6 Finite Element Domain Decomposition Procedures

6.1 Introduction

The finite element (statics) equilibrium equations can be given as

$$[K(b)] \cdot \vec{z} = \vec{S}(b) \tag{6.1}$$

where:

$\vec{b}$ = design variable vector, such as cross-sectional areas of truss members, moment of inertia of beam members, and/or thickness of plate (or shell) members.

$\vec{z}$ = nodal displacement vector

$[K]$ = stiffness matrix

$\vec{S}$ = nodal load vector

Equation (6.1) can be partitioned as

$$\begin{bmatrix} K_{BB} & K_{BI} \\ K_{IB} & K_{II} \end{bmatrix} \times \begin{Bmatrix} z_B \\ z_I \end{Bmatrix} = \begin{Bmatrix} s_B \\ s_I \end{Bmatrix} \tag{6.2}$$

where subscripts B and I denote Boundary and Interior terms, respectively.

From the 2nd part of Eq. (6.2), one has:

$$K_{IB} z_B + K_{II} z_I = s_I \tag{6.3}$$

or

$$\vec{z_I} = [K_{II}]^{-1} \cdot \left(\vec{s_I} - K_{IB} \vec{z_B} \right) \tag{6.4}$$

From the 1st part of Eq. (6.2), one has:

$$K_{BB} z_B + K_{BI} z_I = s_B \tag{6.5}$$

Substitution Eq. (6.4) into Eq. (6.5), one obtains:

$$K_{BB} z_B + K_{BI} . [K_{II}]^{-1} . \left(s_I - K_{IB} z_B \right) = s_B \tag{6.6}$$

or

$$\left[K_{BB} - K_{BI} K_{II}^{-1} K_{IB} \right] . \vec{z_B} = \left(s_B - K_{BI} K_{II}^{-1} s_I \right) \tag{6.7}$$

Let $\overline{\mathrm{K_B}} \equiv \mathrm{K_{BB}} - \mathrm{K_{BI}}\mathrm{K_{II}^{-1}}\mathrm{K_{IB}} \equiv$ Effective Boundary Stiffness (6.8)

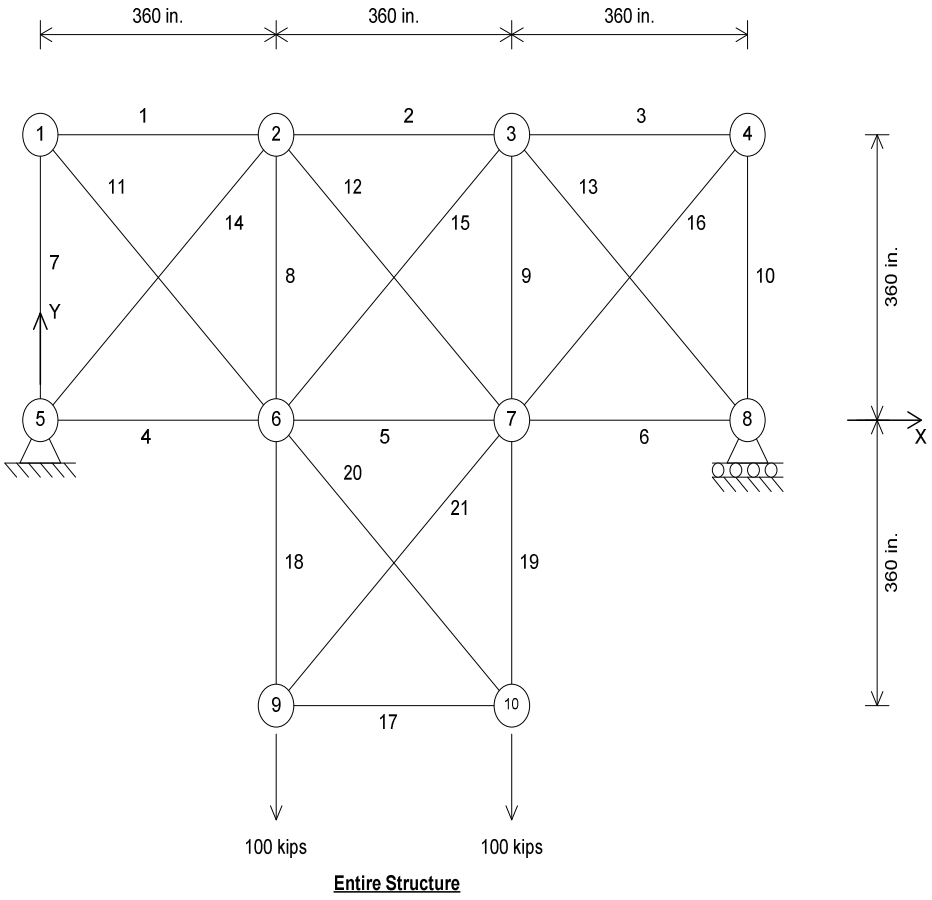
and $\overline{s_B} \equiv s_B - K_{BI}K_{II}^{-1}s_I \equiv$ Effective Boundary Load (6.9)

Equation (6.8) for $\overline{K_B}$ is also called the Schur complement matrix.

Then, Eq. (6.7) can be expressed as:

$$\left[\overline{K_B}\right] \cdot z_B = \overline{s_B} \tag{6.10}$$

Having solved the boundary displacement vector z_B from Eq. (6.10), the interior displacement vector z_I can be solved from Eq. (6.4)



Entire Structure

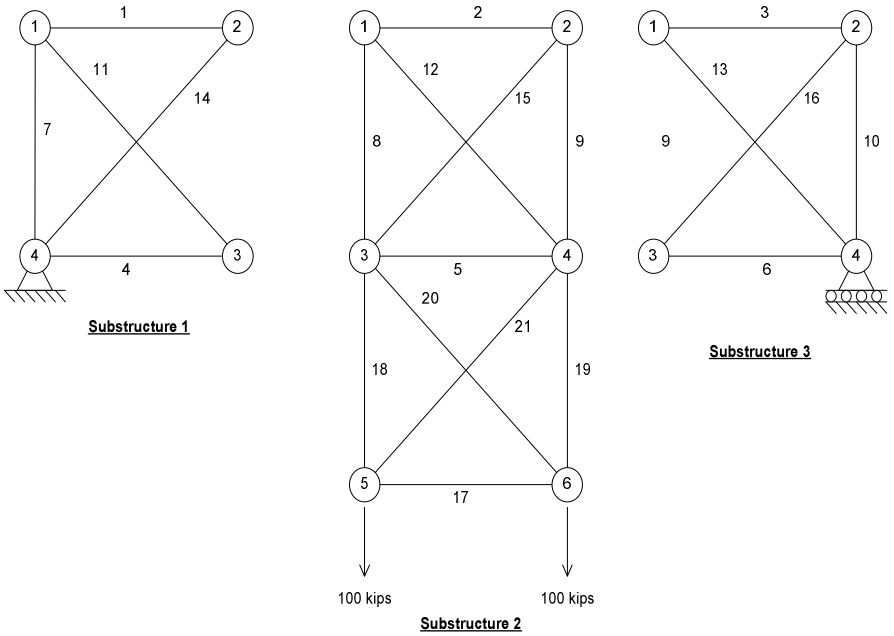


Figure 6.1: A structure is divided into 3 substructures

For very large-scale problems, the original structure can be partitioned into many smaller sub-structures (or sub-domains), see Figure 6.1. Thus, for a typical $r^{\underline{th}}$ sub-domain, one obtains:

$$z_I^{(r)} = \left[K_{II}^{(r)} \right]^{-1} \cdot \left(s_I^{(r)} - K_{IB}^{(r)} z_B^{(r)} \right) \tag{6.11}$$

Similarly, one defines:

$$\overline{K_B}^{(r)} \equiv K_{BB}^{(r)} - K_{BI}^{(r)} \left[K_{II}^{(r)} \right]^{-1} K_{IB}^{(r)} \tag{6.12}$$

$$\overline{s_B}^{(r)} \equiv s_B^{(r)} - K_{BI}^{(r)} \left[K_{II}^{(r)} \right]^{-1} s_I^{(r)} \tag{6.13}$$

The overall (or system) effective boundary stiffness matrix can be assembled and solved as:

$$\overline{K_B} = \sum_{r=1}^{NSU} \overline{K_B}^{(r)} \tag{6.14}$$

and $$\overline{s_B} = \sum_{r=1}^{NSU} \overline{s_B}^{(r)} \tag{6.15}$$

$$\overline{K_B} \cdot z_B = \overline{s_B} \tag{6.16}$$

where NSU ≡Number of SUbstructures

Remarks

(a) For large-scale problems, the operations involved in Eq. (6.12) are quite expensive, because

(i) $\left[K_{II}^{(r)}\right]^{-1} \cdot K_{IB}^{(r)}$ requires solving system of linear equations with "many" right-hand-side vectors.

