NASA-CR-194,474

NASA Contractor Report 194474

Hierarchically Parallelized Constrained Nonlinear Solvers With Automated Substructuring

NASA-CR-194474
19940033043

Joe Padovan and Abel Kwang
The University of Akron
Akron, Ohio

June 1994

Prepared for
Lewis Research Center
Under Grant NAG3–1142

National Aeronautics and
Space Administration

HIERARCHICALLY PARALLELIZED CONSTRAINED NONLINEAR SOLVERS WITH AUTOMATED SUBSTRUCTURING[†]

Joe Padovan[*]

Abel Kwang[∇]

[*]Departments of Mechanical and Polymer Engineering
[∇]Department of Mechanical Engineering

[†]Work supported by NASA Lewis under Grant NAG 3-1142

Abstract

This paper develops a parallelizable multilevel multiple constrained nonlinear equation solver. The substructuring process is automated to yield appropriately balanced partitioning of each succeeding level. Due to the generality of the procedure, both sequential, partially and fully parallel environments can be handled. This includes both single and multi processor assignment per individual partition. Several benchmark examples are presented. These illustrate the robustness of the procedure as well as its capability to yield significant reductions in memory ultilization and calculational effort due both to updating and inversion.

I. Introduction

For nonlinear static and implicit transient finite element or difference simulations, generally some form of Newton Raphson[1-3] iterative algorithm is typically employed to solve the resulting nonlinear set of equations. Prior to the 1980's, generally the classical version of the scheme[1-3] was the preferred method. With the advent of constrained adaptions, problems exhibiting postbuckling behavior can now be standardly handled. To date a wide variety of constraint procedures have been developed, for instance

i) Arc length control[4]

ii) Linear[5], circular[6,7], elliptic[8,9] and piecewise continuous[10] constraints as well as

iii) The enforcement of bounds on successive stress, strain and energy[8,10,11] excursions.

More recently, modifications have been introduced to partition and parallelize the scheme. These include a variety of different approaches, i.e.

i) The least square method[12]

ii) The mixed direct-iterative solution of the stiffness matrix[13]

iii) The use of multiple/multilevel constraints[14,15] and

iv) The use of progressive substructuring[16].

Overall the foregoing[4-16] procedures have greatly widened our ability to tackle highly nonlinear response problems including the interaction of contact, large deformation kinematics, complex material response, postbuckling, etc.

The thrust to parallelize the Newton Raphson family of algorithms has risen out of the need to handle ever increasing problem sizes. The heart of the difficulty lies in the storage and inversion of the tangent matrix. While attempts to use progressive substructuring[13,16,17] point to potentially significant gains, overall the approach typically yields hit or miss improvements[18-20]. This follows from the fact that generally no attempts have been made to balance memory and computational efforts. As a result, progressive substructuring can lead to an imbalance between internal and external variables yielding potentially increased costs in

a) communications

b) memory and

c) computational effort.

In the context of the foregoing, this paper will develop a parallelizable multilevel constrained nonlinear equation solver. To provide for a proper balance between the internal and external variables associated with successive levels, the hierarchical poly tree (HPT) of Padovan and Gute[18-20] will be employed. This will enable the minimization of computational effort, communications and memory requirements. Concomitantly, the multilevel substructuring process will be automated to yield the appropriate partitioning of each succeeding level.

Due to the generality of the procedure, both sequential, partially and fully parallel environments can be handled. Two approaches to parallelism will be undertaken, i.e.

1) Where single processor assignment is defined for each partition and

2) Where multiple processor assignment is employed.

To further extend the range and capability of the scheme

i) A variety of local partition level update schemes will be explored, namely BFGS[21], full, and modified[1-3]

ii) Local and global convergence criteria will be developed and

iii) A variety of constraint procedures will be explored, i.e. global/individual partition level/... etc.

In the sections which follow, detailed discussions will be given on problem formulation, hierarchically substructured constrained solvers and automated substructuring. The results will be benchmarked in a series of numerical examples.

II. Problem Formulation: Assessment of Current Capabilities

For structural systems composed of nonlinear media undergoing potentially large deformations/strain excursions, the governing equations of motion take the form

$$\frac{\partial}{\partial x_j}\left\{S_{jk}\left(\delta_{ik}+\frac{\partial}{\partial x_k}(U_i)\right)\right\}+\rho f_i=\rho\frac{d^2}{dt^2}(U_i) \tag{1}$$

where S_{jk}, U_i, f_i, ρ, δ_{ik}, x_j and t respectively represent the 2nd Piola Kirchoff stress tensor, the displacement vector, body force, density, Kronecker delta, Lagrangian coordinates and time. The boundary conditions associated with (1) are given by[22]

i) for $\forall\, x \,\varepsilon\, \partial V_s$;

$$S_{jk}\left(\delta_{ik}+\frac{\partial}{\partial x_k}(U_i)\right)n_j=S_i^* \tag{2}$$

ii) for $\forall\, x \,\varepsilon\, \partial V_u$;

$$U_i=U_i^* \tag{3}$$

where n_j, S_i^* and U_i^* respectively define the surface normal, prescribed surface traction (on ∂V_s) and displacement (on ∂V_u). Given the use of the 2nd Piola Kirchoff stress measure, the stress-strain relation will be cast in the form[22]

$$S_{ij}=S_{ij}(L_{11},L_{12},\ldots) \tag{4}$$

where L_{ij} are components of the Green-Lagrangran strain tensor, i.e.

$$L_{ij}=\frac{1}{2}\{U_{i,j}+U_{j,i}+U_{\ell,i}U_{\ell,j}\} \tag{5}$$

Assuming a displacement type formulation, it follows that

$$U=[N]Y \tag{6}$$

$$\frac{d^2}{dt^2}(\mathrm{U}) = [N]\,\mathrm{Y} \tag{7}$$

where [N], Y and Y are the shape function, nodal displacement and nodal acceleration. Based on the use of the virtual work principle, (1-7) yield the following FE formulation namely[1-3]

$$\int_V [B^*]^T \mathrm{S}\, dv + [M]\frac{d^2}{dt^2}(\mathrm{Y}) = \mathrm{R} \tag{8}$$

such that S, [M], R are respectively the 2nd Piola-Kirchhoff tensor cast in vector form, the lumped/consistent mass matrix, the nodal force vector and[1-3]

$$[B^*] = [B] + [B_N(\mathrm{Y})] \tag{9}$$

Since (8) is highly nonlinear, both its static and implicit transient solution requires introduction of the tangent stiffness formulation namely

$$\int_V [B^*]^T \mathrm{S}\, dv \sim \int_V [B^*]^T \mathrm{S}\, dv \,\Big|_{\mathrm{Y}_o} + [K_T]\,\Delta\mathrm{Y} \tag{10}$$

where[1-3]

$$[K_T] = \int_V \{[G]^T[S][G] + [B^*]^T[D_T][B^*]\}\, dv \tag{11}$$

such that [S] and [D_T] are the prestress matrix and material tangent stiffness. Based on (10), (8) reduces to the form

$$[M]\frac{d^2}{dt^2}(\mathrm{Y}) + [K_T]\,\Delta\mathrm{Y} = \mathrm{R} - \int_V [B^*]^T \mathrm{S}\, dv|_{\mathrm{Y}_o} \tag{12}$$

Employing various of the implicit solvers[23], (12) can be recast to yield the expression

$$[K_D]\,\Delta\mathrm{Y} = \mathrm{R}_D - \int_V [B^*]^T \mathrm{S}\, dv|_{\mathrm{Y}_o} \tag{13}$$

such that $[K_D]$ and R_D are respectively the dynamic stiffness and load, i.e. a function of Δt the time step. For static problems, (13) reduces to

$$[K_T]\Delta Y = R - \int_V [B^*]^T S \, dv\Big|_{Y_o} \tag{14}$$

Equations (13) and (14) define the core Newton Raphson relation. Cast in algorithmic form, we yield the expression

$$[K_D(Y_{i-1}^j)]\Delta Y_i^j = R_D^j - \int_V [B^*]^T S \, dv\Big|_{Y_{i-1}^j} \tag{15}$$

such that j defines the load/time step increment and i the iteration count for the given increment.

