 & 0 & -150. & 0 & \\ & & & & & & 245. & 0 & 0 & -22.5 & 45. \\ & & & & & & & 240. & 0 & -45. & 60. \\ & & & & & & & & 175.6 & 19.2 & 0 \\ & & & & & & & & & 36.9 & -45. \\ symm & & & & & & & & & & 120. \end{bmatrix} \tag{25.34}$$

The diagonal location pointers of (25.34) are defined by the list

$$\texttt{DLT = \{0,1,3,6,10,15,21,27,34,40,47,52\}} \tag{25.35}$$

Examination of the template (25.34) reveals that some additional zero entries could be removed from the template; for example K_{13}. The fact that those entries are zero is, however, fortuitous. It comes from the fact that some elements such as the beams are aligned along x, which decouples axial and bending stiffnesses.

§25.5.8. MET-VFC Assembler Implementation

Figure 25.12 lists the *Mathematica* implementation of a MET-VFC assembler capable of doing the trussed frame structure of Figure 25.11. The assembler module is invoked as

```
K = PlaneTrussedFrameMasterStiffness[nodxyz,eletyp,elenod,
                                elemat,elefab,nodfat,prcopt]          (25.36)
```

The only new argument with respect to the MET assembler of §25.4.2 is

`nodfat` The Node Freedom Arrangement Table (NFAT).

The module returns the master stiffness matrix as function value.

`PlaneTrussedFrameMasterStiffness` uses five other modules: the element stiffness modules `PlaneBar2Stiffness` and `PlaneBeamColumn2Stiffness` of Chapter 20, the NFMT constructor `NodeFreedomMapTable` of Figure 25.9, the total-DOF-counter `TotalFreedomCount` of Figure 25.9, and the EFT constructor `ElementFreedomTable` of Figure 25.10.

The element fabrication list `elefab` has minor modifications. A plane beam-column element requires two properties: `{ A,Izz }`, which are the cross section area and the moment of inertia about the local z axis. A bar element requires only the cross section area: `A`. For the example trussed frame structure `elefab` is `{ { 0.02,0.004 },{ 0.02,0.004 },0.001,0.003,0.001 }` in accordance with

```
PlaneTrussedFrameMasterStiffness[nodxyz_,eletyp_,
   elenod_,elemat_,elefab_,nodfat_,prcopt_]:=Module[
 {numele=Length[elenod],numnod=Length[nodxyz],numdof,nodfmt,e,enl,
  eftab,n,ni,nj,i,j,k,m,ncoor,Em,A,Izz,options,Ke,K},
  nodfmt=NodeFreedomMapTable[nodfat];
  numdof=TotalFreedomCount[nodfat]; K=Table[0,{numdof},{numdof}];
  For [e=1, e<=numele, e++, enl=elenod[[e]];
       If [eletyp[[e]]=="Bar2", {ni,nj}=enl;
           ncoor={nodxyz[[ni]],nodxyz[[nj]]};
           Em=elemat[[e]]; A=elefab[[e]]; options=prcopt;
           eftab=ElementFreedomTable[enl,{110000,110000},nodfat,nodfmt,6];
           Ke=PlaneBar2Stiffness[ncoor,Em,A,options] ];
       If [eletyp[[e]]=="BeamCol2", {ni,nj}=enl;
           ncoor={nodxyz[[ni]],nodxyz[[nj]]};
           eftab=ElementFreedomTable[enl,{110001,110001},nodfat,nodfmt,6];
           Em=elemat[[e]]; {A,Izz}=elefab[[e]]; options=prcopt;
           Ke=PlaneBeamColumn2Stiffness[ncoor,Em,{A,Izz},options] ];
       If [MemberQ[eftab,0], Print["Zero entry in eftab for element ",e];
           Print["Assembly process aborted"]; Return[K]];
       neldof=Length[eftab];
       For [i=1, i<=neldof, i++, ii=eftab[[i]];
           For [j=i, j<=neldof, j++, jj=eftab[[j]];
               K[[jj,ii]]=K[[ii,jj]]+=Ke[[i,j]] ];
           ];
      ]; Return[K]];
```