(ii) $\left[K_{BI}^{(r)}\right] \cdot \left[\left[K_{II}^{(r)}\right]^{-1} \cdot K_{IB}^{(r)}\right]$ requires "sparse matrix*dense matrix" operations.

(b) Even though each individual matrices $\left[K_{BI}^{(r)}\right], \left[K_{II}^{(r)}\right]$ and $\left[K_{IB}^{(r)}\right]$ are sparse, the triple matrix products (shown in the right-hand-side of Eq. (6.12)) can be "nearly dense". Thus, computer memory requirements can also be quite large.

6.2 A Simple Numerical Example Using Domain Decomposition (DD) Procedures

A simple 2-D truss structure [6.1], shown in Figures 6.2-6.4, will be used to illustrate detailed steps involved in the DD procedures (E = Kips/in^2, A = in^2, for all members)

For each rth substructure, the boundary nodes (or boundary dofs) are numbered first (and start with 1), then the interior nodes (or interior dofs) are subsequently numbered. Element numbers for each rth substructure should also start with 1. In Figure 6.3, while nodes 1 and 2 must be considered as boundary nodes, any interior node of the rth substructure can also be selected as a boundary node. The arrow-directions, shown in Figure 6.3, indicate that a particular truss member is connected from node "i" to node "j". The mapping between the local (substructuring) boundary degree-of-freedom (dof) and global (overall) boundary dof can be identified by comparing Figures 6.3 and 6.4. Thus, in substructure 1 (r = 1), boundary nodes 1 and 2 will correspond to the system (overall) boundary nodes 2 and 1, respectively.

Two vertical forces are applied at nodes 2 and 3 (as shown in Figure 6.2), while the prescribed displacements (say, due to support settlements) are shown at node 4 (of substructure r = 1, see Figure 6.3), and at node 4 (of substructure r = 2, see Figure 6.3)

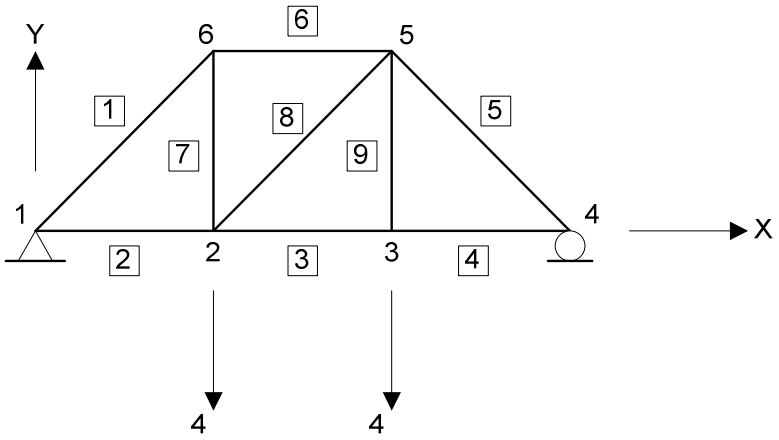


Figure 6.2: Overall original system

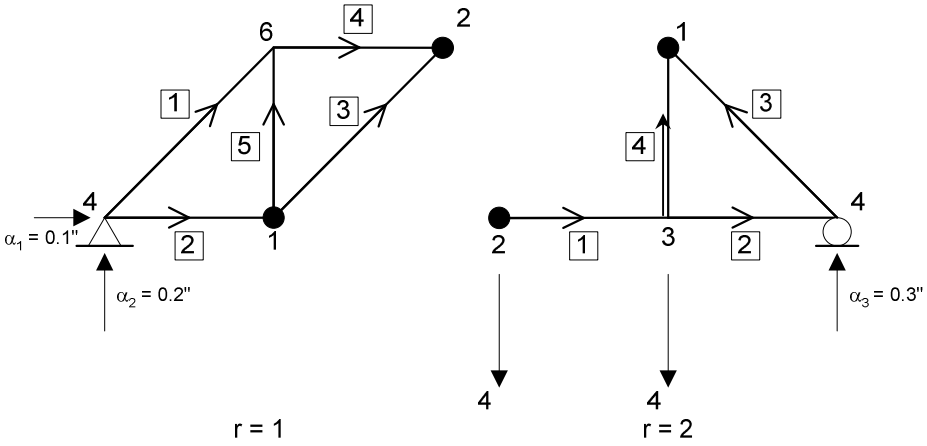


Figure 6.3: Local (substructuring) numbering system

Figure 6.4: Overall boundary nodes numbering system

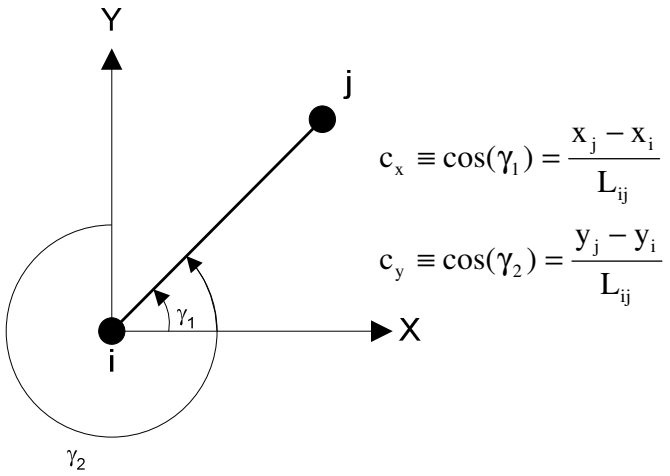


Figure 6.5: Direction Cosines of a 2-D Truss Member

Based on Chapter 1, the 2-D truss element stiffness matrix, in global coordinate X,Y axis, can be given as

$$\left[k^{(e)}\right] = \left(\frac{EA}{L}\right)\begin{bmatrix} c_x^2 & c_x c_y & -c_x^2 & -c_x c_y \\ c_x c_y & c_y^2 & -c_x c_y & -c_y^2 \\ -c_x^2 & -c_x c_y & c_x^2 & c_x c_y \\ -c_x c_y & -c_y^2 & c_x c_y & c_y^2 \end{bmatrix} \tag{6.17}$$

where c_x and c_y have been defined in Figure 6.5. Thus for the 1st substructure (r = 1), one obtains the following element (global) stiffness matrices:

$$k_G^1 = \begin{array}{c|cccc} & 7 & 8 & 5 & 6 \\ 7 & 42.7209 & 42.7209 & -42.7209 & -42.7209 \\ 8 & 42.7209 & 42.7209 & -42.7209 & -42.7209 \\ 5 & -42.7209 & -42.7209 & 42.7209 & 42.7209 \\ 6 & -42.7209 & -42.7209 & 42.7209 & 42.7209 \end{array} \tag{6.18}$$

$$k_G^2 = \begin{array}{c|cccc} & 7 & 8 & 1 & 2 \\ 7 & 120.8333 & 0 & -120.8333 & 0 \\ 8 & 0 & 0 & 0 & 0 \\ 1 & -120.8333 & 0 & 120.8333 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{array} \tag{6.19}$$

$$k_G^3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 42.7209 & 42.7209 & -42.7209 & -42.7209 \\ 2 & 42.7209 & 42.7209 & -42.7209 & -42.7209 \\ 3 & -42.7209 & -42.7209 & 42.7209 & 42.7209 \\ 4 & -42.7209 & -42.7209 & 42.7209 & 42.7209 \end{array} \quad (6.20)$$