To control/constrain successive iterate excursions, typically a one parameter bound is introduced to scale R_D^j namely[4-10]

$$\Delta Y_i^j = [K_D(Y_{i-1}^j)]^{-1}\left\{\lambda_i^j \Delta R_D^j + R_D^{j-1} - \int_V [B^*]^T S \, dv\Big|_{Y_{i-1}^j}\right. \tag{16}$$

where λ_i^j is defined by a constraint condition. Considering the case of an elliptic constraint surface[8], Fig. 2.1, λ_i^j is chosen to satisfy the relation

$$\mu\|Y_i^j\|^2 + \lambda_i^j\|R_D^j\|^2 = \|R_D^j\|^2 \tag{17}$$

such that μ is the aspect ratio of the ellipse. The parameter μ can be selected by various of the following criteria[10], i.e.:

1) By defining an allowable excursion for $\|\Delta Y_i^j\|$, i.e. a global condition;
2) By restricting the allowable excursion of a given node (a local condition) or
3) By restricting the allowable excursions in stress/steam or energy – either globally or locally.

The algorithm (16) can be updated in three different formats

1) Fully – $[K_T(Y_i^j)]$ is continuously updated and inverted

2) Modified – $[K_T(Y_\ell^j)]$ is intermittently updated and inverted – at the beginning of each new load step or

3) BFGS – $[K_T(Y_0^j)]$ is updated by pre and post multiplication via appropriate rescaling metrics[10,21].

These update strategies pose the main short coming to the Newton Raphson scheme. In particular the updating and inversion of $[K_T(Y_\ell^j)]$ requires significant storage and computational effort.

Considering 2-D and 3-D square and cubic regions with N nodes on an edge, it follows that asymptotically the following work load (ω) and memory (μ) requirements are defined namely:

i) 2-D: Fig 2.2

$$\omega_{2D} \sim O(N^4)$$

$$\mu_{2D} \sim O(N^3) \tag{18}$$

ii) 3-D: Fig 2.2

$$\omega_{3D} \sim O(N^7)$$

$$\mu_{3D} \sim O(N^5) \tag{19}$$

These trends obviously point to very disturbing consequences as problem size/ complexity grows.

Substructuring can potentially reduce the calculation burden. For the classic two level case, considering the foregoing 2-D square region, it follows that for partitioning into κ^2 square regions, the net effort is defined by a two tiered expression, i.e.

$$\omega_{2D} \sim (\omega_{2D})_1 + (\omega_{2D})_2 \tag{20}$$

where $(\omega_{2D})_1$, and $(\omega_{2D})_2$ are espectively the work loads associated with the root and branch levels. Recalling the work of Padovan, Gute and Johnson[18], it follows that the operations count can be defined by tracing the number of row and column multiplications and additions. Note here an operation will be defined as a multiplication – addition pair. Several forms of operational control are possible for the two tiered format, in particular:

1) All substructure are handled sequentially by a single processor, Fig. 2.3

2) For a P processor machine

   - Each of the P processors can be individually reattached to separate new partitions upon completion of work in prior assignments, Fig. 2.3 or
   - Sets of processors may be continuously reattached to partitions – single assignment to a branch multiple at root, Fig. 2.3 – multiple assignment at both branch and root, Fig. 2.4.

Performing the requisite operations count, asymptotically the work load at the root and individual 2nd level partitions is given by the relations

$$E_R \sim (\zeta)^3 \frac{(N)^3}{4\kappa} \left(\frac{9\kappa - 24}{\kappa - 1} \right) (\kappa)^2 \tag{21}$$

$$E_P \sim \frac{9}{2} (\zeta)^3 \left(\frac{(N)}{\kappa} \right)^4 \tag{22}$$

such that E_R, E_P, κ and ζ respectively denote the root effort, 2nd level partition effort, the number of partitions on an edge and the number of degrees of freedom per node. Note the net global work load is given by

$$E_G = \frac{1}{2} (\zeta)^3 (N)^4 \tag{23}$$

Employing (23) to normalize the effort counts of the sequential and (partially/fully) parallel cases, we yield the following relationships:

i) Purely sequential (single processor machine)

$$\left(\frac{\omega_{2D}}{E_G}\right)_S = \left(\frac{E_R}{E_G}\right) + \left(\frac{E_P}{E_G}\right)\kappa^2 \sim \frac{\kappa}{4N}\left(\frac{9\kappa-24}{\kappa-1}\right) + \frac{9}{(\kappa)^2} \tag{24}$$

ii) Partially (sub) parallel (P processor machine, single assignment, $(\kappa)^2 > \mathrm{P}$)

$$\left(\frac{\omega_{2D}}{E_G}\right)_{PS} = \left(\frac{E_R}{E_G}\right) + \left(\frac{E_P}{E_G}\right)\frac{(\kappa)^2}{P} \sim \frac{\kappa}{4N}\left(\frac{9\kappa-24}{\kappa-1}\right) + \frac{9}{(\kappa)^2 P} \tag{25}$$

iii) Partially (sub) parallel (P processor machine, multiple assignment)

$$\left(\frac{\omega_{2D}}{E_G}\right)_P = \left(\frac{E_R}{E_G}\right)\frac{1}{P} + \left(\frac{E_P}{E_G}\right)\frac{(\kappa)^2}{P} \sim \frac{\kappa}{4NP}\left(\frac{9\kappa-24}{\kappa-1}\right) + \frac{9}{(\kappa)^2 P} \tag{26}$$

iv) Fully (iso) parallel (κ^2 processor machine, multiple processor assignment)

$$\left(\frac{\omega_{2D}}{E_G}\right)_P = \left(\frac{E_R}{E_G}\right)\frac{1}{\kappa^2} + \left(\frac{E_P}{E_G}\right) \sim \frac{1}{4N\kappa}\left(\frac{9\kappa-24}{\kappa-1}\right) + \frac{9}{(\kappa)^4} \tag{27}$$

v) Super parallel ($\mathrm{P} > \kappa^2$ processor machine, multiple processor assignment)

$$\left(\frac{\omega_{2D}}{E_G}\right)_P = \left(\frac{E_R}{E_G}\right)\frac{1}{P} + \left(\frac{E_P}{E_G}\right)\frac{(\kappa)^2}{P} \tag{28}$$

Based on the trends defined by (24)-(28), it follows that significant improvements are possible for both the sequential and sub/iso/super parallel arrangements. This must be tempered by the fact that in the case of multiple processor assignment, significant loses in efficiency occur – a direct result of Amdhal's law. Noting Table 5.6 which illustrates the effects of the number of processors on overall efficiency, we see that significant reductions are recorded as (P,N) are increased. In this context, the P in multiple assignment areas must be replaced by

$$P_e = P_e(P,N) \tag{29}$$

such that $\mathrm{P_e}$ denotes the effective number of processors.

