Shape function

Standard
hierarchical] element shape function

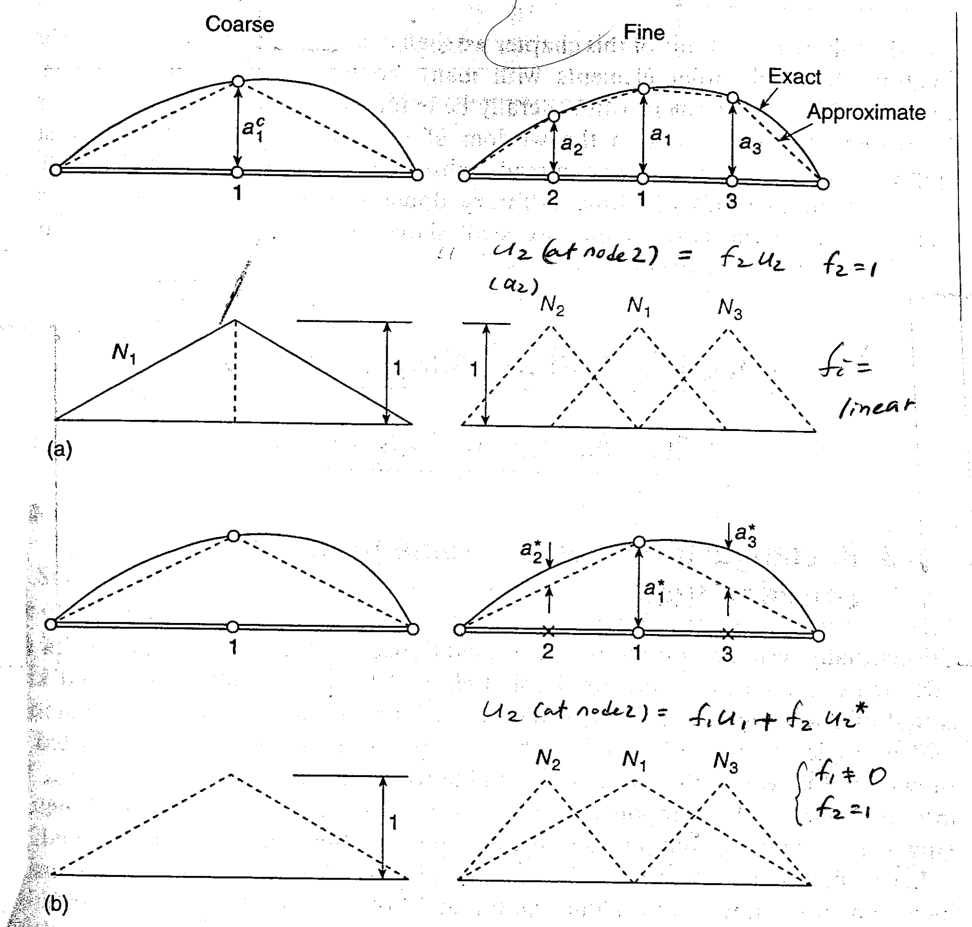


Fig. 8.1 A one-dimensional problem of stretching of a uniform elastic bar by prescribed body forces. (a) 'Standard approximation. (b) Hierarchic approximation.

hierarchical shape function

The shape function does not depend on the number of nodes in the mesh.

With hierarchic forms, it is convenient to consider the finer mesh as still using the same, coarse, elements but now adding additional refining functions.

Standard Shape function

Lagrange family:

$$l_k^n(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}$$

numerator

denominator

Lagrange polynomials

at $x = x_k$ $\quad l_k^n(x) = 1$

at other points $\quad l_k^n(x) = 0$

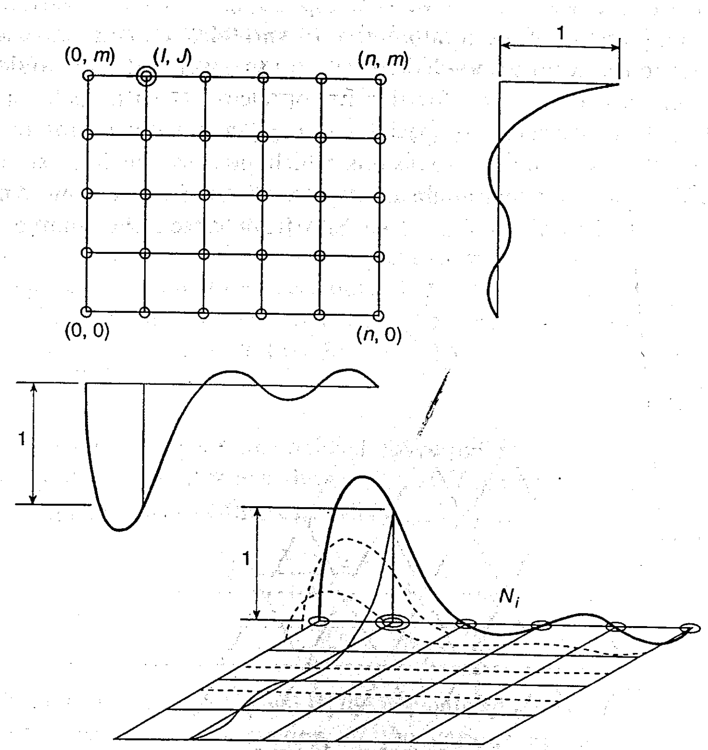


Fig. 8.6 A typical shape function for a Lagrangian element ($n = 5$, $m = 4$, $I = 1$, $J = 4$).