$$k_G^4 = \begin{array}{c|cccc} & 5 & 6 & 3 & 4 \\ 5 & 120.8333 & 0 & -120.8333 & 0 \\ 6 & 0 & 0 & 0 & 0 \\ 3 & -120.8333 & 0 & 120.8333 & 0 \\ 4 & 0 & 0 & 0 & 0 \end{array} \quad (6.21)$$

$$k_G^5 = \begin{array}{c|cccc} & 1 & 2 & 5 & 6 \\ 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 120.8333 & 0 & -120.8333 \\ 5 & 0 & 0 & 0 & 0 \\ 6 & 0 & -120.8333 & 0 & 120.8333 \end{array} \quad (6.22)$$

submatrices $\left[K_{BB}^{(r=1)}\right]$, $\left[K_{BI}^{(r=1)}\right]$, $\left[K_{IB}^{(r=1)}\right]$ and $\left[K_{II}^{(r=1)}\right]$ of substructure 1 can be assembled as:

$$K_{BB}^{(r=1)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 163.5542 & 42.7209 & -42.7209 & -42.7209 \\ 2 & 42.7209 & 163.5542 & -42.7209 & -42.7209 \\ 3 & -42.7209 & -42.7209 & 163.5542 & 42.7209 \\ 4 & -42.7209 & -42.7209 & 42.7209 & 42.7209 \end{array} \quad (6.23)$$

$$K_{BI}^{(r=1)} = \begin{array}{c|cccc} & 5 & 6 & 7 & 8 \\ 1 & 0 & 0 & -120.8333 & 0 \\ 2 & 0 & -120.8333 & 0 & 0 \\ 3 & -120.8333 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \end{array} \quad (6.24)$$

$$K_{II}^{(r=1)} = \begin{array}{c|cccc} & 5 & 6 & 7 & 8 \\ 5 & 163.5542 & 42.7209 & -42.7209 & -42.7209 \\ 6 & 42.7209 & 163.5542 & -42.7209 & -42.7209 \\ 7 & -42.7209 & -42.7209 & 163.5542 & 42.7209 \\ 8 & -42.7209 & -42.7209 & 42.7209 & 42.7209 \end{array} \quad (6.25)$$

$$K_{IB}^{(r=1)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 5 & 0 & 0 & -120.8333 & 0 \\ 6 & 0 & -120.8333 & 0 & 0 \\ 7 & -120.8333 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 \end{array} \quad (6.26)$$

After imposing the support boundary conditions, the above 4 equations become:

$$K_{BB\ b.c.}^{(r=1)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 163.5542 & 42.7209 & -42.7209 & -42.7209 \\ 2 & 42.7209 & 163.5542 & -42.7209 & -42.7209 \\ 3 & -42.7209 & -42.7209 & 163.5542 & 42.7209 \\ 4 & -42.7209 & -42.7209 & 42.7209 & 42.7209 \end{array} \tag{6.27}$$

$$K_{BI\ b.c.}^{(r=1)} = \begin{array}{c|cccc} & 5 & 6 & 7 & 8 \\ 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & -120.8333 & 0 & 0 \\ 3 & -120.8333 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \end{array} \tag{6.28}$$

$$K_{II\ b.c.}^{(r=1)} = \begin{array}{c|cccc} & 5 & 6 & 7 & 8 \\ 5 & 163.5542 & 42.7209 & 0 & 0 \\ 6 & 42.7209 & 163.5542 & 0 & 0 \\ 7 & 0 & 0 & 1 & 0 \\ 8 & 0 & 0 & 0 & 1 \end{array} \tag{6.29}$$

$$K_{IB\ b.c.}^{(r=1)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 5 & 0 & 0 & -120.8333 & 0 \\ 6 & 0 & -120.8333 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 \end{array} \tag{6.30}$$

The boundary and interior load vectors can be obtained as

$$F_B^{(r=1)} = \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}, \quad F_I^{(r=1)} = \begin{array}{c} 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{6.31}$$

After imposing the support (prescribed) boundary conditions, Eq. (6.31) becomes:

$$F_{B\ b.c.}^{(r=1)} = \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{Bmatrix} 12.08333 \\ 0 \\ 0 \\ 0 \end{Bmatrix}, \quad F_{I\ b.c.}^{(r=1)} = \begin{array}{c} 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{Bmatrix} 12.81627 \\ 12.81627 \\ 0.1 \\ 0.2 \end{Bmatrix} \tag{6.32}$$

The effective boundary stiffness matrix, and load vector can be computed from Eqs. (6.12-6.13) and, and are given as

$$\overline{K}_B^{(r=1)} = \begin{array}{c|cccc} & 3 & 4 & 1 & 2 \\ 3 & 163.5542 & 42.7209 & -42.7209 & -42.7209 \\ 4 & 42.7209 & 67.7463 & -17.6955 & -42.7209 \\ 1 & -42.7209 & -17.6955 & 67.7463 & 42.7209 \\ 2 & -42.7209 & -42.7209 & 42.7209 & 42.7209 \end{array} \tag{6.33}$$

$$\overline{F}_B^{(r=1)} = \begin{Bmatrix} 12.08333 \\ 7.5076 \\ 7.5076 \\ 0 \end{Bmatrix} \tag{6.34}$$

Similarly, for substructure r = 2, element stiffness matrices (in global coordinate references) can be computed from Eq. (6.17) as:

$$k_G^1 = \begin{array}{c|cccc} & 3 & 4 & 5 & 6 \\ 3 & 120.8333 & 0 & -120.8333 & 0 \\ 4 & 0 & 0 & 0 & 0 \\ 5 & -120.8333 & 0 & 120.8333 & 0 \\ 6 & 0 & 0 & 0 & 0 \end{array} \tag{6.35}$$

$$k_G^2 = \begin{array}{c|cccc} & 5 & 6 & 7 & 8 \\ 5 & 120.8333 & 0 & -120.8333 & 0 \\ 6 & 0 & 0 & 0 & 0 \\ 7 & -120.8333 & 0 & 120.8333 & 0 \\ 8 & 0 & 0 & 0 & 0 \end{array} \tag{6.36}$$

$$k_G^3 = \begin{array}{c|cccc} & 7 & 8 & 1 & 2 \\ 7 & 42.7209 & -42.7209 & -42.7209 & 42.7209 \\ 8 & -42.7209 & 42.7209 & 42.7209 & -42.7209 \\ 1 & -42.7209 & 42.7209 & 42.7209 & -42.7209 \\ 2 & 42.7209 & -42.7209 & -42.7209 & 42.7209 \end{array} \tag{6.37}$$

$$k_G^4 = \begin{array}{c|cccc} & 5 & 6 & 1 & 2 \\ 5 & 0 & 0 & 0 & 0 \\ 6 & 0 & 120.8333 & 0 & -120.8333 \\ 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & -120.8333 & 0 & 120.8333 \end{array} \tag{6.38}$$

submatrices $\left[K_{BB}^{(r=2)}\right]$, $\left[K_{BI}^{(r=2)}\right]$, $\left[K_{IB}^{(r=2)}\right]$ and $\left[K_{II}^{(r=2)}\right]$ can be assembled as:

$$K_{BB}^{(r=2)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 42.7209 & -42.7209 & 0 & 0 \\ 2 & -42.7209 & 163.5542 & 0 & 0 \\ 3 & 0 & 0 & 120.8333 & 0 \\ 4 & 0 & 0 & 0 & 0 \end{array} \tag{6.39}$$

$$K_{BI}^{(r=2)} = \begin{array}{c|cccc} & 5 & 6 & 7 & 8 \\ 1 & 0 & 0 & -42.7209 & 42.7209 \\ 2 & 0 & -120.8333 & 42.7209 & -42.7209 \\ 3 & -120.8333 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \end{array} \tag{6.40}$$

$$K_{IB}^{(r=2)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 5 & 0 & 0 & -120.8333 & 0 \\ 6 & 0 & -120.8333 & 0 & 0 \\ 7 & -42.7209 & 42.7209 & 0 & 0 \\ 8 & 42.7209 & -42.7209 & 0 & 0 \end{array} \tag{6.41}$$

$$K_{II}^{(r=1)} = \begin{array}{c|cccc} & 5 & 6 & 7 & 8 \\ 5 & 241.6666 & 0 & -120.8333 & 0 \\ 6 & 0 & 120.8333 & 0 & 0 \\ 7 & -120.8333 & 0 & 163.5542 & -42.7209 \\ 8 & 0 & 0 & -42.7209 & 42.7209 \end{array} \tag{6.42}$$