FIGURE 25.12. A MET-VFC assembler written for the plane trussed frame structure.

```
nodxyz={{-4,3},{0,0},{0,3},0,{4,3}};
eletyp= {"BeamCol2","BeamCol2","Bar2","Bar2","Bar2"};
elenod= {{1,3},{3,5},{1,2},{2,3},{2,5}};
elemat= Join[Table[30000,{2}],Table[200000,{3}]];
elefab= {{0.02,0.004},{0.02,0.004},0.001,0.003,0.001};
prcopt= {True};
nodfat={110001,110000,110001,000000,110001};
K=PlaneTrussedFrameMasterStiffness[nodxyz,eletyp,
       elenod,elemat,elefab,nodfat,prcopt]; K=Chop[K];
Print["Master Stiffness of Example Trussed Frame:"];
Print[K//MatrixForm];
Print["Eigs of K=",Chop[Eigenvalues[N[K]]]];
```

Master Stiffness of Example Trussed Frame:

$$\begin{pmatrix} 175.6 & -19.6 & 0 & -25.6 & 19.2 & -150. & 0 & 0 & 0 & 0 & 0 \\ -19.2 & 36.9 & 45. & 19.2 & -14.4 & 0 & -22.5 & 45. & 0 & 0 & 0 \\ 0 & 45. & 120. & 0 & 0 & 0 & -45. & 60. & 0 & 0 & 0 \\ -25.6 & 19.2 & 0 & 51.2 & 0 & 0 & 0 & 0 & -25.6 & -19.2 & 0 \\ 19.2 & -14.4 & 0 & 0 & 228.8 & 0 & -200. & 0 & -19.2 & -14.4 & 0 \\ -150. & 0 & 0 & 0 & 0 & 300. & 0 & 0 & -150. & 0 & 0 \\ 0 & -22.5 & -45. & 0 & -200. & 0 & 245. & 0 & 0 & -22.5 & 45. \\ 0 & 45. & 60. & 0 & 0 & 0 & 0 & 240. & 0 & -45. & 60. \\ 0 & 0 & 0 & -25.6 & -19.2 & -150. & 0 & 0 & 175.6 & 19.2 & 0 \\ 0 & 0 & 0 & 19.2 & -14.4 & 0 & -22.5 & -45. & 19.2 & 36.9 & -45. \\ 0 & 0 & 0 & 0 & 0 & 0 & 45. & 60. & 0 & -45. & 120. \end{pmatrix}$$

Eigs of K= {460.456, 445.431, 306.321, 188.661, 146.761, 82.7415, 74.181, 25.4476, 0, 0, 0}

FIGURE 25.13. Script to test the assembler of Figure 25.12 with the structure of Figure 25.11.

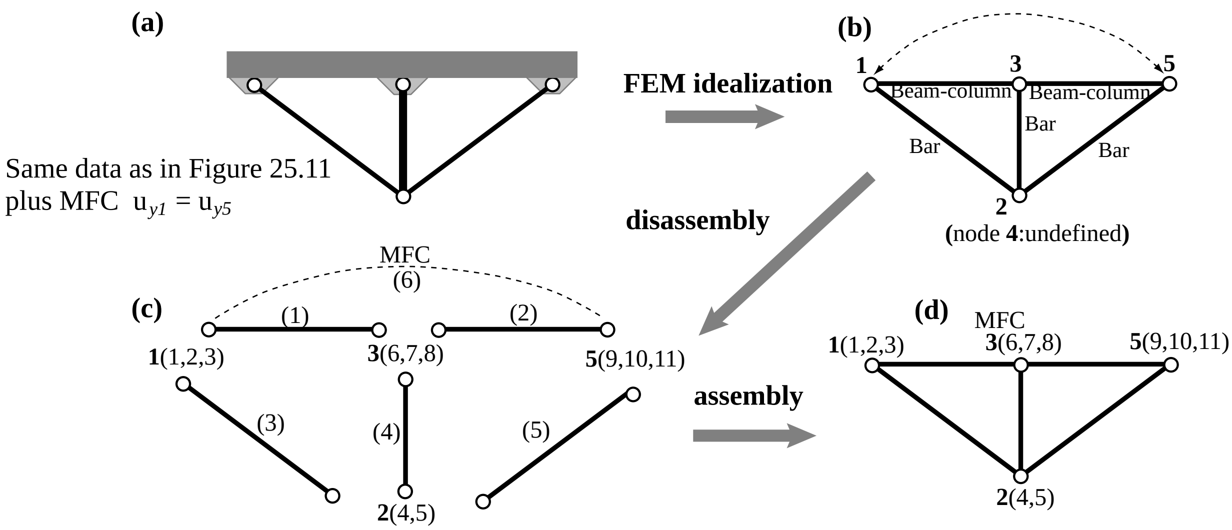

FIGURE 25.14. Finite element discretization, disassembly and assembly of trussed frame example structure with MFC $u_{y1} = u_{y5}$.

the data in Figure 25.11. The element material property list `elemat` is modified on the account that the elastic moduli for the beam columns (made of reinforced concrete) and bars made of (steel) are different.