In two dimensions

$$f_{ij} = l_i^n(x)\, l_j^m(y)$$

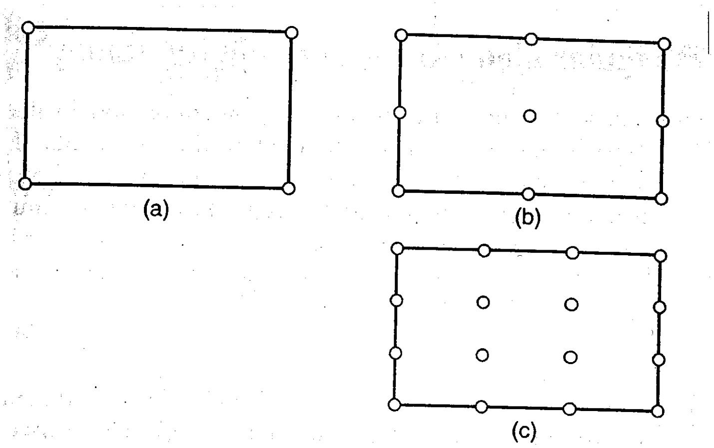


Fig. 8.7 Three elements of the Lagrange family: (a) linear, (b) quadratic, and (c) cubic.

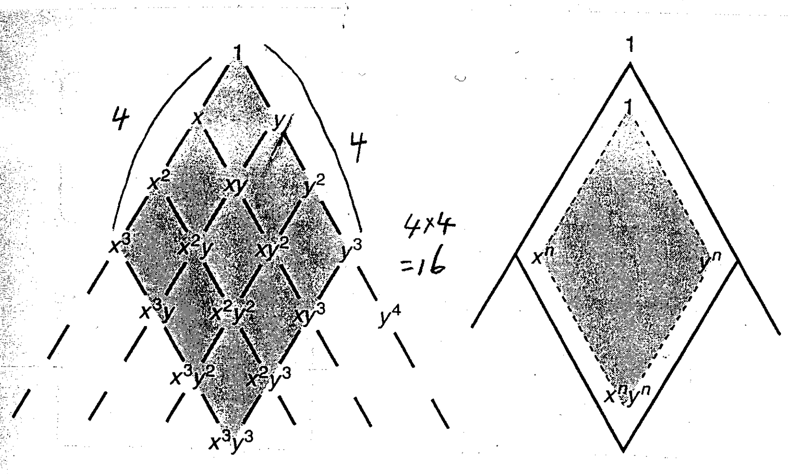


Fig. 8.8 Terms generated by a lagrangian expansion of order 3 × 3 (or $n \times n$). Complete polynomials of order 3 (or n).

lagrange polynomial shape functions → complete polynomials of order n

if $n+1$ nodes are selected in both x and y direction.

Serendipity family

It is usually more efficient to make the functions dependent on nodal values placed on the element boundary

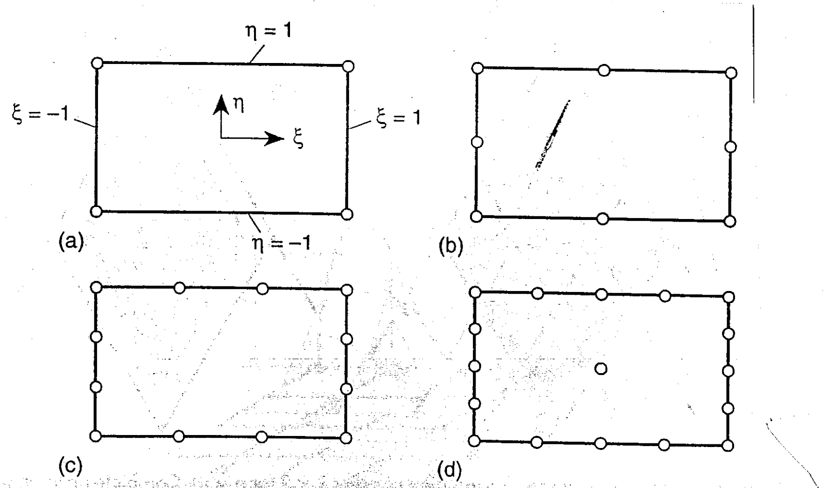
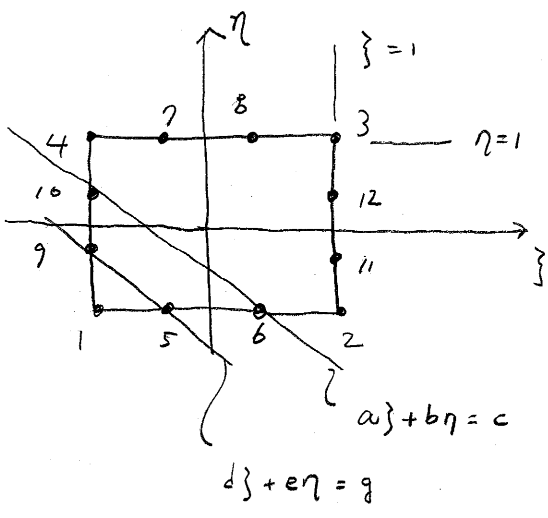


Fig. 8.9 Rectangles of boundary node (serendipity) family: (a) linear, (b) quadratic, (c) cubic, (d) quartic.

The shape functions were originally derived by inspection, and progression to yet higher members is difficult and requires some ingenuity. It was therefore named as Serendipity after the famous princes of Serendip noted for their chance of discoveries

However, a quite systematic way of generating the serendipity shape function can be devised.



Question)

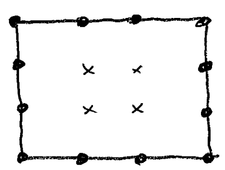
obtain shape function f_1

$$f_1 = A(1-\xi)(1-\eta)(a\xi + b\eta - c)(d\xi + e\eta - g)$$

Roles of adding nodes

1) Increases dof's. Thus higher order displacement function can be used.

2) Connectivity between elements is increased. Thus, the force-equilibrium and displ. - compatibility between elements are enhanced.