After imposing the support boundary conditions, Eqs. (6.39-6.42) become:

$$K_{BB\ b.c.}^{(r=2)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 42.7209 & -42.7209 & 0 & 0 \\ 2 & -42.7209 & 163.5542 & 0 & 0 \\ 3 & 0 & 0 & 120.8333 & 0 \\ 4 & 0 & 0 & 0 & 0 \end{array} \tag{6.43}$$

$$K_{BI\ b.c.}^{(r=2)} = \begin{array}{c|cccc} & 5 & 6 & 7 & 8 \\ 1 & 0 & 0 & -42.7209 & 0 \\ 2 & 0 & -120.8333 & 42.7209 & 0 \\ 3 & -120.8333 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \end{array} \tag{6.44}$$

$$K_{IB\ b.c.}^{(r=2)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 5 & 0 & 0 & -120.8333 & 0 \\ 6 & 0 & -120.8333 & 0 & 0 \\ 7 & -42.7209 & 42.7209 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 \end{array} \tag{6.45}$$

$$K_{II\ b.c.}^{(r=1)} = \begin{array}{c|cccc} & 5 & 6 & 7 & 8 \\ 5 & 241.6666 & 0 & -120.8333 & 0 \\ 6 & 0 & 120.8333 & 0 & 0 \\ 7 & -120.8333 & 0 & 163.5542 & 0 \\ 8 & 0 & 0 & 0 & 1 \end{array} \tag{6.46}$$

The boundary and interior load vectors of substructure r = 2 can be computed as

$$F_B^{(r=2)} = \begin{matrix}1\\2\\3\\4\end{Bmatrix}\begin{Bmatrix}0\\0\\0\\-4\end{Bmatrix}, \quad F_I^{(r=2)} = \begin{matrix}5\\6\\7\\8\end{Bmatrix}\begin{Bmatrix}0\\-4\\0\\0\end{Bmatrix} \tag{6.47}$$

After imposing the support boundary conditions, Eq. (6.47) becomes:

$$F_B^{(r=2)}{}_{b.c.} = \begin{matrix}1\\2\\3\\4\end{Bmatrix}\begin{Bmatrix}0\\-4\\-12.81627\\-12.81627\end{Bmatrix}, \quad F_I^{(r=2)}{}_{b.c.} = \begin{matrix}5\\6\\7\\8\end{Bmatrix}\begin{Bmatrix}0\\-4\\12.81627\\0.3\end{Bmatrix} \tag{6.48}$$

Applying Eqs. (6.12-6.13), the effective boundary stiffness matrix, and load vector can be computed as:

$$\overline{K}_B^{(r=2)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 25.0254 & -25.0254 & -25.0254 & 0 \\ 2 & -25.0254 & 25.0254 & 25.0254 & 0 \\ 3 & -25.0254 & 25.0254 & 25.0254 & 0 \\ 4 & 0 & 0 & 0 & 0 \end{array} \tag{6.49}$$

$$\overline{F}_B^{(r=2)} = \begin{Bmatrix}7.5074\\-4\\-7.5074\\3.5074\end{Bmatrix} \tag{6.50}$$

The overall (system) effective boundary load vector, and stiffness matrix can be assembled as:

$$\overline{F_B} = \begin{Bmatrix}19.5907\\3.5076\\0.0002\\3.5074\end{Bmatrix} \tag{6.51}$$

$$\overline{K}_B^{(r=2)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 92.7711 & 17.6955 & -67.7463 & -17.6955 \\ 2 & 17.6955 & 67.7463 & -17.6955 & -42.7209 \\ 3 & -67.7463 & -17.6955 & 188.5796 & 42.7209 \\ 4 & -17.6955 & -42.7209 & 42.7209 & 67.7463 \end{array} \tag{6.52}$$

All boundary displacements can be solved from Eq. (6.16) as:

$$\underset{z_B}{\overset{\text{—}}{=}} \begin{matrix} 3 \\ 4 \\ 1 \\ 2 \end{matrix} \begin{Bmatrix} 0.133 \\ 0.054 \\ 0.089 \\ 0.098 \end{Bmatrix} \tag{6.53}$$

The interior displacements of each substructure can be recovered from Eq. (6.11)

$$z_B^{(r=1)} = \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} \begin{Bmatrix} 0.089 \\ 0.098 \\ 0.133 \\ 0.054 \end{Bmatrix} \qquad z_I^{(r=1)} = \begin{matrix} 5 \\ 6 \\ 7 \\ 8 \end{matrix} \begin{Bmatrix} 0.121 \\ 0.085 \\ 0.1 \\ 0.2 \end{Bmatrix} \tag{6.54}$$

$$z_B^{(r=2)} = \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} \begin{Bmatrix} 0.133 \\ 0.054 \\ 0.089 \\ 0.098 \end{Bmatrix} \qquad z_I^{(r=2)} = \begin{matrix} 5 \\ 6 \\ 7 \\ 8 \end{matrix} \begin{Bmatrix} 0.165 \\ 0.065 \\ 0.198 \\ 0.3 \end{Bmatrix} \tag{6.55}$$

6.3 Imposing Boundary Conditions on "Rectangular" Matrices $\left[K_{BI}^{(r)} \right]$

Imposing the Dirichlet boundary conditions on the "square" substructuring stiffness matrices $\left[K_{BB}^{(r=1)}{}_{b.c.} \right]$ and $\left[K_{II}^{(r=1)}{}_{b.c.} \right]$ (see Eqs. 6.27,6.29) are rather straight forward, especially this topic has already been explained with great details in Chapters 1 and 4. Thus, the following paragraphs are intended to explain how to impose the Dirichlet boundary conditions on the rectangular (in general) substructuring stiffness matrix $\left[K_{BI}^{(r=1)}{}_{b.c.} \right]$, such as the one shown in Eq. (6.28). To facilitate the discussions, let's consider the entire stiffness matrix of the first substructure (r=1), as shown in Figure 6.3

$$\left[K^{(r=1)}\right] = \begin{array}{c} \\ \mathrm{B} \\ \mathrm{B} \\ \mathrm{B} \\ \mathrm{B} \\ \mathrm{I} \\ \mathrm{I} \\ \mathrm{I} \\ \mathrm{I} \end{array} \begin{array}{c} \begin{array}{cccccccc} \mathrm{B} & \mathrm{B} & \mathrm{B} & \mathrm{B} & \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{I} \end{array} \\ \left[\begin{array}{cccc|cccc} K_{11} & K_{12} & K_{13} & K_{14} & K_{15} & K_{16} & K_{17} & K_{18} \\ K_{21} & K_{22} & K_{23} & K_{24} & K_{25} & K_{26} & K_{27} & K_{28} \\ K_{31} & K_{32} & K_{33} & K_{34} & K_{35} & K_{36} & K_{37} & K_{38} \\ K_{41} & K_{42} & K_{43} & K_{44} & K_{45} & K_{46} & K_{47} & K_{48} \\ \hline K_{51} & K_{52} & K_{53} & K_{54} & K_{55} & K_{56} & K_{57} & K_{58} \\ K_{61} & K_{62} & K_{63} & K_{64} & K_{65} & K_{66} & K_{67} & K_{68} \\ K_{71} & K_{72} & K_{73} & K_{74} & K_{75} & K_{76} & K_{77} & K_{78} \\ K_{81} & K_{82} & K_{83} & K_{84} & K_{85} & K_{86} & K_{87} & K_{88} \end{array}\right] \end{array} = \begin{bmatrix} K_{BB} & K_{BI} \\ K_{IB} & K_{II} \end{bmatrix}$$