All this points to the fact that great care must be taken in balancing the number of substructure as well as their external to internal variable ratios. For instance noting the subparallel work effort ratio defined by (26), it follows that replacing P by P_e, i.e. (29), we yield the expression

$$\left(\frac{\omega_{2D}}{E_G}\right)_P \sim \frac{\kappa}{4NP_e}\left(\frac{9\kappa-24}{\kappa-1}\right)+\frac{9}{(\kappa)^4 P_e} \tag{30}$$

Given realistic machine configurations, P_e generally reaches a saturation value, i.e. P_S. In this context (30) reduces to the form

$$\left(\frac{\omega_{2D}}{E_G}\right)_P \sim \frac{1}{P_S}\left\{\frac{\kappa}{4N}\left(\frac{9\kappa-24}{\kappa-1}\right)+\frac{9}{(\kappa)^2}\right\} \tag{31}$$

Here we see that the delimits controlling improvement are κ and N. For a given N, (31) illustrates that there is a critical value of κ yielding peak work load reduction, i.e.

$$\kappa_{critical} \sim \sqrt[5]{8N} \tag{32}$$

This points to the fact that for a given machine configuration, the proper tuning of the substructuring process strongly influences speedup potential.

Similar trends apply to memory usage. Note for the 2-D $(\kappa)^2$ square region, the net memory requirement is defined by the expression

$$\mu_{2D} = (\mu_{2D})_1 + (\mu_{2D})_2 \tag{33}$$

where $(\mu_{2D})_1$ and $(\mu_{2D})_2$ are respectively the memory requirements associated with the root and branch levels. Performing the requisite count, the root and individual partition level memory requirements are defined by the expressions

$$(\mu_{2D})_1 \sim \frac{1}{2}(\zeta)^2[6(\kappa)^3 - 8(\kappa)^2 - 2\kappa + 2]\left(\frac{N}{\kappa}\right)^2 \tag{34}$$

$$(\mu_{2D})_2 \sim \frac{3\zeta^2}{2}\left(\frac{N}{\kappa}\right)^2 \kappa^2 \tag{35}$$

Based on (33-35), we again see that the optimal memory reduction is controlled by the (N,κ) pair. To achieve further gains, one must move to multilevel substructuring. Additionally, procedures need to be established which enable the proper tuning/ balancing of the substructuring for general shapes.

Beyond the inversion problem, difficulties also arise out of the need to update. For nonlinear problems, the update process is generally on the same order of magnitude as the inverse problem. In this context the net computational burden is given by

$$\omega_{net} \sim (w_{2D})_{inverse} + (\omega_{2D})_{update} \tag{36}$$

where typically

$$\begin{aligned}(\omega_{2D})_{update} &\sim O\,\{(w_{2D})_{inverse}\}\\ &\sim \Gamma\,(w_{2D})_{inverse}\end{aligned} \tag{37}$$

such that generally $\Gamma\ \varepsilon\ (O, 1)$. In this context, it follows that to parallelize the net effort, assuming partial parallelism

$$\omega_{2D} \sim \frac{1}{P}\sum_{\ell}^{NE} E_U^{\ell} \tag{38}$$

wherein NE and $E_U{}^{\ell}$ are the number of elements and the update effort for the ℓth element. If the element level data is cloned off into partitioned sets, then the level of contention between processors is reduced especially for systems with some localized memory. Such characteristics tend to drive the architecture to large P, i.e. a large number of processors. Since the two level tree is κ limited, a multilevel tree (MLT) would be required to enable a balanced growth of the number of separate top level partitions. Furthermore, there would have to be a balancing of multiprocessor reassignments due to system saturation.

III. Hierarchically Scaled and Partitioned Constrained Solvers

As noted earlier, to provide a proper balance between externals and internals, depending on problem size and connectivity, several levels of substructuring may be necessary. Referring to Fig. 3.1, a multilevel tree (MLT) can be associated with the partitioning process. Such a MLT defines the flow of control of the various stages of intermediate forward elimination, backsubsitution and assembly. Overall the MLT consists of the top-intermediate branches and root. Element generation occurs at the top branches. The intermediate branches and root are a result of the successive stages of the forward elimination process.

At a typical top level branch, the partitioned version of (15) takes the form

$$ {}^{p}_{\Lambda}[K_D(Y^j_{i-1})]\, {}^{p}_{\Lambda}\Delta Y^j_i = {}^{p}_{\Lambda}R^j_D - \int_{{}^{p}_{\Lambda}V} [B^*]^T S\, dv \Big|_{Y^j_{i-1}} \tag{39} $$

where the various pre-post super-subscripts denote

p – partition number

Λ – top level branch

j – increment number

i – iteration count

Introducing separate partition level constraints, (39) yields the following expression namely

$$ {}^{p}_{\Lambda}[K_D(Y^j_{i-1})]\, {}^{p}_{\Lambda}\Delta Y^j_i = {}^{p}_{\Lambda}R^{j-1}_D + {}^{p}_{\Lambda}[\Delta R^j_D]\, {}^{p}_{\Lambda}\lambda^j_i - \int_{{}^{p}_{\Lambda}V} [B^*]^T S\, dv \Big|_{Y^j_{i-1}} \tag{40} $$

such that here

$$ {}^{p}_{\Lambda}[\Delta R^j_D] = {}^{p}_{\Lambda}\begin{bmatrix} (\Delta R^j_D)_I & 0 \\ & \\ 0 & (\Delta R^j_D)_E \end{bmatrix} \tag{41} $$

$$ {}_{\Lambda}^{p}\lambda_i^j = \left\{ \begin{array}{c} {}_{\Lambda}^{p}\lambda_i^j \\ \\ {}_{E}\lambda_i^j \end{array} \right\} \qquad (42) $$

where

${}_{\Lambda}^{p}\lambda_i^j$ – p^th partition constraint of Λ^{th} top level branch

${}_{E}\lambda_i^j$ – common root level

Note the parameter ${}_{E}\lambda_i^j$ is used to provide a global constraint on the solution. Overall it controls the I/O among the various participating substructure. Its solution is obtained once the root equations are assembled. The local parameters ${}_{\Lambda}^{p}\lambda_i^j$ control the flow of partition level I/O into the system. In this context they serve a dual role namely:

1) To control the flow of ${}_{\Lambda}^{p}(\Delta R_D^j)_I$ as well as,
2) Provide a bound on the ${}_{\Lambda}^{p}\Delta T_i^j$ generated during successive partition level iterations. Their values are established via a local partition/substructure level solution.

Overall ${}_{E}\lambda_i^j$ and ${}_{\Lambda}^{p}\lambda_i^j$ are obtained in a three pass operation. This is described below.