Running the script listed in the top of Figure 25.13 produces the output shown in the bottom of that figure. As can be observed this agrees with (25.32) and (25.33).

§25.6. *Handling MultiFreedom Constraints

To see the effect of imposing an MFC through the Lagrange multiplier method on the configuration of the master stiffness matrix, suppose that the trussed frame example structure of the previous section is subjected to the constraint that nodes 1 and 5 must move vertically by the same amount. That is,

$$u_{y1} = u_{y5} \qquad \text{or} \qquad u_{y1} - u_{y5} = 0. \tag{25.37}$$

For assembly purposes (25.37) may be viewed as a fictitious element[4] labeled as (6). See Figure (25.14).

The degrees of freedom of the assembled structure increase by one to 12. They are ordered as

Global DOF #:	1	2	3	4	5	6	7	8	9	10	11	12
DOF:	u_{x1}	u_{y1}	θ_{z1}	u_{x2}	u_{y2}	u_{x3}	u_{y3}	θ_{z3}	u_{x5}	u_{y5}	θ_{z5}	$\lambda^{(6)}$
Node #:	1	1	1	2	2	3	3	3	5	5	5	none

(25.38)

The assembly of the first five elements proceeds as explained before, and produces the same master stiffness as (25.34), except for an extra zero row and column. Processing the MFC element (6) yields the 12 × 12

[4] This device should not be confused with penalty elements, which have stiffness matrices. The effect of adding a "Lagrange multiplier element" is to append rows and columns to the master stiffness.

bordered stiffness

$$
\mathbf{K} = \begin{bmatrix}
175.6 & -19.2 & 0 & -25.6 & 19.2 & -150. & 0 & 0 & 0 & 0 & 0 & 0 \\
-19.2 & 36.9 & 45. & 19.2 & -14.4 & 0 & -22.5 & 45. & 0 & 0 & 0 & 1. \\
0 & 45. & 120. & 0 & 0 & 0 & -45. & 60. & 0 & 0 & 0 & 0 \\
-25.6 & 19.2 & 0 & 51.2 & 0 & 0 & 0 & 0 & -25.6 & -19.2 & 0 & 0 \\
19.2 & -14.4 & 0 & 0 & 228.8 & 0 & -200. & 0 & -19.2 & -14.4 & 0 & 0 \\
-150. & 0 & 0 & 0 & 0 & 300. & 0 & 0 & -150. & 0 & 0 & 0 \\
0 & -22.5 & -45. & 0 & -200. & 0 & 245. & 0 & 0 & -22.5 & 45. & 0 \\
0 & 45. & 60. & 0 & 0 & 0 & 0 & 240. & 0 & -45. & 60. & 0 \\
0 & 0 & 0 & -25.6 & -19.2 & -150. & 0 & 0 & 175.6 & 19.2 & 0 & 0 \\
0 & 0 & 0 & -19.2 & -14.4 & 0 & -22.5 & -45. & 19.2 & 36.9 & -45. & -1. \\
0 & 0 & 0 & 0 & 0 & 0 & 45. & 60. & 0 & -45. & 120. & 0 \\
0 & 1. & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1. & 0 & 0
\end{bmatrix} \tag{25.39}
$$

in which the coefficients 1 and -1 associated with the MFC (25.37) end up in the last row and column of $\mathbf{K}$.

There are several ways to incorporate multiplier adjunction into an automatic assembly procedure. The most elegant ones associate the fictitious element with a special freedom signature and a reserved position in the NFA. Being of advanced nature such schemes are beyond the scope of this Chapter.

Notes and Bibliography

The DSM assembly process for the simplest case of §25.3 is explained in any finite element text. Few texts cover, however, the complications that arise in more general scenarios. Especially when VFCs are allowed.