(6.56)

Since the Dirichlet boundary conditions are imposed on the 7th & 8th degree-of-freedom, Eq. (6.56) becomes:

$$\left[K_{b.c.}^{(r=1)}\right] = \begin{array}{c} \\ \mathrm{B} \\ \mathrm{B} \\ \mathrm{B} \\ \mathrm{B} \\ \mathrm{I} \\ \mathrm{I} \\ \mathrm{I} \\ \mathrm{I} \end{array} \begin{array}{c} \begin{array}{cccccccc} \mathrm{B} & \mathrm{B} & \mathrm{B} & \mathrm{B} & \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{I} \end{array} \\ \left[\begin{array}{cccc|cccc} \times & \times & \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & \times & \times & 0 & 0 \\ \hline \times & \times & \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & \times & \times & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \end{array} = \begin{bmatrix} K_{BB} & K_{BI} \\ K_{IB} & K_{II} \end{bmatrix}$$

(6.57)

Thus

$$\left[K_{BI_{b.c.}}^{(r=1)}\right] = \begin{array}{c} \\ \mathrm{B} \\ \mathrm{B} \\ \mathrm{B} \\ \mathrm{B} \end{array} \begin{array}{c} \begin{array}{cccc} \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{I} \end{array} \\ \left[\begin{array}{cccc} \times & \times & 0 & 0 \\ \times & \times & 0 & 0 \\ \times & \times & 0 & 0 \\ \times & \times & 0 & 0 \end{array}\right] \end{array}$$

(6.58)

It should be noted that Eq. (6.28) has the same form (last 2 columns) as Eq. (6.58)

6.4 How to construct Sparse Assembly of "Rectangular" Matrix $\left[K_{BI}^{(r)}\right]$

The numerical procedures for finite element sparse assembly of the rectangular matrix $\left[K_{BI}^{(r)}\right]$ can be summarized in the following 2-step procedures:

Step 1: For each rth substructure, identify how many and which finite elements will have contributions to both Boundary & Interior nodes (assuming there are 2 dof per node, and 3 nodes per element). Also, assuming there are 6 boundary nodes, and 8 interior nodes in the rth substructure.
At the end of step 1, say elements 5, 10 and 11 are recorded. Also, assuming the element nodes connectivity are:

El. #	node-i	node-j	node-k
5	6 = B	1 = B	8 = I
10	5 = B	6 = B	9 = I
11	1 = B	13 = I	11 = I

Note: B ≡ Boundary node, if ≤ node 6. I ≡ Interior, if > node 6.

Step 2: Based on the information provided by step 1, one obtains:

K_{BI}=

	I.Node1 (=7-6)	I.Node2 (=8-6)	3	4	5	6	7 (=13-6)	8 (=14-6)
B.Node1		1,2 7,8			5,6 11,12		3,4 9,10	
B.Node2								
3								
4								
5								
6								

The above table, obtained for K_{BI} , can be generated by the following "pseudo" FORTRAN algotihms :

icount = 0
Do loop
  J = Boundary Node 1
    El. 5 = [6, 1, 8] ➔ B. node has coupling with I. node 2 (= 8-6) ➔ icount = 1
    El. 10 = [5, 6, 9] ➔ B. node 1 has no coupling with element 10.
    El. 11 = [1, 13 11] ➔ B. node 1 has coupling
                with I. node 7 (= 13-6) ➔ icount = icount+1 = 2
                with I. node 5 (=11-6). Hence, icount = 3
    so: $IE_{bi}(1) = 1$ (6.59)
    $IE_{bi}(2) = IE_{bi}(1)+(\text{icount} = 3)*(\text{ndofpn}=2)^2 = 13$ (6.60)

$$JE_{bi} = \{\text{columns } 3, 4, 13, 14, 9, 10, 3, 4, 13, 14, 9, 10\ \} \quad (6.61)$$

J = Boundary Node 2
El. 5 = [6, 1, 8] ➔ Boundary node 2 has no coupling with El. 5
El. 10 = [5, 6, 9] ➔ Boundary node 2 has no coupling with El. 10
El. 11 = [1, 13 11] ➔ Boundary node 2 has no coupling with El. 11

Enddo

Remarks:

In Eq. (6.61), column numbers 3, 4, 13, 14, 9, 10 are repeated again, the reason is because each node is assumed to have 2 dof.

6.5 Mixed Direct-Iterative Solvers for Domain Decomposition

Using the domain decomposition (D.D.) formulation, one first needs to solve the unknown boundary displacement vector $\overrightarrow{z_B}$ from Eq. (6.16). However, the assemble process for obtaining the effective boundary stiffness matrix $\overline{K_B}$ (see Eq. (6.14)) will require the computation of the triple matrix products of $K_{BI}^{(r)}\left[K_{II}^{(r)}\right]^{-1}K_{IB}^{(r)}$ (for each r^{th} substructure), as indicated in Eq. (6.12). For large-scale-applications, Eq. (6.12) is both computational and memory intensive, since the related system of linear equations has a lot of right-hand-side vectors. Thus, forward and backward solution phases need be done repeatedly. Furthermore, although each individual matrices $\left[K_{BI}^{(r)}\right]$, $\left[K_{II}^{(r)}\right]$ and $\left[K_{IB}^{(r)}\right]$ can be sparse, the triple products of $K_{BI}^{(r)}\left[K_{II}^{(r)}\right]^{-1}K_{IB}^{(r)}$ is usually dense, and therefore, a lot of computer memory is required. For the above reasons, mixed direct-iterative solver is suggested for solving Eq. (6.16).

Preconditioning Matrix

Consider the linear system $[A]\vec{x} = \vec{b}$, where the matrix [A] is assumed to be symmetric positive definite (SPD). If the solution vector $\vec{x}$ is sought by an iterative solver, then one normally prefers to improve the condition number of the coefficient matrix ($=\frac{\lambda_{max}}{\lambda_{min}}$ = ratio of largest over smallest eigen-values of the coefficient matrix) by the "preconditioning process" as described in the following paragraphs :

Option 1 (Symmetrical property is preserved)

Let P be a preconditioning matrix, which is assumed to be non-singular. Then PAP^T is SPD. Instead of solving $[A]\vec{x} = \vec{b}$, one solves :

$$[P\,A\,\underbrace{P^T]*\{P^{-T}}_{\text{Identity Matrix}}\,\vec{x}\} = [P]\vec{b}$$

or

$$[A^*]*\{\vec{y}\} = \{\vec{b^*}\}$$

where :

$$[A^*] \equiv [P\,A\,P^T] = \text{symmetrical matrix}$$

$$\vec{y} \equiv \left\{P^{-T}\,\vec{x}\right\} \quad ; \text{hence} \quad \vec{x} = [P^T]\vec{y}$$

$$\vec{b^*} \equiv [P]\vec{b}$$

The preconditioner [P] should be selected so that :

1. Matrix $[A^*]$ will have better condition number than its original matrix [A].
2 Matrix [P] should be "cheap" to factorize.

As an example, the Cholesky factorization of [A] can be obtained as :

$$[A] = [U]^T\,[U]$$

where [U] is an upper triangular, factorized matrix.