1. First Pass. Initially the local and global constraints are set equal to each other, i.e.

$$ {}_{E}\lambda_i^j = {}_{\Lambda}^{p}\lambda_i^j \qquad (43) $$

for ∀ p and Λ. Hence (40) reduces to the form

$$ {}_{\Lambda}^{p}[K_D(Y_{i-1}^j)]\,{}_{\Lambda}^{p}\Delta Y_i^j = {}_{\Lambda}^{p}R_D^{j-1} + {}_{E}\lambda_i^j\,({}_{\Lambda}^{p}\Delta R_D) - \int_{{}_{\Lambda}^{p}V} [B^*]^T S\, dv \Big|_{Y_{i-1}^j} \qquad (44) $$

Next (44) are solved in a substructural sense, eliminating all local top branch internal variables, i.e.

$${}^{p}_{\Lambda}(\Delta Y^{j}_{i})_{I}$$

This is achieved by restructuring the local tangent stiffness matrix and incremental displacement vector into the following form namely

$${}^{p}_{\Lambda}[K_D] = {}^{p}_{\Lambda}\begin{bmatrix} {}_{I}K_D & {}_{IE}K_D \\ {}_{IE}K_D^T & {}_{E}K_D \end{bmatrix} \tag{45}$$

$${}^{p}_{\Lambda}\Delta Y^{j}_{i} = {}^{p}_{\Lambda}\left\{ \begin{matrix} (\Delta Y)_I \\ (\Delta Y)_E \end{matrix} \right\}^{j}_{i} \tag{46}$$

In terms of (45) and (46), (44) can be rearranged to obtain an expression for the external incremental displacement of each (p,Λ) top branch, in particular:

$${}^{p}_{\Lambda}[{}_{E}K^{*}_{D}(Y^{j}_{i})]({}^{p}_{\Lambda}\Delta Y^{j}_{i})_{E} = {}_{E}\lambda^{j}_{i}\,({}^{p}_{\Lambda}\Delta R_{D})_{E} + {}^{p}_{\Lambda}\Gamma_{E} \tag{47}$$

where

$$\begin{aligned} {}^{p}_{\Lambda}\Gamma_{E} = {} & ({}^{p}_{\Lambda}R^{j-1}_{D})_{E} - \left(\int_{{}^{p}_{\Lambda}V} [B^{*}]^{T}\, S\, dv\Big|_{Y^{j}_{i-1}} \right)_{E} - \\ & - {}^{p}_{\Lambda}([{}_{IE}K^{T}_{D}(Y^{j}_{i})][{}_{I}K_{D}(Y^{j}_{i})]^{-1}) \left(\int_{{}^{p}_{\Lambda}V} [B^{*}]^{T}\, S\, dv\Big|_{Y^{j}_{i-1}} \right)_{I} \end{aligned} \tag{48}$$

Assembling (48) at the next branch level, (Λ-1), we obtain the following intermediate expression

$${}^{p}_{\Lambda-1}[K^{*}_{D}(Y^{j}_{i})]({}^{p}_{\Lambda-1}\Delta Y^{j}_{i}) = {}_{E}\lambda^{j}_{i}\,({}^{p}_{\Lambda-1}\Delta R_{D}) + {}^{p}_{\Lambda-1}\Gamma \tag{49}$$

such that the (Λ-1) presubscript denotes the assembled coefficients and dependent variables. Restructuring into the (Λ-1)th level externals and internals yields the relation

$$ {}_{\Lambda-1}^{p}[{}_{E}K_{D}^{*}(\mathrm{Y}_{i}^{j})]({}_{\Lambda-1}^{p}\Delta\mathrm{Y}_{i}^{j})_{E} = {}_{E}\lambda_{i}^{j}({}_{\Lambda-1}^{p}\Delta\mathrm{R}_{D})_{E} + {}_{\Lambda-1}^{p}\Gamma_{E} \qquad (50) $$

Repeating the process recursively for each of the requisite branch levels, we obtain the following algorithmic relationships namely

$$ {}_{\Lambda-\ell}^{p}[K_{D}^{*}(\mathrm{Y}_{i}^{j})]({}_{\Lambda-\ell}^{p}\Delta\mathrm{Y}_{i}^{j}) = {}_{E}\lambda_{i}^{j}({}_{\Lambda-\ell}^{p}\Delta\mathrm{R}_{D}) + {}_{\Lambda-\ell}^{p}\Gamma \qquad (51) $$

$$ {}_{\Lambda-\ell}^{p}[{}_{E}K_{D}^{*}(\mathrm{Y}_{i}^{j})]({}_{\Lambda-\ell}^{p}\Delta\mathrm{Y}_{i}^{j})_{E} = {}_{E}\lambda_{i}^{j}({}_{\Lambda-\ell}^{p}\Delta\mathrm{R}_{D})_{E} + {}_{\Lambda-\ell}^{p}\Gamma_{E} \qquad (52) $$

where $\ell \varepsilon [0, \ldots, \Lambda\text{-}1]$. Once the root level is reached, we can solve for ${}_{E}\lambda_{i}^{j}$ and ${}_{1}^{p}\Delta \mathrm{Y}_{i}^{j}$. This is achieved by simultaneously satisfying the algorithms

$$ {}_{1}\Delta\mathrm{Y}_{i}^{j} = {}_{1}[K_{D}^{*}(\mathrm{Y}_{i}^{j})]^{-1}\{\lambda_{i}^{j}({}_{1}\Delta\mathrm{R}_{D}) + {}_{1}\Gamma\} \qquad (53) $$

and

$$ {}_{1}\mu\|{}_{1}\mathrm{Y}_{i}^{j}\|^{2} + (\lambda_{i}^{j}\|{}_{1}\mathrm{R}_{D}^{j}\|)^{2} = \|{}_{1}\mathrm{R}_{D}^{j}\|^{2} \qquad (54) $$

where μ is chosen from the inequality

$$ {}_{1}\mu = \underset{\forall(p,\ell)}{Max}\left\{\frac{\|{}_{\Lambda-\ell}^{p}\mathrm{Y}_{i}^{j}\|^{2}}{\|{}_{\Lambda-\ell}^{p}\mathrm{R}_{D}^{j}\|^{2}}\right\} \qquad (55) $$

2. <u>Second Pass</u>. The results obtained in the first pass are backsubstituted up the MLT to yield the intermediate and top level dependent fields. To obtain the corrected top level results, local iterations are necessary. The requisite algorithms are given by the expressions

$$ {}_{\Lambda}^{p}[{}_{I}K_{D}^{*}(\mathrm{Y}_{i}^{j})]({}_{\Lambda}^{p}\Delta\mathrm{Y}_{i}^{j})_{I} = {}_{\Lambda}^{p}\lambda_{i}^{j}({}_{\Lambda}^{p}\Delta\mathrm{R}_{D})_{I} + {}_{\Lambda}^{p}\Gamma_{I} \qquad (56) $$

and

$$ {}_{\Lambda}^{p}\mu\|{}_{\Lambda}^{p}\mathrm{Y}_{i}^{j}\|^{2} + ({}_{\Lambda}^{p}\lambda_{i}^{j}\|{}_{\Lambda}^{p}\mathrm{R}_{D}^{j}\|)^{2} = (\|{}_{\Lambda}^{p}\mathrm{R}_{D}^{j}\|)^{2} \qquad (57) $$

where the local (p, Λ) scaling parameters are defined by the inequities

$$ {}^{p}_{\Lambda}\mu = \underset{\forall (p, \Lambda, k)}{Max} \left\{ \frac{{}^{p}_{\Lambda}(\Delta Y^{j}_{i})_{k}}{\Delta Y_{a}} \right\} \tag{58} $$

such that ΔY_a defines the allowable individual degree of freedom excursion. The iteration process is continued until all top branch formulations are converged.