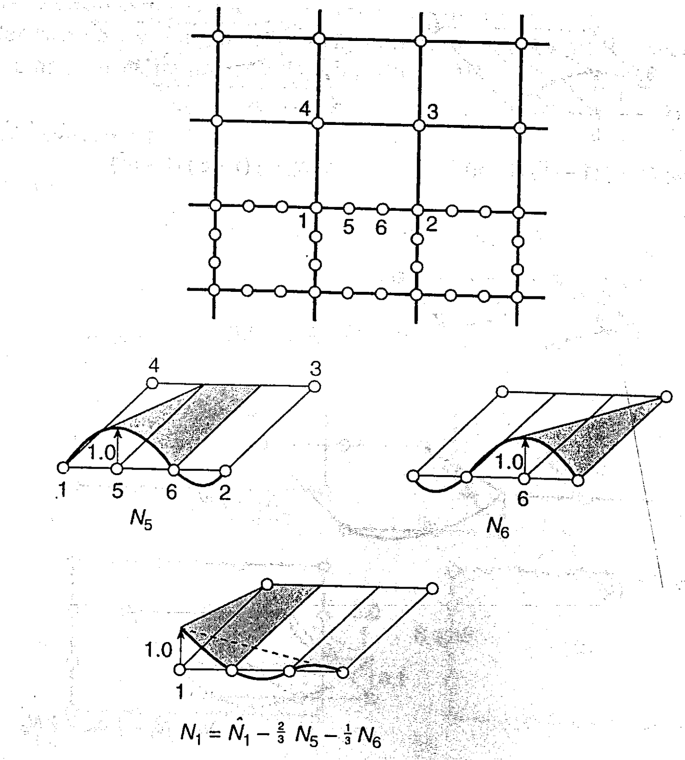


Exterior nodes provide both 1) and 2)

interior nodes provide only 1)

Thus, the importance of the interior nodes is less than that of the exterior nodes.

This means that increasing exterior nodes is better than increasing interior nodes.



$N_1 = \hat{N}_1 - \frac{2}{3} N_5 - \frac{1}{3} N_6$

Fig. 8.11 Shape functions for a transition 'serendipity' element, cubic/linear.

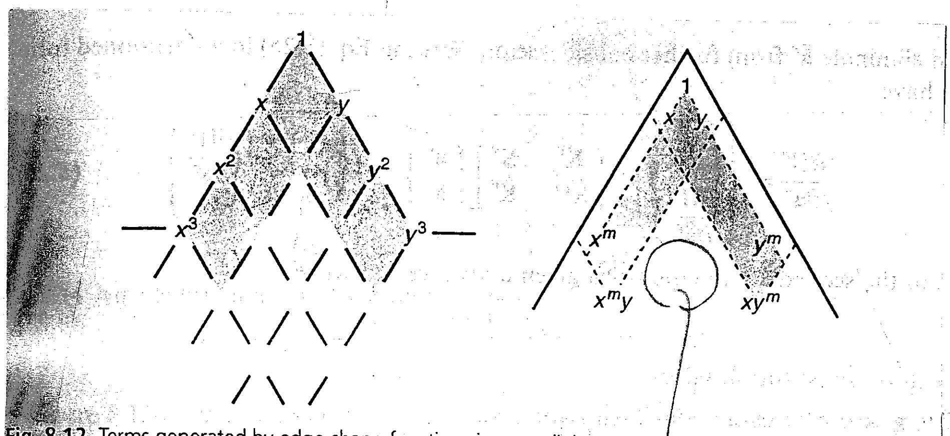
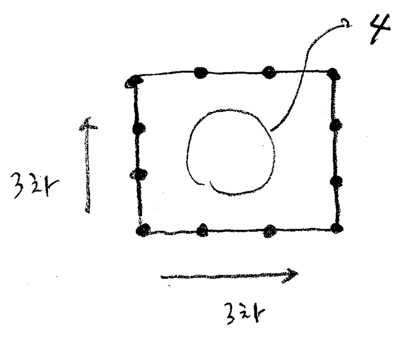


Fig. 8.12 Terms generated by edge shape functions in serendipity-type elements (3 × 3 and $m \times m$).

Lagrange polynomial로 보면 부족하는 node 갯수 만큼 생략



1

$x \quad y$

$x^2 \quad xy \quad y^2$

$x^3 \quad x^2y \quad xy^2 \quad y^3$

$x^3y \quad x^2y^2 \quad xy^3$

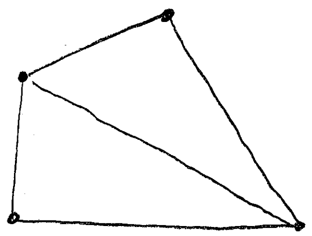
$x^3y^2 \quad x^2y^3$

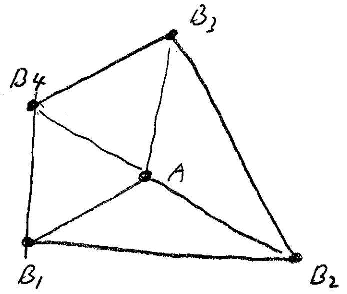
x^3y^3

$\rightarrow 4 \Rightarrow$ removed

Quadrilateral Element

A variety of the geometry of the quadrilateral can be made with the combination of triangular elements.