Assembler implementation flavors have fluctuated since the DSM became widely accepted as standard. Variants have been strongly influenced by limits on random access memory (RAM). Those limitations forced the use of "out-of-core" blocked equation solvers, meaning that heavy use was made of disk auxiliary storage to store and retrieve the master stiffness matrix in blocks. Only a limited number of blocks could reside on RAM. Thus a straightforward element-by-element assembly loop (as in the assemblers presented here) was likely to "miss the target" on the receiving end of the merge, forcing blocks to be read in, modified and saved. On large problems this swapping was likely to "trash" the system with heavy I/O, bringing processing to a crawl.

One solution favored in programs of the 1965–85 period was to process all elements without assembling, saving matrices on disk. The assembler then cycled over stiffness blocks and read in the contributing elements. With smart asynchronous buffering and tuned-up direct access I/O such schemes were able to achieved reasonable efficiency. However, system dependent logic quickly becomes a maintenance nightmare.

A second way out emerged by 1970: the frontal solvers referenced in Chapter 11. A frontal solver carries out assembly, BC application and solution concurrently. Element contributions are processed in a special order that traverses the FEM mesh as a "wavefront." Application of displacement BCs and factorization can "trail the wave" once no more elements are detected as contributing to a given equation. As can be expected of trying to do too much at once, frontal solvers can be extremely sensitive to changes. One tiny alteration in the element library over which the solver sits, and the whole thing may crumble like a house of cards.

The availability of large amounts of RAM (even on PCs and laptops) since the mid 90s, has had a happy consequence: interweaved assembly and solver implementations, as well as convoluted matrix blocking, are no longer needed. The assembler can be modularly separated from the solver. As a result even the most complex assembler presented here fits in one page of text. Of course it is too late for the large scale FEM codes that got caught in the limited-RAM survival game decades ago. Changing their assemblers and solvers incrementally is virtually impossible. The gurus that wrote those thousands of lines of spaghetti Fortran are long gone. The only practical way out is rewrite the whole shebang from scratch. In a commercial environment such investment-busting decisions are unlikely.

Homework Exercises for Chapter 25
The Assembly Process

EXERCISE 25.1 [D:10] Suppose you want to add a six-node plane stress quadratic triangle to the MET assembler of §25.4. Sketch how you would modify the module of Figure 25.6 for this to happen.

EXERCISE 25.2 [C:10] Exercise the module `ElementFreedomTable` listed in Figure 25.10 on the trussed frame structure. Verify that it produces the last column of (25.26).

EXERCISE 25.3 [D:15] Describe the logic of module `ElementFreedomTable` listed in Figure 25.10. Can it return zero entries in the EFT? If yes, give a specific example of how this can happen. Hint: read the Chapter carefully.

EXERCISE 25.4 [A/C:30] The trussed frame structure of Figure 25.4 is reinforced with two triangular steel plates attached as shown in Figure **?**(a). The plate thickness is 1.6 mm= 0.0016 m; the material is isotropic with $E = 240000$ MPa and $\nu = 1/3$. Each reinforcing plate is modeled with a single plane stress 3-node linear triangle. The triangles are numbered (6) and (7), as illustrated in Figure E25.1(c). Compute the master stiffness matrix **K** of the structure.[5]