3. Third Pass. Once the second pass iteration process is complete, recursive passes up and down the tree can be employed to yield global convergence for the given load/ time step. All this is illustrated in Fig. 3.2.

The foregoing multipass algorithms can be performed either sequentially or in a partially-fully parallel format. Recalling Fig. 2.3, the sequential multilevel adaption possesses several branch levels, i.e. Fig. 3.3. When the processor is stationed at a particular substructure, it must perform several functions. These include:

1) Update local stiffness – for top level only;
2) Forward elimination – forward pass;
3) Back substitution – backward pass;
4) Matrix assembly – forward pass;
5) Newton Raphson iteration – top and root levels.

For multilevel subparallel applications, the flow of control is given in Fig. 3.4. As with the sequential case, the individual processors are also reassigned to the tasks devoted by 1)-5) above. This can be achieved in either a single or multiple processor assignment process.

Note the partition leap frogging described in the flow diagrams given in Figs. 3.3-3.5 is a result of the subparallel nature of the setup, i.e. P is less than the number of top branch partitions. Note, the leap frogging occurs both on a given branch as well as between succeeding levels. As the solution process moves down the tree, the number of partitions on succeeding levels reduces. In this context, while the upper levels may be subparallel, the lower ones particularly the root will be superparallel. For MLT wherein the work load of each level is uniformly distributed among the associated partitions, then the scheduling problem is fairly well defined. In particular as the process moves from sub to iso to super parallel levels, multiple processor assignment occurs. In such situations, scheduling difficulties are somewhat mitigated by the fact that work load uniformity regulates the problem. For problems involving localized material or boundary induced nonlinearity, the updating and iterative convergence will create complications for those partitions where multiple assignment is scheduled. Note such scheduling problems may be handled directly by the system compiler-operating system which directs the reassignment of processors to the ongoing tasks. Figure 3.6 illustrates the reassignment process associated with superparallel situations. Note once Ps the saturation level is reached, no more reassignment should be continued at that partition.

To close the discussion on the solver algorithm, we must address the issues of updating and convergence checks. Concerning updating, as noted earlier, three forms are possible. For highly nonlinear zones, continuous updating is typically necessary, i.e. for regions involving contact-impact, complex media (hyperelastic, inelastic, . .), large strains/rotations, and complex boundary interactions (follower forces . .). In regions which are primarily linear elastic but undergoing moderate rotations, the BFGS scheme can be employed to effect a quasi update. Traditionally the method has been employed in a global context. Here it can be employed at the local partition level.

In tree applications, several considerations are possible. These include:

1) Top level branches where direct element updating occurs;
2) Intermediate branch levels which are linked directly to top partitions undergoing updating;

3) Intermediate branch levels which are connected to a mix of up and non-updated top partitions and

4) Top and intermediate levels not undergoing updating due to association with essentially modestly nonlinear substructural zones.

The foregoing points to two issues, namely

1) when to update and,
2) how to update – full or quasi (BFGS).

Several investigators have considered the problem of when to update[10,14,16]. For instance in the work of Sheu et. al.[16], two criteria were employed namely

i) Incremental ratio tests of the normed out of balance loads or deflections and the appropriate reference states and
ii) Evaluation of successive variations in incremental energy states.

In fact a wide variety of possible flags can be established. Depending on problem type, these can be categorized into several basic tests

1) Measures of large rotation/small strain
2) Large strain/volumetric/shape distortion
3) Material nonlinearity and
4) Bounding induced effects.

Overall, the tests can be grouped into two classes of solution checks, i.e.:

1) Direct-performed at the top level and;
2) Indirect-performed primarily at intermediate and root levels.

The direct monitoring involves the evaluation of the conditioning of the localized dependent field variables. This includes both kinematic and stress measure namely[10]

- Invariants of the deformation gradient, Green Lagrange measure, Fingers tensor, Cauchy stress, . . etc.
- Rotations associated with designated target nodes

- Dilatation
- Element eigenvalues and so on.

Note since the stress and kinematic states define natural response, all system characteristics are covered. To assess conditioning, incremental variations can be ratio tested. For instance, if we consider the Green-Lagrange invariants I_1, I_2, I_3, the element level test would have the form

$$Tol > \frac{{}^{p}_{\Lambda}(\Delta I_k)^{j}_{i}\big|_e}{{}^{p}_{\Lambda}(\Delta I_k)^{j}_{i}\big|_e} \; k\varepsilon(1,3) \tag{59}$$

such that e designates the element number and ΔI_k the averaged kth invariant. Similar tests could be run on the other field variables. Overall such a test would be run for the top level branches only. This would gage the need to perform element level updating.

The indirect tests are performed to evaluate the state of the intermediate branches and root. Since a typical intermediate branch may be composed of various updated and nonupdated top level partitions, its update is contingent on the degree to which local effects penetrate to lower levels of the tree. Since it is less meaningful to define the kinematic and stress fields at such tree levels, we resort to gross/norm states. This can be achieved for both the global and intermediate branches. The tests consist of ratios of incremental variations in out-of-balance load, deflection, energy, inelastic growth, gross partition rotation and volume/area change. At the intermediate partition level we have

i) Out-of-balance load

$$Tol > \frac{\|{}^{p}_{\ell}\Delta F^{j}_{i}\|}{\|{}^{p}_{\ell}\Delta R^{j}_{D}\|} \tag{60}$$

ii) Deflection

$$Tol > \frac{\| {}^{p}_{\ell}\Delta \mathrm{Y}^{j}_{i} \|}{\| {}^{p}_{\ell}\Delta \mathrm{Y}^{j}_{0} \|} \tag{61}$$

iii) Energy

$$Tol > \frac{{}^{p}_{\ell}E^{j}_{i}}{{}^{p}_{\ell}E^{j}_{1}} \tag{62}$$

iv) Inelastic growth

$$Tol > \frac{{}^{p}_{\ell}A^{j}_{pi}}{{}^{p}_{\ell}A^{j}_{p1}} \tag{63}$$

v) Gross rotation

$$Tol > \frac{{}^{p}_{\ell}\Delta\omega^{j}_{i}}{{}^{p}_{\ell}\Delta\omega^{j}_{1}} \tag{64}$$

where $\| \, \|$ defines the Euclidean norm, and

$${}^{p}_{\ell}\Delta \mathrm{F}^{j}_{i} = {}^{p}_{\Lambda}\mathrm{R}^{j}_{D} - \int_{{}^{p}_{\Lambda}V} [B^{*}]^{T} \mathrm{S} \, dv \Big|_{\mathrm{Y}^{j}_{i-1}} \tag{65}$$

$${}^{p}_{\Lambda}E^{j}_{i} = \frac{1}{2} (\Delta \mathrm{Y}^{j}_{i})^{T} \int_{{}^{p}_{\Lambda}V} \{[B^{*}]^{T} \mathrm{S}\Big|_{\mathrm{Y}^{j}_{i-1}} - [B^{*}]^{T} \mathrm{S}\Big|_{\mathrm{Y}^{j}_{0}}\} \, dv \tag{66}$$

For the gross problem and root levels, two tests are possible namely

1) Those restricted to strictly the root variables or

2) Those involving all or various of the branch level externals as one telescopes from the root to the top of the tree.