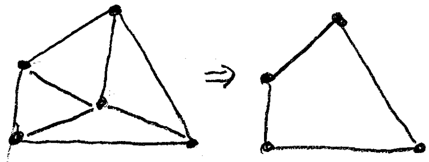


A : internal node

B : external node

By static condensation, the stiffness matrix can be condensed into that only with external dof's.

$$\begin{bmatrix} K_{AA} & K_{AB} \\ K_{BA} & K_{BB} \end{bmatrix} \begin{bmatrix} \delta_A \\ \delta_B \end{bmatrix} = \begin{bmatrix} P_A \\ P_B \end{bmatrix} \quad \begin{matrix} — ① \\ — ② \end{matrix}$$



$$① \Rightarrow \quad K_{AA}\,\delta_A + K_{AB}\,\delta_B = P_A$$

$$\delta_A = K_{AA}^{-1}\,(P_A - K_{AB}\,\delta_B)$$

$$② \Rightarrow \quad K_{BA}\,\delta_A + K_{BB}\,\delta_B = P_B$$

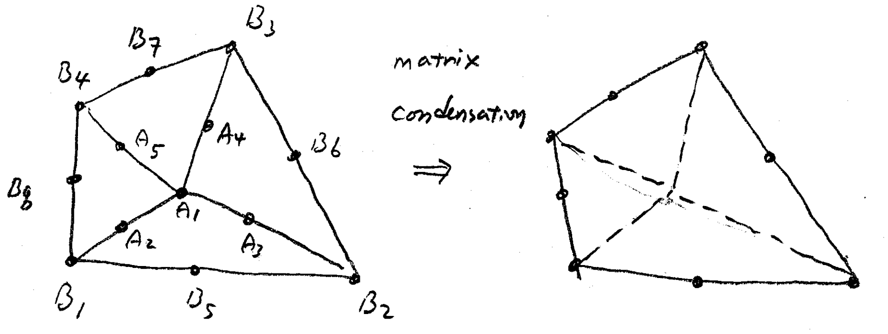
$$K_{BA}\,K_{AA}^{-1}\,(P_A - K_{AB}\,\delta_B) + K_{BB}\,\delta_B = P_B$$

$$\underbrace{(K_{BB} - K_{BA}\,K_{AA}^{-1}\,K_{AB})}_{K_B^*}\,\delta_B = \underbrace{P_B - K_{BA}\,K_{AA}^{-1}\,P_A}_{P_B^*}$$

$$\begin{bmatrix} K_{AA} & K_{AB} \\ 0 & K_{BB}^* \end{bmatrix} \begin{bmatrix} \delta_A \\ \delta_B \end{bmatrix} = \begin{bmatrix} P_A \\ P_B^* \end{bmatrix}$$

This process can be performed by Gauss Elimination.

$$\begin{cases} K_B^* = \text{modified stiffness referring to the external dofs} \\ P_B^* = \text{modified nodal loads} \end{cases}$$



26×26 $\qquad\qquad$ 16×16

$$\left\{ \begin{aligned} a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 &= a_5 \\ b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4 &= b_5 \\ c_3 x_3 + c_4 x_4 &= c_5 \\ d_3 x_3 + d_4 x_4 &= d_5 \end{aligned} \right.$$

$\Rightarrow$ first, x_1 and x_2 can be eliminated from the first and second equations.

Assemblage & Solution

1. Input — Geometry
   Material Propertes
   Element and node numbers
   Loads

2. Element stiffness $\underset{\sim}{k}' = \int \underset{\sim}{B}^T \underset{\sim}{E} \underset{\sim}{B}\, dv$, $\underset{\sim}{k} = \underset{\sim}{R}^T \underset{\sim}{k}' \underset{\sim}{R}$ if neassary

   equivalent nodal forces of element $\underset{\sim}{P}_o$, $\underset{\sim}{P}_b$

3. Assemble Element Stiffness

   Structural Stiffness $\underset{\sim}{K} = \Sigma \underset{\sim}{k}$

4. Assemble Equivalent nodal forces $\underset{\sim}{P}_o$, $\underset{\sim}{P}_b$

5. Set-up Equilibrium Equation

   $$\underset{\sim}{K}\underset{\sim}{U} = \underset{\sim}{P} + \underset{\sim}{P}_o + \underset{\sim}{P}_b$$

6. Modified $\underset{\sim}{K}$ and $\underset{\sim}{P}$s with Boundary conditions

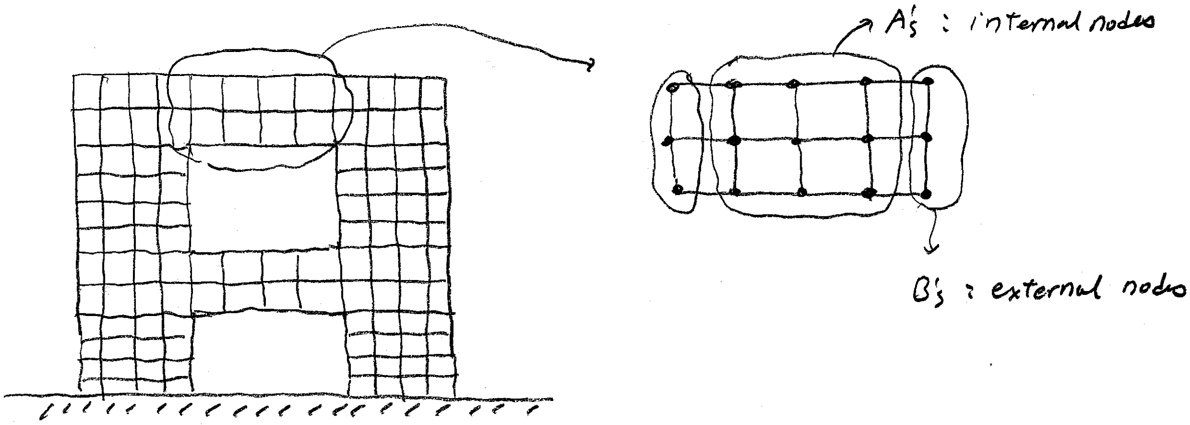
7. Solve $\underset{\sim}{U}$ (free displacements) and reactions

8. Calculate strains and stresses with calculated displacements

   $$\underset{\sim}{\varepsilon} = \underset{\sim}{B}\,\underset{\sim}{\delta}$$

   $$\underset{\sim}{\sigma} = \underset{\sim}{E}(\underset{\sim}{\varepsilon} - \underset{\sim}{\varepsilon}_o)$$