Suppose one selects $[P] = U^{-T}$, then :

$$A^* \equiv P[A]P^T = \underbrace{U^{-T}[U^T}\,\underbrace{U]U^{-1}} = [I]$$

The condition number of $[A^*] = \dfrac{\lambda_{max}}{\lambda_{min}} = \dfrac{1.0}{1.0} = 1$

Therefore, the Conjugate Gradient method, when applied to $[A^* = I]\vec{y} = \vec{b^*}$ will converge in 1 iteration. However, in this case it is "too expensive" to factorize [P] !
The compromised strategies will be :

$$[P] = [U_a]^{-T}$$

where $[U_a] \equiv$ inexpensive approximation of [U], and the amount of fill-in terms occurred in $[U_a]$ can be controlled (or specified) by the user. Various strategies for "incomplete Cholesky factorization" have been suggested to obtain $[U_a]$ for preconditioning purposes. The original CG, and its PCG algorithms are summarized in the following section :

CG Without Preconditioner	**CG With Preconditioner**
Solving $[A]\vec{x} = \vec{b}$	Solving $[A^*]\vec{y} = \vec{b^*}$
Given an initial guess $\vec{x}^{(0)}$	Given $\vec{x}^{(0)}$
	Compute incomplete factor $[P]$
Compute initial residual $\vec{r}^{(0)} = \vec{b} - A\,x^{(0)}$	Compute $\vec{r}^{(0)} = \{Pb\} - \left[P\,A\,P^T\right]\vec{x}^{(0)}$
Set $\vec{d}^{(0)} = 0 \;;\; \rho_{-1} = 1$	Set $\vec{d}^{(0)} = 0 \;;\; \rho_{-1} = 1$
DO i=1, 2, ...	DO i=1, 2, ...
$\rho_{i-1} = \left\{r^{(i-1)}\right\}^T \left\{r^{(i-1)}\right\}$	$\rho_{i-1} = \left\{r^{(i-1)}\right\}^T \left\{r^{(i-1)}\right\}$
$\beta_{i-1} = \rho_{i-1} \Big/ \rho_{i-2}$	$\beta_{i-1} = \rho_{i-1} \Big/ \rho_{i-2}$
$d^{(i)} = r^{(i-1)} + \beta_{i-1}\, d^{(i-1)}$	$d^{(i)} = r^{(i-1)} + \beta_{i-1}\, d^{(i-1)}$
$q^{(i)} = [A]\,d^{(i)}$	$q^{(i)} = [P\,A\,P^T]\,d^{(i)}$
$\alpha_i = \rho_{i-1} \Big/ \left\{d^{(i)}\right\}^T \left\{q^{(i)}\right\}$	$\alpha_i = \rho_{i-1} \Big/ \left\{d^{(i)}\right\}^T \left\{q^{(i)}\right\}$
$x^{(i)} = x^{(i-1)} + \alpha_i\, d^{(i)}$	$x^{(i)} = x^{(i-1)} + \alpha_i\, d^{(i)}$
$r^{(i)} = r^{(i-1)} - \alpha_i\, q^{(i)}$	$r^{(i)} = r^{(i-1)} - \alpha_i\, q^{(i)}$
Converge ??	Converge ??
END DO	END DO
	If converged, then set $\vec{x} = [P]^T\,\vec{x}$

Option 2 (Symmetrical property may be destroyed)

Instead of solving $[A]\vec{x} = \vec{b}$, one solves :

$$\underbrace{[P]^{-1}[A]}\vec{x} = \underbrace{[P]^{-1}\,\vec{b}}$$

or

$$[A^*]\vec{x} = \left\{\vec{b^*}\right\}$$

where :

$$[A^*] \equiv [P]^{-1}[A] = \text{may NOT be symmetrical}$$

$$\left\{\overrightarrow{b^*}\right\} \equiv [P]^{-1}\,\vec{b}$$

This formulation is simpler than the one discussed in Option 1. However, $[A^*]$ may NOT be a symmetrical matrix.

Remarks

1. If the original matrix [A] is unsymmetrical, then this Option 2 may be a preferable choice.

2. If one selects $[P] = [A]$, then $[A^*] = [I]$ and the iterative solver will converge in 1 iteration.

In Table 6.1, Preconditioned Conjugate Gradient (PCG) algorithm for solving system of symmetrical linear equations $[A]\vec{x} = \vec{b}$, with the preconditioned matrix [B] is summarized.

Table 6.1: Preconditioned Conjugate Gradient Algorithm For Solving $[A]\vec{x} = \vec{b}$

Step 1: Initialized $\overrightarrow{x_o} = \vec{0}$

Step 2: Residual vector $\overrightarrow{r_o} = \vec{b}$ (or $\overrightarrow{r_o} = \vec{b} - A\overrightarrow{x_0}$, for "any" initial guess $\overrightarrow{x_o}$)

Step 3: "Inexpensive" preconditioned $\overrightarrow{z_0} = [B]^{-1} \cdot \overrightarrow{r_0}$

Step 4: Search direction $\overrightarrow{d_0} = \overrightarrow{z_0}$

For i = 0, 1, 2, …, maxiter

Step 5: $\alpha_i = \dfrac{r_i^T z_i}{d_i^T \{A \cdot d_i\}}$

Step 6: $x_{i+1} = x_i + \alpha_i d_i$

Step 7: $r_{i+1} = r_i - \alpha_i [A d_i]$

Step 8: Convergence check: if $\|r_{i+1}\| < \|r_0\| \cdot \varepsilon$ ➔ stop

Step 9: $z_{i+1} = B^{-1} r_{i+1}$

Step 10: $\beta_i = \dfrac{r_{i+1}^T z_{i+1}}{r_i^T z_i}$

Step 11: $d_{i+1} = z_{i+1} + \beta_i d_i$

End for

6.6 Preconditioned Matrix For PCG Algorithm with DD Formulation

The difficulty in constructing an efficient preconditioned matrix [B] in conjunction with PCG algorithm with D.D. formulation is compounded by the fact the coefficient matrix $\left[\overline{K_B}\right]=\sum_{r=1}^{NSU}\overline{K_B}^{(r)}=\sum_{r=1}^{NSU}\left(K_{BB}^{(r)}-K_{BI}^{(r)}\left[K_{II}^{(r)}\right]^{-1}K_{IB}^{(r)}\right)$ has not been assembled explicitly. In other words, how can we construct a preconditioned matrix [B] when the original coefficient matrix $\left[\overline{K_B}\right]$ has not even been formed? Even the most simple "diagonal preconditioned" scheme, one still has to introduce "some approximation" about $\left[\overline{K_B}\right]$. The following 2 options are possible for considerations :

Option 1: let $[B]\approx\left[\overline{K_B}\right]\approx\sum_{r=1}^{NSU}K_{BB,Diag}^{(r)}$

Option 2: For the preconditioned purpose only, approximate:

$$\left[K_{II}^{(r)}\right]\approx diagonal\ of\left[K_{II}^{(r)}\right]$$

Hence $\left[K_{II}^{(r)}\right]_{Approx}^{-1}$ is inexpensive, and

$$[B]\approx\left[\overline{K_B}\right]\approx\sum_{r=1}^{NSU}\left(K_{BB,Diag}^{(r)}-K_{BI}^{(r)}\left[K_{II}^{(r)}\right]_{Approx}^{-1}K_{IB}^{(r)}\right)$$

In Table 6.2, the corresponding 11-step procedure for Pre-conditioned Conjugate Gradient Algorithm within the context of Domain Decomposition (DD) formulation is summarized.

Table 6.2: Pre-conditioned Conjugate Gradient D.D. Algorithm For solving

$$\left[\overline{K_B}\right]\overrightarrow{z_B}=\overrightarrow{f_B}$$

Initialized Phase

Step 1: $\overrightarrow{z_{B_i}}=\vec{0}$

Step2: Residual vector $\vec{r_i}=\overrightarrow{f_B}-\overrightarrow{K_B}\ \overrightarrow{z_{B_i}}=\sum_{r=1}^{NSU}\overrightarrow{f_B}^{(r)}$ or

$$\vec{r_i}=\sum_{r=1}^{NSU}\left(f_B^{(r)}-K_{BI}^{(r)}\left[K_{II}^{(r)}\right]^{-1}f_I^{(r)}\right)$$

DO 2 r=1,NSU (in parallel computation)

Step 2.1: Direct sparse solver F/B solution with 1 RHS interior load vector

$$\vec{T}_1 = \left[K_{II}^{(r)} \right]^{-1} \cdot f_I^{(r)}$$

Step 2.2: Sparse matrix times vector

$$\vec{T}_2 = K_{BI}^{(r)} \cdot \vec{T}_1$$

Step 2.3: Compute sub-domain residual

$$\vec{T}_1 = f_B^{(r)} - \vec{T}_2$$

Step 2.4: Insert sub-domain residual into proper location of $\vec{r}_i$

The partial resulted vector $\vec{T}_1$ from each processor will be sent to the Master processor, together with the mapping information about local-global boundary dofs.