Such tests are exemplified by the following expressions:

1. Root only:

- Out of balance loads

$$Tol > \frac{\|{}_1\Delta \mathbf{F}_i^j\|}{\|{}_1\Delta \mathbf{R}_D^j\|} \tag{67}$$

- Deflection

$$Tol > \frac{\|{}_1\Delta \mathbf{Y}_i^j\|}{\|{}_1\Delta \mathbf{Y}_0^j\|} \tag{68}$$

- Energy

$$Tol > \frac{{}_1E_i^j}{{}_1E_0^j} \tag{69}$$

2. Telescoping (all levels)

- Out of balance loads

$$Tol > \frac{\sum_{\ell}^{\Lambda} \sum_{p} \|{}_{\ell}^{p}\Delta \mathbf{F}_i^j\|}{\sum_{\ell}^{\Lambda} \sum_{p} \|{}_{\ell}^{p}\Delta R_D^j\|} \tag{70}$$

- Deflection

$$Tol > \frac{\sum_{\ell}^{\Lambda} \sum_{p} \|{}_{\ell}^{p}\Delta \mathbf{Y}_i^j\|}{\sum_{\ell}^{\Lambda} \sum_{p} \|{}_{\ell}^{p}\Delta \mathbf{Y}_1^j\|} \tag{71}$$

- Energy

$$Tol > \frac{\sum_{\ell}^{\Lambda} \sum_{p} {}^{p}_{\ell}E^{j}_{i}}{\sum_{\ell}^{\Lambda} \sum_{p} {}^{p}_{\ell}E^{j}_{0}} \tag{72}$$

Contingent on modelling needs, one or more of the foregoing tests can be implemented.

If the foregoing tests remain below a specified tolerance limit, then a modified version of the BFGS scheme can be employed to update local partition level stiffness. This enables reasonable iterative convergence while bypassing the need for an inverse at the given substructure. Adapting the scheme to a partition level application, it follows that the updated local stiffness takes the form

$${}_{\Lambda-\ell}^{\ p}[K^{*}_{D}(Y^{j}_{i})]_{i} = [\Phi_{i}]^{T}\ {}_{\Lambda-\ell}^{\ p}[K^{*}_{D}(Y^{j}_{i})]_{i} = [\Phi_{i}] \tag{73}$$

where

$$[\Phi_{i}]^{T} = [I] + V_{i}(W_{i})^{T} \tag{74}$$

The vectors V_i and W_i are calculated from the known nodal point forces and displacement using the relations

$$V_{i} = -Y_{i} - \left[\frac{(\delta_{i})^{T} Y_{i}}{(\delta_{i})^{T}\ {}_{\Lambda-\ell}^{\ p}[K^{*}_{D}(Y^{j}_{i})]\delta_{i}} \right]^{\frac{1}{2}} {}_{\Lambda-\ell}^{\ p}[K^{*}_{D}(Y^{j}_{i})]\delta_{i} \tag{75}$$

and

$$W_{i} = \frac{1}{(\delta_{i})^{T} Y_{i}}\delta_{i} \tag{76}$$

where

$$\delta_{i} = Y_{i} - Y_{i-1} \tag{77}$$

$$\gamma_i = F_i - F_{i-1} \tag{78}$$

Due to the generality of the formulation described by (73)-(78), BFGS type updating can be applied to any level in the MLT and any partition of that level. Note, near bifurcations and turning points, buckling, the full update should be employed to yield the proper transitional characteristics. This also holds true in inelastic/history-dependent processes wherein the proper updating is needed to trace the plastic event. As the calculations proceed down the MLT, it is possible that an isolated local zone can be handled via BFGS updating at intermediate levels. This would greatly reduce the work load.

IV. Automated Substructuring

Overall to automate substructuring four basic steps are required. These include:

i) Symmetry checks on both geometric and nodal topology

ii) Establish multilevel partitioning

iii) Bandwidth minimize each individual substructural component and

iv) Optimize memory and work load by selecting best number of levels and decomposition per level.

Often times structure have localized symmetries which, due to loading and boundary conditions, cannot be taken advantage of in a global sense. Such symmetries can be at the root – intermediate – top levels. To determine such attributes, an automated symmetry check must be used to help optimize the substructuring. While visually such properties are easily spotted, from a numerical-analytical point of view, such is not the case. Here we adopt the following multistep procedure namely

i) Find CG

ii) Determine inertia tensor (I) relative to CG

iii) Find principal coordinates and components of (I), i.e. (I_p)

iv) Establish nature of symmetry based on

- properties of (I_R)
- coordinate check about principal axes

v) Define nature of partitioning based on symmetry type and

vi) Perform such checks recursively at succeeding levels.

From an FE point of view, the determination of the global/substructural CG can be established by employing the element properties. Specifically at the ℓth level and pth partition. The associated elements define the following expression

$$ {}^{p}_{\ell}\left(\overline{X_1};\overline{X_2};\overline{X_3}\right) = \frac{1}{{}^{p}_{\ell}V}\sum_{e}\int (X_1;X_2;X_3)\,dv \tag{79}$$

where

$$ {}^{p}_{\ell}V = \sum_{e} {}^{p}_{\ell}V_e \tag{80}$$

such that

${}^{p}_{\ell}V_e$ – element volume

${}^{p}_{\ell}V$ – net volume of (p, ℓ) pair

Based on the CG location, the inertia tensor takes the form

$$ {}^{p}_{\ell}[I_{CG}] = \sum_{e}\int_{{}^{p}_{\ell}V_e}[i_{CG}]\,dv \tag{81}$$

such that $\int(\ )\,dv$ defines typical Gauss quadrature and

$$[i_{CG}] = \begin{bmatrix} X_2^2+X_3^2 & -X_1X_2 & -X_2X_3 \\ -X_2X_1 & X_1^2+X_3^2 & -X_2X_3 \\ -X_3X_1 & -X_3X_2 & X_1^2+X_2^2 \end{bmatrix} \tag{82}$$

Since $[I_{CG}]$ is a 2nd order tensor, its principal orientation and properties can be established in the usual manner, this yields

$$[I_p] = \begin{bmatrix} I_{p1} & 0 & 0 \\ 0 & I_{p2} & 0 \\ 0 & 0 & I_{p3} \end{bmatrix} \tag{83}$$

As noted earlier depending on the makeup of $[I_p]$, various symmetries can be identified in particular

i) If two components are repeated, there is planar symmetry

ii) If three are repeated, there is 3-D symmetry.

If no repeated roots exist, then potential symmetries about each principal can be ascertained via a sum check. Specifically, for nodes symmetrically placed relative to the CG, the following identies hold

$$\begin{gathered} \sum_i X_{1i} < Tol. \quad \text{(1 axis symmetry)} \\ \sum_i X_{2i} < Tol. \quad \text{(2 axis symmetry)} \\ \sum_i X_{3i} < Tol. \quad \text{(3 axis symmetry)} \end{gathered} \tag{84}$$

where the tolerance depends on user expectations.