Substructuring – matrix condensation

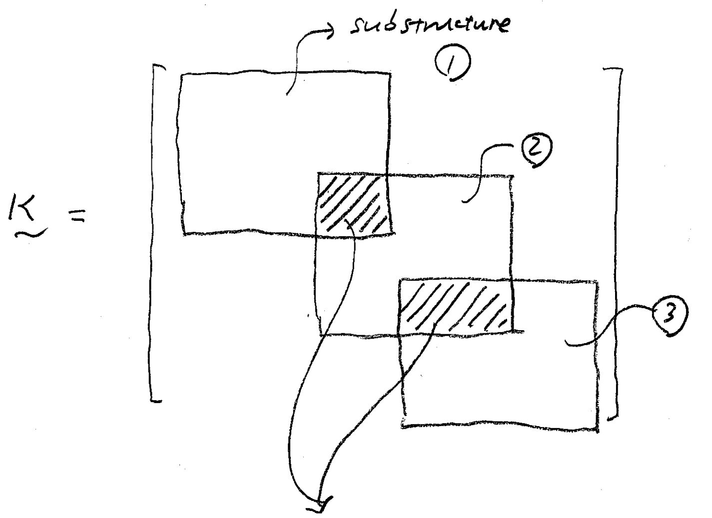


$$\begin{bmatrix} \underset{\sim}{K}_{AA} & \underset{\sim}{K}_{AB} \\ \underset{\sim}{K}_{BA} & \underset{\sim}{K}_{BB} \end{bmatrix} \begin{bmatrix} \underset{\sim}{u}_A \\ \underset{\sim}{u}_B \end{bmatrix} = \begin{bmatrix} \underset{\sim}{P}_A \\ \underset{\sim}{P}_B \end{bmatrix}$$

By using matrix condensation (substructuring technique)

$$\underbrace{(\underset{\sim}{K}_{BB} - \underset{\sim}{K}_{BA} \underset{\sim}{K}_{AA}^{-1} \underset{\sim}{K}_{AB})}_{\underset{\sim}{K}_B^*} \underset{\sim}{u}_B = \underbrace{\underset{\sim}{P}_B - \underset{\sim}{K}_{BA} \underset{\sim}{K}_{AA}^{-1} \underset{\sim}{P}_A}_{\underset{\sim}{P}_B^*}$$

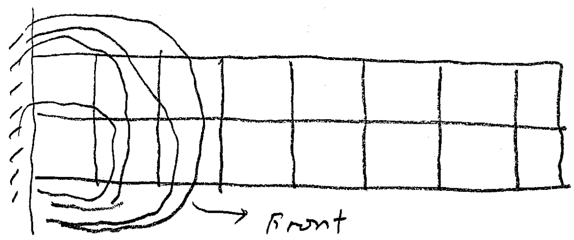
By using substructuring, the number of calculations and the memory space to be required for structural stiffness can be saved.



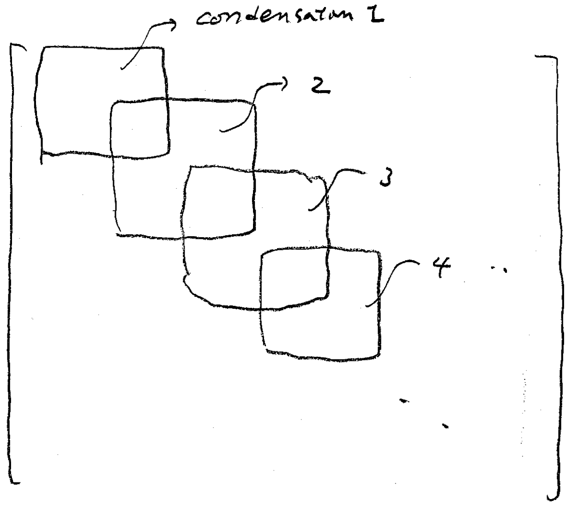
The condensed matrix which remains after substructuring ①, ②, and ③.

Frontal Method

Element by Element matrix condensation



$\underset{\sim}{K} =$



Solution Technique

- Inversion
- Direct method
  - Cramer's Rule
  - Gauss Elimination
  - LU Factorization , cholesky's method
- Indirect Method
  - Iterative method
    - Total steps - Jacobi
    - Single step - Gauss - Seidel

Inversion

$$\underset{\sim}{A}\,\underset{\sim}{x} = \underset{\sim}{b} \qquad \underset{\sim}{x} = \underset{\sim}{A}^{-1}\underset{\sim}{b}$$

$\Rightarrow$ Inefficient since it requires the evaluation of a number of determinants of high order in calculating $\underset{\sim}{A}^{-1}$

Gauss Elimination

$$\underset{\sim}{K}\,\underset{\sim}{u} = \underset{\sim}{P}$$

$$\begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1n} \\ k_{21} & k_{22} & \cdots & k_{2n} \\ \vdots & & & \vdots \\ k_{n1} & k_{n2} & \cdots & k_{nn} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{bmatrix}$$

$$\Downarrow$$

$$\begin{array}{r} \\ \text{elimination} \rightarrow \\ \text{elimination} \rightarrow \\ \\ \\ \end{array} \underbrace{\begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1n} \\ 0 & k_{22}^* & \cdots & k_{2n}^* \\ 0 & 0 & k_{33}^* & \vdots \\ & & \ddots & \\ 0 & 0 & 0 \cdots & k_{nn} \end{bmatrix}}_{\text{triangular matrix}} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \\ u_n \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2^* \\ P_3^* \\ \vdots \\ P_n^* \end{bmatrix} \Rightarrow \text{forwarding}$$

Back-substitution

from the Bottom, u_i can be calculated "subsequently.