Receive & Copy $\vec{r}_i = \vec{T}_1^{(r)}$, by the Master Processor.

2 Continue

Step 3: "Inexpensive" preconditioning, by the Master Processor.

$$\vec{z}_i = [B]^{-1} \cdot \vec{r}_i$$

Step 4: Search direction, by the Master Processor.

$$\vec{d}_i = \vec{z}_i$$

Iteration loop begins (for i = 0, 1, 2, …, maxiter)

Step 5: Compute scalar $\alpha_i = \dfrac{r_i^T z_i}{d_i^T \left\{ \left[\overline{K_B} \right] \cdot d_i \right\}}$

Step 5.1: $up = r_i^T \cdot z_i$, by the Master Processor

Master Processor broadcasts $\vec{d}_i$ to all other processors

Step 5.2: compute

$$\overline{K_B} \cdot \vec{d}_i = \left(\sum_{r=1}^{NSU} \overline{K_B}^{(r)} \right) \cdot \vec{d}_i = \sum_{r=1}^{NSU} \left(K_{BB}^{(r)} - K_{BI}^{(r)} \left[K_{II}^{(r)} \right]^{-1} K_{IB}^{(r)} \right) \cdot \vec{d}_i$$

DO 5 r=1,NSU (in parallel computation)

Step 5.2a: Sparse matrix times vector

$$\vec{T}_1 = K_{IB}^{(r)} \cdot \vec{d}_i$$

Step 5.2b: Direct sparse solver F/B with 1 RHS vector

$$\vec{T}_2 = \left[K_{II}^{(r)} \right]^{-1} \cdot \vec{T}_1$$

Step 5.2c: Sparse matrix times vector

$$\vec{T}_1 = K_{BI}^{(r)} \cdot \vec{T}_2$$

Step 5.2d: Sparse matrix times vector

$$\vec{T}_2 = K_{BB}^{(r)} \cdot \vec{d}_i$$

Step 5.2e:

$$\vec{T}_2 = \vec{T}_2 - \vec{T}_1$$

Step 5.2f: Put vector $\vec{T}_2$ into proper location of

$$\left[\overline{K_B}\right] \cdot d_i = \overrightarrow{stored}$$

Each processor will send its own $\vec{T}_2$ to the Master Processor.

5 Continue

The following steps will be done by the Master Processor.

Step 5.2g: Received vectors $\vec{T}_2$ (from each processor) and assembled vector $\overrightarrow{stored}$. Then, compute:

$$down = \vec{d}_i \cdot \overrightarrow{stored}$$

$$\alpha_i = \frac{up}{down}$$

Step 6: Compute new, improved solution

$$\overrightarrow{z_{B_{i+1}}} = \overrightarrow{z_{B_i}} + \alpha_i \vec{d}_i$$

Step 7: Compute new residual vector

$$\overrightarrow{r_{i+1}} = \vec{r}_i - \alpha_i \cdot \overrightarrow{stored}$$

Step 8: Convergence check:

$$r_{0,norm} = \left\| \vec{r}_0 \right\|$$

$$r_{i+1,norm} = \left\| \overrightarrow{r_{i+1}} \right\|$$

Iteration steps will stop when $r_{i+1,norm} < \varepsilon \cdot r_{0,norm}$. Where ε is user input parameter

Step 9: "Inexpensive" preconditioning

$$\overrightarrow{z_{i+1}} = [B]^{-1} \cdot \overrightarrow{r_{i+1}}$$

Step 10: Compute $\beta_i = \dfrac{r_{i+1}^T z_{i+1}}{r_i^T z_i}$

$$up = r_{i+1}^T z_{i+1}$$

$$down = r_i^T z_i$$

$$\beta_i = \frac{up}{down}$$

Step 11: New search direction

$$\overrightarrow{d_{i+1}} = \overrightarrow{z_{i+1}} + \beta_i \overrightarrow{d_i}$$

Based upon the "primal" DD formulation, discussed in Sections 6.1-6.6, the MPI/Fortran software package DIPSS (Direct Iterative Parallel Sparse Solver) has been developed to solve few large scale, practical engineering problems which are summarized in the following paragraphs:

Example 1 – Three dimensional acoustic finite element model. In this example, DIPSS is exercised to study the propagation of plane acoustic pressure waves in a 3-D hard wall duct without end reflection and airflow.

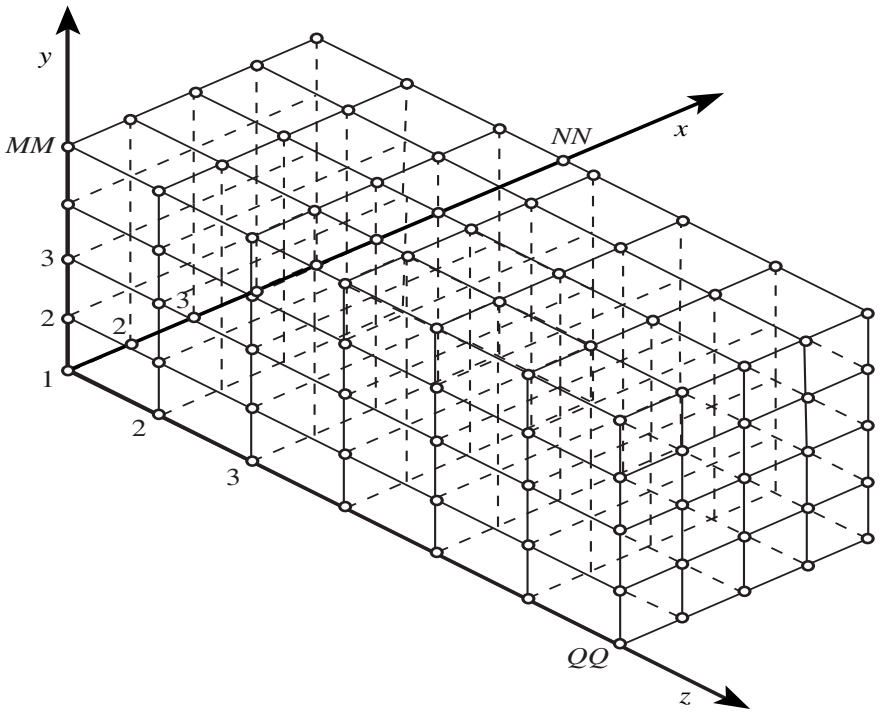


Figure 6.6: Finite element model for a three-dimensional hard wall duct

The duct is shown in Figure 6.6 and is modeled with brick elements. The source and exit planes are located at the left and right boundary, respectively. The matrix, K, contains complex coefficients and the dimension of K is determined by the product of NN, MM, and QQ (N=MMxNNxQQ). Results are presented for two grids (N=751,513 and N=1,004,400) and the finite element analysis procedure for generation the complex stiffness matrix, K, was presented in Reference [6.2].

DIPSS [Ref. 6.3] memory and wallclock statistics were also compared to those obtained using the platform specific SGI parallel sparse solver (i.e., ZPSLDLT). These statistics were computed on an SGI ORIGIN 2000 computer platform that was located at the NASA Langley Research Center. The SGI platform contained 10 gigabytes of memory and eight ORIGIN 2000 processors were used. It should be noted that the ZPSLDLT is part of the SCSL library (version 1.4 or higher) and is considered to be one of the most efficient commercialized direct sparse solvers that is capable of performing complex arithmetic. Due to the 3-D nature of hard wall duct example problem, K encounters lots of fill-in during the factorization phase. Thus, only the small grid (N=751,513) could fit within the allocated memory on the ORIGIN~2000. ZPSLDLT required 6.5 wallclock hours to obtain the solution on the small grid whereas DIPSS wallclock was only 2.44 hours. DIPSS also required nearly 1 gigabyte less memory than ZPSLDLT, and the DIPSS and ZPSLDLT solution vector were in excellent agreement.