In the case that repeated roots are found, then the foregoing sum tests are automatically satisfied about any Cartesian coordinate system, i.e. arbitrary planar orientation for two repeated roots and arbitrary 3-D orientation for three roots. For the case of $I_{p1} = I_{p2}$ (two roots), to find the symmetry axes we search for the max points, i.e. outer bounds of the object. These will lie in a skew symmetric format. In this context we search for

$$X_{1M} = Max\{X_i\} \, i \varepsilon [1, {}^{p}_{\ell}N] \tag{85}$$

where M defines the outlier node, i.e. (X_{1M}, X_{2M}, X_{3M}). Once obtained, the appropriate transformations can be established. For 2-D/3-D situations these are represented by the relations

i) 2-D;

$$a = TAN^{-1}(X_{2M}/X_{1M}) \tag{86}$$

$$\begin{Bmatrix} X_1 \\ X_1 \end{Bmatrix}^* = \begin{bmatrix} COS(a) & SIN(a) \\ -SIN(a) & COS(a) \end{bmatrix} \begin{Bmatrix} X_1 \\ X_2 \end{Bmatrix} \tag{87}$$

ii) 3-D

$$a_1 = TAN^{-1}(X_{2M}/X_{1M}) \tag{88}$$

$$a_2 = TAN^{-1}(X_{3M}/X_{1M}) \tag{89}$$

$$\begin{Bmatrix} X_1 \\ X_2 \\ X_3 \end{Bmatrix}^* = [T(a_1, a_2)] \begin{Bmatrix} X_1 \\ X_2 \\ X_3 \end{Bmatrix} \tag{90}$$

such that ( )* denotes the symmetry axis system. Note for repeated principal inertias, a multitude of such axis systems are possible.

Once the nature of the available symmetry is established, the type of partitioning must be chosen. In the context of the HPT recently developed by Padovan et.al.[18,19], it follows that optimal results require the appropriate substructural arrangement. For instance, considering the 2-D case, there is either 1 or 2 axes of symmetry. These respectively require κ_1 and $\kappa_1 \kappa_2$ partitioning such that κ_i are even. In this way, in addition to yielding the proper choice of external-internal load balancing, maximum use can be made of cyclic substructural generation for linear elements. Such cloning can yield respectively at least a two and four fold speedup for 1 and 2 axes of symmetry. This can be achieved at each level with symmetric partitions. Note such enhancements are in addition to the benefits of the MLT. For the 3-D case, κ_1, $\kappa_1\ \kappa_2$ and $\kappa_1\ \kappa_2\ \kappa_3$ partitions must be established for 1, 2 and 3 axisymmetric states. Again, for linear substructure, such symmetries enable at least a 2, 4 or 8 fold speedup via cloning at each possible level.

Note beyond defining the cloning and substructuring characterization, symmetry properties can be employed to define multiply seeding points from which to initiate simultaneous partitioning. Two procedures are possible, i.e.

i) The consecutive node attachment (CNA) scheme of Wilson and Farhat[17] and

ii) The direct element connect filling (DECF) scheme

In the Wilson-Farhat scheme, substructuring is automated into a series of recursive steps involving:

1) Bandwidth minimization
2) Connection of elements attached to nodes of ascending order
3) Partition definition completed when appropriate number of attachments achieved
4) Procedure repeated with already defined substructure substracted from process: at this point bandwidth minimization preapplied to the reduced model.

Due to the recursive use of bandwidth minimization, the current procedure is limited to starting from the initial node number. This follows from the fact that due to the goal of reducing the skyline height, it is possible to have noncontiguous node numbering. Hence, potentially nonconnected, i.e. discontinuous partitions may develop during the agglomeration process. This is especially true if starting points further up the node count were attempted.

To bypass the foregoing difficulty, a DECF scheme will be employed. Here the steps include

1) Establish multiple starting points; Fig. 4.1
2) Using element connectivity map, adjoin elements directly connected to starter nodes, Fig. 4.2; proceed simultaneously at each initiation site
3) Continue process in successive waves of attachment; Figs. 4.3 and 4.4
4) In symmetrical problems, attachments are preformed in balanced pairs; Fig. 4.2
5) Fill process continued until requisite number of elements adjoined – contingent on number of partitions defined for level.

The starting points are chosen via similar criteria to traditional bandwidth minimizer, (Cuthill-McKee[24], Gibbs-Poole-Stockmeyer[25]), i.e.:

1) Determine points with minimum connectivity – outside corners, kinks, edges;
2) Due to symmetry, several starting points may be employed simultaneously
   - 2-D/2 axis – 4 points/1 axis – 2 points
   - 3-D/3 axis – 8 points/2 axis – 4 points
3) Beyond providing seeding points, symmetry can be used to define constraint surfaces which control the growth of partitions during the agglomeration process, in particular
   i. 2-D problems, Fig. 4.5
      - 1 axis – single axis disection
      - 2 axis – two axis
   ii. 3-D problems, Fig. 4.5
      - 1 axis/prismatic – single planar dissection
      - 2 axis/prismatic – two plane
      - 3 axis – three plane

As noted earlier, the partitioning process can be reapplied at each succeeding level. In such applications global-root level symmetrics may give way to symmetric or asymmetric intermediate levels, Fig. 4.6. If multiple levels of symmetry are noted, Fig. 4.6, cloning where possible can lead to significant speedup.

Once each of the various substructure are defined, the associated external-internal node number count must be minimized. The traditional bandwidth minimizers[26] are formulated to minimize the skyline height associated with individual nodes. In this context the nodes can be shifted to any location on the diagonal. For substructural problems, the external nodes must be positioned at the base or top of the diagonal. This can cause a suboptimal arrangement between the externals and internals. Noting Fig. 4.7, minimizing strictly the internals yields a skyline structure with large coupling side bands. For instance considering the square region discussed earlier, the overall computational effort for a typical column elimination operation is given by the expression, Fig. 4.8

$$\text{Single Column Effort} \sim E_1 + E_{2/3} + E_4 \tag{91}$$

such that

E_1 ~ Effort in purely internal block

$E_{2/3}$ ~ Side band effort

E_4 ~ External block

When averaged overall rows, we obtain

$$\text{Net Effort} \sim E_1 + E_{2/3} + E_4 \tag{92}$$

where

$$E_1 \sim \frac{1}{2}(N-2)^2 N^4 \tag{93}$$

$$E_{2/3} \sim 4N(N-1)(N-2)^2 \tag{94}$$

$$E_4 \sim 4\,(N-1)^2\,(N-2)^2 \tag{95}$$

Asymptotically, as N → (large), it follows that

$$\text{Net Effort} \rightarrow \frac{9}{2} N^4 \tag{96}$$

In this context, substructuring yields a 450% increase in work load over the straight solution.

The root cause of the increase lies in the fact that the bandwidth minimization did not account for the fact that the optimal treatment of the problem involves the handling of the multipoint constraint defined by the externals. In view of this, instead of starting the search for the minimum skyline profile from single least connected internal nodes, the substructured version could alternatively start from the externals. Based on this, the operational steps of the minimizer become

1. Starter points involve all externals
2. Determine all directly connected internal nodes from connectivity map – this generates a "shell" of nodes
3. Adjoin shell directly to starter nodes
4. Determine next layer of direct connected internal nodes
5. Adjoin to previous shell
6. Continue process in successive shells of attachment, Fig. 4.9.

Overall the procedure generates an inwardly spiralling numbering pattern. To illustrate the improved computational efficiency of such a skyline, we reconsider the square patch. Noting Fig. 4.10, the spirally count causes the succeeding attachment shells to have reduced bandwidths, i.e. going from 4N-4 for the external first shell to 4N-4-8(ℓ-1) for the ℓ^{th} internal shell. The population of nodes associated with such skyline heights reduce as one moves inward into the substructure. Namely, from (4N-4) in the first shell to 4N-4-8(ℓ-1) for the ℓ^{th} shell. This is illustrated in Fig. 4.11.