LU - Factorization

$$\underset{\sim}{K}\,\underset{\sim}{V} = \underset{\sim}{P}$$

$$\underset{\sim}{K} = \underset{\sim}{L}\,\underset{\sim}{U}$$

$\underset{\sim}{L}$ = Lower triangular matrix

$\underset{\sim}{U}$ = upper triangular matrix

$$\begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 0 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

$$\underset{\sim}{L}\,\underset{\sim}{U}\,\underset{\sim}{V} = \underset{\sim}{P}$$

$$\underset{\sim}{U}\,\underset{\sim}{V} = \underset{\sim}{Y} \quad \text{— ②}$$

$$\underset{\sim}{L}\,\underset{\sim}{Y} = \underset{\sim}{P} \quad \text{— ①}$$

solve ① first, Then ②

cholesky's method

If $\underset{\sim}{K}$ is symmetric and positive definite, $(\underset{\sim}{X}^T \underset{\sim}{K}\,\underset{\sim}{X} > 0)$,

$$\underset{\sim}{U} = \underset{\sim}{L}^T \qquad \underset{\sim}{K} = \underset{\sim}{L}\,\underset{\sim}{L}^T$$

$$\begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

$$\underset{\sim}{L}\,\underset{\sim}{L}^T \underset{\sim}{V} = \underset{\sim}{P}$$

$$\underset{\sim}{L}\,\underset{\sim}{Y} = \underset{\sim}{P} \quad \text{— ①}$$

$$\underset{\sim}{L}^T \underset{\sim}{V} = \underset{\sim}{Y} \quad \text{— ②}$$

Iterative Method

iterative method is advantageous in calculation (fast convergence) when

1) Matrices have large main diagonal entries.

2) Matrices are sparse, that is, have very many zeros.

$$\underset{\sim}{K}\,\underset{\sim}{u} = \underset{\sim}{P}$$

Gauss - Seidel Iteration Method $\Rightarrow$ successive correction

$$\begin{bmatrix} 1 & -0.25 & -0.25 & 0 \\ -0.25 & 1 & 0 & -0.25 \\ -0.25 & 0 & 1 & -0.25 \\ 0 & -0.25 & -0.25 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 50 \\ 50 \\ 25 \\ 25 \end{bmatrix}$$

$$\underset{\sim}{K}^{*} \text{(with diagonals = 1)} \quad \underset{\sim}{u} = \underset{\sim}{P}^{*}$$

modified stiffness

Assuming $\underset{\sim}{u}^{(0)} = \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \end{bmatrix}$

$$u_1^{(1)} = \qquad 0.25\, u_2^{(0)} + 0.25\, u_3^{(0)} + 50 = 100$$

$$u_2^{(1)} = 0.25\, u_1^{(1)} \qquad + 0.25\, u_4^{(0)} + 50 = 10$$

$$u_3^{(1)} = 0.25\, u_1^{(1)} \qquad + 0.25\, u_4^{(0)} + 25 = 75$$

$$u_4^{(1)} = \qquad 0.25\, u_2^{(1)} + 0.25\, u_3^{(1)} \qquad = 68.7$$

$$\underset{\sim}{K} = \underset{\sim}{I} + \underset{\sim}{L} + \underset{\sim}{U}$$

$$\underset{\sim}{K}\,\underset{\sim}{u} = (\underset{\sim}{I} + \underset{\sim}{L} + \underset{\sim}{U})\,\underset{\sim}{u} = \underset{\sim}{P}$$

$$\underset{\sim}{u}^{(m+1)} = \underset{\sim}{P} - \underset{\sim}{L}\,\underset{\sim}{u}^{(m+1)} - \underset{\sim}{U}\,\underset{\sim}{u}^{(m)}$$

$$(\underset{\sim}{I} + \underset{\sim}{L})\,\underset{\sim}{u}^{(m+1)} = \underset{\sim}{P} - \underset{\sim}{U}\,\underset{\sim}{u}^{(m)}$$

$$\underset{\sim}{u}^{(m+1)} = (\underset{\sim}{I} + \underset{\sim}{L})^{-1}\,\underset{\sim}{P} - (\underset{\sim}{I} + \underset{\sim}{L})^{-1}\,\underset{\sim}{U}\,\underset{\sim}{u}^{(m)}$$

Jacobi Iteration – simultaneous correction
(total)

$$\underset{\sim}{K}\,\underset{\sim}{u} = \underset{\sim}{I}\,\underset{\sim}{X} + (\underset{\sim}{A} - \underset{\sim}{I})\,\underset{\sim}{X} = \underset{\sim}{P}$$

$$\underset{\sim}{u}^{(m+1)} = \underset{\sim}{P} + (\underset{\sim}{I} - \underset{\sim}{A})\,\underset{\sim}{u}^{(m)}$$

An **LU-factorization** of a given square matrix **A** is of the form

(2)
$$\boxed{\mathbf{A} = \mathbf{LU}}$$

where **L** is lower triangular and **U** is upper triangular. For example,

$$\mathbf{A} = \begin{bmatrix} 2 & 3 \\ 8 & 5 \end{bmatrix} = \mathbf{LU} = \begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ 0 & -7 \end{bmatrix}.$$

It can be proved that for any nonsingular matrix (see Sec. 6.7) the rows can be reordered so that the resulting matrix **A** has an LU-factorization (2) in which **L** turns out to be the matrix of the *multipliers* m_{jk} of the Gauss elimination, with main diagonal $1, \cdots, 1$, and **U** is the matrix of the triangular system at the end of the Gauss elimination. (See Ref. [E3], pp. 155–156, listed in Appendix 1.)

The *crucial idea* now is that **L** and **U** in (2) can be computed directly, without solving simultaneous equations (thus, without using the Gauss elimination). As a count shows, this needs about $n^3/3$ operations, about half as many as the Gauss elimination, which needs about $2n^3/3$ (see Sec. 18.1). And once we have (2), we can use it for solving $\mathbf{Ax} = \mathbf{b}$ in two steps, involving only about n^2 operations, simply by noting that $\mathbf{Ax} = \mathbf{LUx} = \mathbf{b}$ may be written

(3)
$$\boxed{\text{(a)} \quad \mathbf{Ly} = \mathbf{b} \qquad \text{where} \qquad \text{(b)} \quad \mathbf{Ux} = \mathbf{y}}$$

and solving first (3a) for **y** and then (3b) for **x**. This is called **Doolittle's method.** Both systems (3a) and (3b) are triangular, so their solution is the same as back substitution in the Gauss elimination.