Because DIPSS uses MPI for interprocess communications, it can be ported to other computer platforms. To illustrate this point the DIPSS software was ported to the SUN 10000 platform at Old Dominion University and used to solve the large grid duct acoustic problem (N=1,004,400). Wallclock statistics and speedup factors were obtained using as many as 64~SUN~10000 processors. Results are presented in Table 6.3. It should be noted that a superlinear speedup factor of 85.95 has been achieved when 64 SUN 10000 processors are used. This super-linear speedup factor is due to two primary reasons:

1. The large finite element model has been divided into 64 sub-domains. Since each processor is assigned to each smaller subdomain, the number of operations performed by each processor has been greatly reduced. Note that the number of operations are proportional to $\left(n^{(r)}\right)^3$ for the dense matrix, or $n^{(r)} \cdot BW^2$ for the banded, sparse matrix, where BW represent the half Band Width of the coefficient stiffness matrix.

2. When the entire finite element model is analyzed by a direct, conventional sparse solver, more computer "paging" is required due to a larger problem size.

Table 6.3 : Performance of DIPSS software for 3-D hard wall duct (N=1,004,400 complex equations)

# Processor (SUN 10000@ODU)	1	2	4	8	16	32	64
Sparse Assembly Time (seconds)	19.38	10.00	5.08	2.49	1.26	0.70	0.27
Sparse Factorization (seconds)	131,229	58,976	26,174	10,273	3,260	909	56
Total time (entire FEA)	131,846	61,744	27,897	11,751	3,817	1,967	1,534
Total Speed-Up Factor	1.00	2.14	4.73	11.22	34.54	67.03	85.95

Examples 2 – Three dimensional structural bracket finite element model. The DD formulation has also been applied to solver the 3-D structural bracket problem shown in Figure 6.7. The finite element model contains 194,925 degrees of freedom (N=194,925) and the elements in the matrix, K, are real numbers. Results were computed on a cluster of 1-6 personal computers (PCs) running under Windows environments with Intel Pentium. It should be noted that the DIPSS software was not ported to the PC cluster, but the DD formulation was programmed (from scratch, in C^{++}) on the PC cluster processors [6.3].

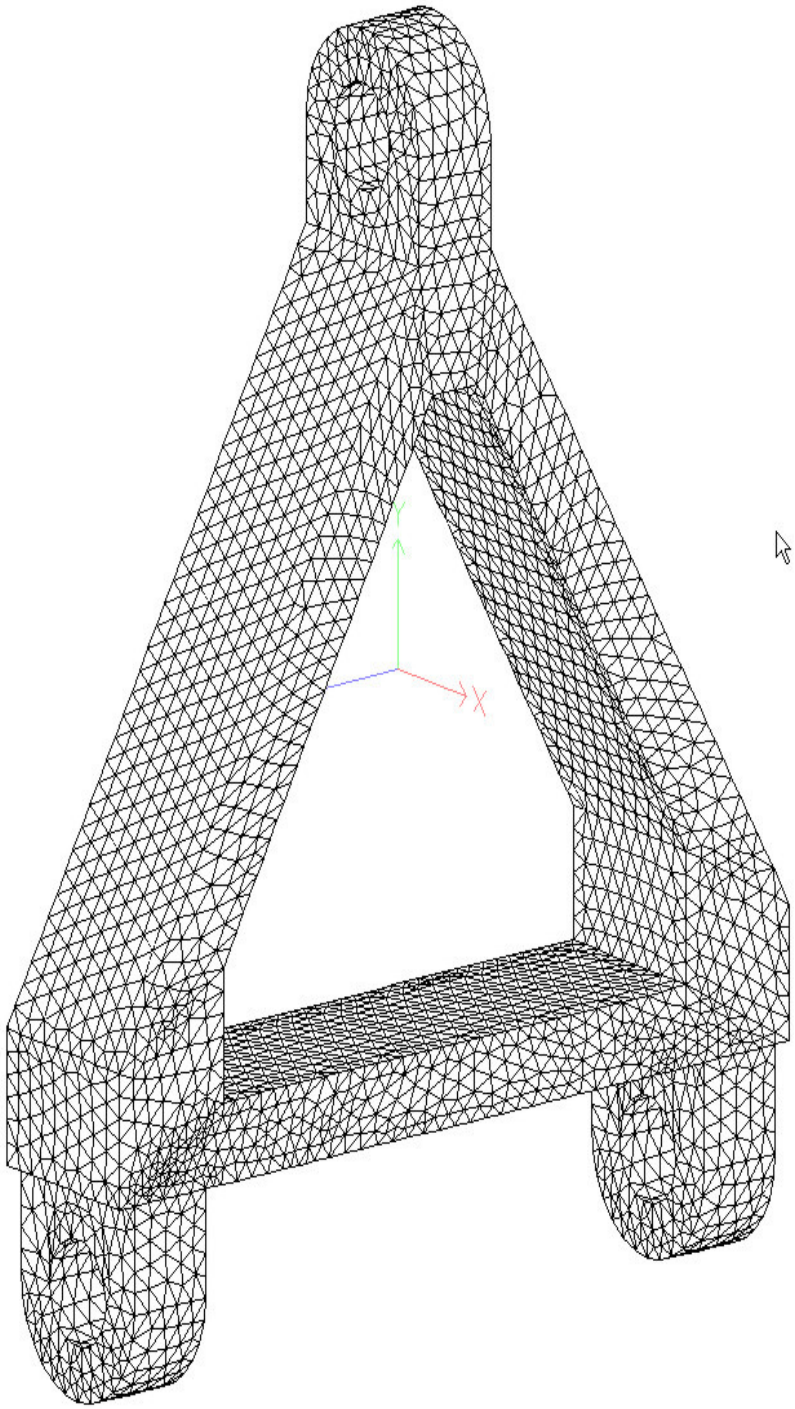


Figure 6.7: Finite element model for a three-dimensional structural bracket

The wallclock time (in seconds) to solve this example is documented in Table 6.4. A superlinear speedup factor of 10.35 has been achieved when 6 processors were used.

Table 6.4: 3-D Structural bracket model (194,925 dofs, K= real numbers)

# Processor (Intel PC @ ASU)	1	2	3	4	5	6
Total Walll Clock Time (seconds)	2,670	700	435	405	306	258
Total Speed-Up Factor (seconds)	1.00	3.81	6.14	6.59	8.73	10.35

6.7 Generalized Inverse

First, let us consider some key concepts of the generalized inverse. Given a matrix $A \in R^{m\times n}$, $A^{+} \in R^{n\times m}$ is called the generalized inverse of A if

$$A_{m\times n} A^{+}_{n\times m} A_{m\times n} = A_{m\times n} \tag{6.62}$$

Now, given the system of linear equations

$$A_{m\times n}\, \vec{x}_{n\times 1} = \vec{b}_{m\times 1} \tag{6.63}$$

with $\vec{b} \in$ Range of A ($\vec{b}$ is a linear combinations of independent columns of A), the solution(s) of Eq. (6.63) can be given in the form

$$\vec{x}_{n\times 1} = A^{+}_{n\times m}\vec{b}_{m\times 1} + \left(I_{n\times n} - A^{+}_{n\times m}A_{m\times n}\right)\vec{y}_{n\times 1} \tag{6.64}$$

for $\vec{y} \in R^{n}$

Proof

To prove that Eq. (6.64) is the solution of Eq. (6.63), one starts with pre-multiplying both sides of Eq. (6.64) with A, thus:

$$A\vec{x} = AA^{+}\vec{b} + A\left(I - A^{+}A\right)\vec{y} \tag{6.65}$$

From the definition given by Eq. (6.62), one has

$$0 = A - AA^{+}A = A\left(I - A^{+}A\right) = 0 \tag{6.66}$$

Utilizing Eq. (6.66), Eq. (6.65) becomes:

$$A\vec{x} = AA^{+}b \tag{6.67}$$

Also, pre-multiplying both sides of Eq. (6.63) by AA^{+}, one obtains:

$$AA^{+}\left(A\vec{x}\right) = AA^{+}b \tag{6.68}$$