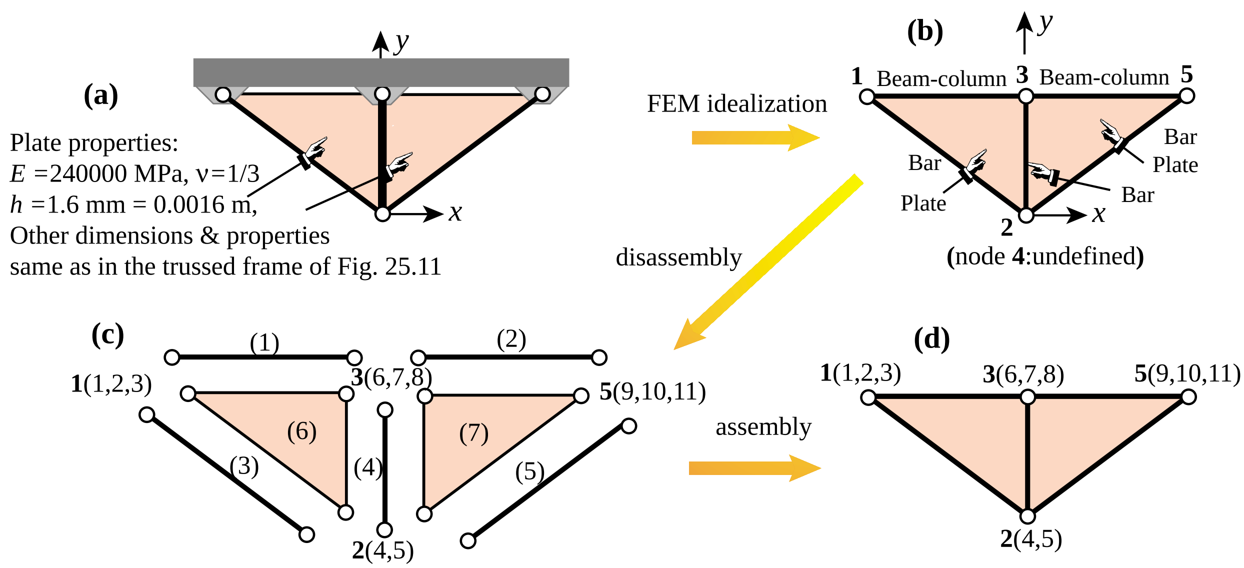

FIGURE E25.1. Plate-reinforced trussed frame structure for Exercise 25.4.

This exercise may be done through *Mathematica*. For this download Notebook `ExampleAssemblers.nb` from this Chapter index, and complete Cell 4 by writing the assembler.[6] The plate element identifier is `"Trig3"`. The driver script to run the assembler is also provided in Cell 4 (blue text).

[5] This structure would violate the compatibility requirements stated in Chapter 19 unless the beams are allowed to deflect laterally independently of the plates. This is the fabrication actually sketched in Figure E25.1(a).

[6] Cells 1–3 contain the assemblers presented in §25.3, §25.4 and §25.5, respectively, which may be used as guides. The stiffness modules for the three element types used in this structure are available in Cells 2 and 3 and may be reused for this Exercise.

When reusing the assembler of Cell 3 as a guide, please do not remove the internal Print commands that show element information (red text). Those come in handy for debugging. Please keep that printout in the returned homework to help the grader.

Another debugging hint: check that the master stiffness (25.32) is obtained if the plate thickness, called `hplate` in the driver script, is temporarily set to zero.

Target:

$$\mathbf{K} = \begin{bmatrix} 337.6 & -19.2 & 0 & -25.6 & 91.2 & -312. & -72. & 0 & 0 & 0 & 0 \\ -19.2 & 90.9 & 45. & 91.2 & -14.4 & -72. & -76.5 & 45. & 0 & 0 & 0 \\ 0 & 45. & 120. & 0 & 0 & 0 & -45. & 60. & 0 & 0 & 0 \\ -25.6 & 91.2 & 0 & 243.2 & 0 & -192. & 0 & 0 & -25.6 & -91.2 & 0 \\ 91.2 & -14.4 & 0 & 0 & 804.8 & 0 & -776. & 0 & -91.2 & -14.4 & 0 \\ -312. & -72. & 0 & -192. & 0 & 816. & 0 & 0 & -312. & 72. & 0 \\ -72. & -76.5 & -45. & 0 & -776. & 0 & 929. & 0 & 72. & -76.5 & 45. \\ 0 & 45. & 60. & 0 & 0 & 0 & 0 & 240. & 0 & -45. & 60. \\ 0 & 0 & 0 & -25.6 & -91.2 & -312. & 72. & 0 & 337.6 & 19.2 & 0 \\ 0 & 0 & 0 & -91.2 & -14.4 & 72. & -76.5 & -45. & 19.2 & 90.9 & -45. \\ 0 & 0 & 0 & 0 & 0 & 0 & 45. & 60. & 0 & -45. & 120. \end{bmatrix} \tag{E25.1}$$

EXERCISE 25.5 [A:30] §25.5 does not explain how to construct the NFAT from the input data. (In the trussed frame example script, `nodfat` was set up by inspection, which is OK only for small problems.) Explain how this table could be constructed automatically if the EFS of each element in the model is known. Note: the logic is far from trivial.