A similar method, **Crout's method,** is obtained from (2) if **U** (instead of **L**) is required to have main diagonal $1, \cdots, 1$. In either case the factorization (2) is unique.

Doolittle's method

Solve the system in Example 1 of Sec. 18.1 by Doolittle's method.

Solution. The decomposition (2) is obtained from

$$\mathbf{A} = [a_{jk}] = \begin{bmatrix} 3 & 5 & 2 \\ 0 & 8 & 2 \\ 6 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

by determining the m_{jk} and u_{jk}, using matrix multiplication. By going through **A** row by row we get successively

$a_{11} = 3 = u_{11}$	$a_{12} = 5 = u_{12}$	$a_{13} = 2 = u_{13}$
$a_{21} = 0 = m_{21}u_{11}$	$a_{22} = 8 = m_{21}u_{12} + u_{22}$	$a_{23} = 2 = m_{21}u_{13} + u_{23}$
$m_{21} = 0$	$u_{22} = 8$	$u_{23} = 2$
$a_{31} = 6 = m_{31}u_{11}$	$a_{32} = 2 = m_{31}u_{12} + m_{32}u_{22}$	$a_{33} = 8 = m_{31}u_{13} + m_{32}u_{23} + u_{33}$
$= m_{31} \cdot 3$	$= 2 \cdot 5 + m_{32} \cdot 8$	$= 2 \cdot 2 - 1 \cdot 2 + u_{33}$
$m_{31} = 2$	$m_{32} = -1$	$u_{33} = 6$

(4)
$$\begin{aligned}
u_{1k} &= a_{1k} & & k = 1, \cdots, n \\
m_{j1} &= \frac{a_{j1}}{u_{11}} & & j = 2, \cdots, n \\
u_{jk} &= a_{jk} - \sum_{s=1}^{j-1} m_{js}u_{sk} & & k = j, \cdots, n; \quad j \geqq 2 \\
m_{jk} &= \frac{1}{u_{kk}}\left(a_{jk} - \sum_{s=1}^{k-1} m_{js}u_{sk}\right) & & j = k+1, \cdots, n; \quad k \geqq 2.
\end{aligned}$$

Cholesky's Method

For a ***symmetric, positive definite*** matrix $\mathbf{A}$ (thus $\mathbf{A} = \mathbf{A}^T$, $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for all $\mathbf{x} \neq \mathbf{0}$) we can in (2) even choose $\mathbf{U} = \mathbf{L}^T$, thus $u_{jk} = m_{kj}$ (but impose no conditions on the main

The popular method of solving $\mathbf{Ax} = \mathbf{b}$ based on this factorization $\mathbf{A} = \mathbf{LL}^T$ is called **Cholesky's method.** In terms of the entries of $\mathbf{L} = [l_{jk}]$ the formulas for the factorization are

$$l_{11} = \sqrt{a_{11}}$$

$$l_{j1} = \frac{a_{j1}}{l_{11}} \qquad j = 2, \cdots, n$$

$$(6) \qquad l_{jj} = \sqrt{a_{jj} - \sum_{s=1}^{j-1} l_{js}^2} \qquad j = 2, \cdots, n$$

$$l_{pj} = \frac{1}{l_{jj}}\left(a_{pj} - \sum_{s=1}^{j-1} l_{js} l_{ps}\right) \qquad p = j + 1, \cdots, n; \quad j \geqq 2.$$

If $\mathbf{A}$ is symmetric but not positive definite, this method could still be applied, but then leads to a *complex* matrix $\mathbf{L}$, so that it becomes impractical.

Cholesky's method

Solve by Cholesky's method:

$$\begin{aligned} 4x_1 + 2x_2 + 14x_3 &= 14 \\ 2x_1 + 17x_2 - 5x_3 &= -101 \\ 14x_1 - 5x_2 + 83x_3 &= 155. \end{aligned}$$

Solution. From (6) or from the form of the factorization

$$\begin{bmatrix} 4 & 2 & 14 \\ 2 & 17 & -5 \\ 14 & -5 & 83 \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

we compute, in the given order,

$$l_{11} = \sqrt{a_{11}} = 2 \qquad l_{21} = \frac{a_{21}}{l_{11}} = \frac{2}{2} = 1 \qquad l_{31} = \frac{a_{31}}{l_{11}} = \frac{14}{2} = 7$$

$$l_{22} = \sqrt{a_{22} - l_{21}^2} = \sqrt{17 - 1} = 4$$

$$l_{32} = \frac{1}{l_{22}}(a_{32} - l_{31}l_{21}) = \frac{1}{4}(-5 - 7 \cdot 1) = -3$$

$$l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2} = \sqrt{83 - 7^2 - (-3)^2} = 5.$$

Gauss–Seidel Iteration Method

This is an iterative method of great practical importance, which we can simply explain in terms of an example.

Gauss–Seidel iteration

We consider the linear system

$$
\begin{array}{rrrrl}
x_1 & -\,0.25x_2 & -\,0.25x_3 & & = 50 \\
-0.25x_1 & +\;\; x_2 & & -\,0.25x_4 & = 50 \\
-0.25x_1 & & +\;\; x_3 & -\,0.25x_4 & = 25 \\
 & -\,0.25x_2 & -\,0.25x_3 & +\;\; x_4 & = 25.
\end{array} \tag{1}
$$

(Equations of this form arise in the numerical solution of partial differential equations and in spline interpolation.) We write the system in the form

$$
\begin{array}{rlrrr}
x_1 = & & 0.25x_2 + 0.25x_3 & & +\ 50 \\
x_2 = & 0.25x_1 & & +\ 0.25x_4 & +\ 50 \\
x_3 = & 0.25x_1 & & +\ 0.25x_4 & +\ 25 \\
x_4 = & & 0.25x_2 + 0.25x_3 & & +\ 25.
\end{array} \tag{2}
$$

We use these equations for iteration, that is, we start from a (possibly poor) approximation to the solution, say, $x_1^{(0)} = 100$, $x_2^{(0)} = 100$, $x_3^{(0)} = 100$, $x_4^{(0)} = 100$, and compute from (2) a presumably better approximation

Use "old" values

("New" values here not yet available)

$$
\begin{array}{rlrrl}
x_1^{(1)} = & & 0.25x_2^{(0)} + 0.25x_3^{(0)} & & + 50.00 = 100.00 \\
x_2^{(1)} = & 0.25x_1^{(1)} & & + 0.25x_4^{(0)} & + 50.00 = 100.00 \\
x_3^{(1)} = & 0.25x_1^{(1)} & & + 0.25x_4^{(0)} & + 25.00 = 75.00 \\
x_4^{(1)} = & & 0.25x_2^{(1)} + 0.25x_3^{(1)} & & + 25.00 = 68.75.
\end{array} \tag{3}
$$

Use "new" values

We see that these equations are obtained from (2) by substituting on the right the ***most recent*** approximations. In fact, corresponding elements replace previous ones as soon as they have been computed, so that in the second and third equations we use $x_1^{(1)}$ (not $x_1^{(0)}$), and in the last equation of (3) we use $x_2^{(1)}$ and $x_3^{(1)}$ (not $x_2^{(0)}$ and $x_3^{(0)}$). The next step yields

$$
\begin{array}{rlrrl}
x_1^{(2)} = & & 0.25x_2^{(1)} + 0.25x_3^{(1)} & & + 50.00 = 93.75 \\
x_2^{(2)} = & 0.25x_1^{(2)} & & + 0.25x_4^{(1)} & + 50.00 = 90.62 \\
x_3^{(2)} = & 0.25x_1^{(2)} & & + 0.25x_4^{(1)} & + 25.00 = 65.62 \\
x_4^{(2)} = & & 0.25x_2^{(2)} + 0.25x_3^{(2)} & & + 25.00 = 64.06.
\end{array}
$$

In practice, one would do further steps and obtain a more accurate approximate solution.

The reader may show that the exact solution is $x_1 = x_2 = 87.5$, $x_3 = x_4 = 62.5$. ◀

To obtain an algorithm for the Gauss–Seidel iteration, let us derive the general formulas for this iteration.

We assume that $a_{jj} = 1$ for $j = 1, \cdots, n$. (Note that this can be achieved if we can rearrange the equations so that no diagonal coefficient is zero; then we may divide each equation by the corresponding diagonal coefficient.) We now write

$$\mathbf{A} = \mathbf{I} + \mathbf{L} + \mathbf{U} \qquad (a_{jj} = 1) \tag{4}$$

where $\mathbf{I}$ is the $n \times n$ unit matrix and $\mathbf{L}$ and $\mathbf{U}$ are respectively lower and upper triangular matrices with zero main diagonals. If we substitute (4) into $\mathbf{Ax} = \mathbf{b}$, we have

$$\mathbf{Ax} = (\mathbf{I} + \mathbf{L} + \mathbf{U})\mathbf{x} = \mathbf{b}.$$

Taking $\mathbf{Lx}$ and $\mathbf{Ux}$ to the right, we obtain, since $\mathbf{Ix} = \mathbf{x}$,

$$\mathbf{x} = \mathbf{b} - \mathbf{Lx} - \mathbf{Ux}. \tag{5}$$

Remembering from our computation in Example 1 that below the main diagonal we took "new" approximations and above the main diagonal "old" approximations, we obtain from (5) the desired iteration formulas

$$\boxed{\mathbf{x}^{(m+1)} = \mathbf{b} - \mathbf{Lx}^{(m+1)} - \mathbf{Ux}^{(m)}} \qquad (a_{jj} = 1) \tag{6}$$

where $\mathbf{x}^{(m)} = [x_j^{(m)}]$ is the mth approximation and $\mathbf{x}^{(m+1)} = [x_j^{(m+1)}]$ is the $(m + 1)$st approximation. In components this gives the formula in line 1 in Table 18.2. The matrix

Jacobi Iteration

The Gauss–Seidel iteration is a method of **successive corrections** because we replace approximations by corresponding new ones as soon as the latter have been computed. A method is called a method of **simultaneous corrections** if no component of an approximation $\mathbf{x}^{(m)}$ is used until ***all*** the components of $\mathbf{x}^{(m)}$ have been computed. A method of this type is the **Jacobi iteration,** which is similar to the Gauss–Seidel iteration but involves *not* using improved values until a step has been completed and then replacing $\mathbf{x}^{(m)}$ by $\mathbf{x}^{(m+1)}$ at once, directly before the beginning of the next cycle. Hence, if we write $\mathbf{Ax} = \mathbf{b}$ (*with* $a_{jj} = 1$ *as before!*) in the form $\mathbf{x} = \mathbf{b} + (\mathbf{I} - \mathbf{A})\mathbf{x}$, the Jacobi iteration in matrix notation is

$$\boxed{\mathbf{x}^{(m+1)} = \mathbf{b} + (\mathbf{I} - \mathbf{A})\mathbf{x}^{(m)}} \qquad (a_{jj} = 1). \tag{13}$$

This method converges for every choice of $\mathbf{x}^{(0)}$ if and only if the spectral radius of $\mathbf{I} - \mathbf{A}$ is less than 1. It has recently gained greater practical interest since on parallel processors all n equations can be solved simultaneously at each iteration step.