To determine the work effort associated with the foregoing numbering scheme, it is noted that unlike the traditional approach, no large coupling side bands are encountered. The computational effort can be cast in the form

$$\text{Net Effort} \sim \frac{1}{2}\sum_{\ell=1}^{N/2}(4N+4-8\ell)^3 - \frac{1}{2}(4N-4)^3, \tag{97}$$

which for arbitrary N yields

$$\text{Net Effort} \sim 4N^2(N^2-2) - \frac{1}{2}(4N-4)^3 \tag{98}$$

From an asymptotic point of view, the net computation effort associated with the succeeding attachment shell numbering pattern takes the form

$$\text{Net Effort} \sim 4N^4 \tag{99}$$

This represents a potential $12^{+}\%$ improvement over the classical approach, i.e. Eq. (91). Note such improvements are strongly dependent on geometry, element type and element density within a given region. Hence, care must be exercised in its use.

As has been seen by Padovan and Gute[19], the choice of the optimal hierarchy of levels and partitions is highly problem dependent. In this context to determine the best arrangement an iterative strategy must be implemented. Overall it consists of the following steps

1) Recursively choose number of levels
2) Partition each level through various permutations
3) Estimate work load for bandwidth minimized partitions
4) Compare work loads of various level/partition arrangements to achieve optimal results.

V. Discussion – Benchmarking

In the previous sections, a hierarchically scaled multilevel nonlinear equation solver was developed. Overall the procedure provides for the possibility of assigning separate constraints to each partition throughout all the levels of the MLT. The focus of the benchmarking of the scheme will be two fold, namely to illustrate

1) The numerical performance of the nonlinear solver and

2) The memory-computational savings afforded by the Tree scheme.

Figure 5.1 illustrates the arch type truss structure used to evaluate the numerical/iterative performance. The arch was chosen since it possesses highly nonlinear force deflection attributes. These are illustrated in Fig. 5.2. Depending on whether compressive or tensile loading is applied, either softening or hardening behavior is excited. The associated deflected shapes are given in Fig. 5.3. Such nonlinear characteristics yield varying numerical sensitivities. In this context, Tables 5.1-5.3 illustrate the effects of initial step size, truss geometry and loading direction on the numerical convergence. Various types of iterative update schemes were evaluated. These include

Root	Second Level
Full	Full
BFGS	Full
BFGS	No update
No update	Full
No update	No update
Automatic	Automatic

For the automatic case, updating and constraint control is triggered by the appropriate criteria. Here due to the large deformation/rotation behavior associated with the truss, a rotation check can be employed. Once the requisite change in rotation is accumulated, the local partition stiffness is updated. Here both full and BFGS type schemes were tested.

As can be seen from Tables 5.1 & 5.2, the full and automated update schemes yielded essentially the same results. In the case of the automated scheme, significantly less updating was required. For the cases noted, the automatic scheme yielded update savings of 30-40%. This was achieved with no changes recorded in comparisons with the global scheme.

When localized constraints are introduced for each partition, convergence was obtained for all the load ranges considered. Note care must be taken when employing such a robust approach especially for problems with multiple solution states. In such situations, the load readjustment generated by the individual constraints may cause movement to different loading paths thereby leading to an alternative solution state. This can be prevented by tightening up on the admissible dependent field excursions generated during successive iterations.

For large loads, generally an incremental application is necessary. The arches sensitivity to such a loading approach is depicted in Table 5.3. As can be seen, as the increment is decreased, the iterative requirements become essentially the same for all the schemes. Conversely, in the case of the largest increment, only the full and automated updating schemes converge. Here the automated scheme requires 37% less updating. This is a result of the possibility of intermittent reformation at the various partition levels. This of course is highly dependent on the geometry/connectivity as well as the loading history of the model treated.

To illustrate the parallel attributes of the nonlinear MLT scheme, we will consider the large scale truss problem depicted in Fig. 5.4. The model was substructured into a three level tree consisting of 1, $(3)^2$ and $(9)^2$, 1st, 2nd and 3rd level square partitions. The problem was tested on the NASA Lewis Alliant system with eight available processors. This enabled a partially parallel application wherein multiple processor assignment was employed. To simplify the programming, all the processors were assigned to a given substructure. At the top/third level, the effort includes both updating, assembly condensation and partial assembly for the next level. Once completed, all the processors are reassigned to the next partition and so on in a succession of assignment steps. The next lower branch level is performed in a level by level format down the MLT. Once the root is handled, the condensation step is complete. Next the process is reversed level by level for the backsubstitution step. Two forms of testing were considered, i.e. single and multiple processor assignment. This enabled the evaluation of potential system contention problems.

For the given problem and style of Fortran programming, 25 edge nodes defined the break even point of the chosen MLT. For larger problems, significant improvements can be obtained. For instance at 41 and 56 per side, the single assignment (purely sequential) scheme yielded speedups of 2. and 3. relative to the global scheme. Employing 8 processors, the 41 and 56 edge noded truss yielded speedup factors of 11.4 and 15.5 respectively. In contrast, for the 25 edge noded case, a factor 6.02 was recorded. One would have expected a (two/three) to one difference between the 25 and (41/56) noded examples. The variation is a result of the increasing contention which occurs as problem size increases. The increased number of memory fetches required by larger partitions causes increased traffic control problems in the system buss thereby delaying computations. In this context, for architecture employing multiple processor assignment, smaller partitions are more advantages. This is clearly illustrated in Table 5.4 which describes the contention problem as a function of size and number of processors: this is of course machine dependent.

Next we shall consider the problem of how to select the proper number of partitions and levels to yield optimal results. As an initial demonstration, we shall consider a square mesh defined by 2D four node quad elements. Employing either model fill or the Wilson-Farhat[17] scheme, the region is dissected into 2, 3, . . ., 9, . . partitions,i.e. Fig. 5.5. Noting Fig. 5.5, optimal speedup results are obtained for $(\kappa)^2$ type arrangements, i.e. symmetrical square partitions. Applying the model fill scheme recursively on a level by level basis, the same $(\kappa)^2$ type decomposition yields the most optimal results at all the various branch levels, i.e. top, intermediate and root. Based on such an approach, Tables 5.5 and 5.6 illustrate the effort and memory reduction potential of the tree scheme for several sized problems. As can be seen, many order of magnitude reductions can be obtained.

In the case of multiply connected regions, the use of the CNA[17] scheme yields nonoptimal partitions. This follows from the fact that bandwidth minimization often leads to noncontinguous node numbering. During the agglomeration process used to define the partitions, potential nonconnected or disjoint substructure are possible. Consider the meshes depicted in Figs. 5.6 and 5.7. The CNA[17] procedure tends to lead to distended substructural regions depicted in Fig. 5.8. In contrast, the DECF scheme yields more regular subregions, i.e. Fig. 5.9. Here a variety of seeding points were employed to start and continue the process.

To reduce the periphery of the various partitions, salient control was applied on the second pass. Overall the procedure consists of checking boundary attached elements for the level of connectivity within their partition. Specifically, the number of adjacencies bordering each edge of a given element are determined. By employing the connectivity information, the number of elements bordering the candidate element are determined in each of the $\perp$ and parallel directions to the interconnect boundary. If the noted element has less than a preset number of adjacencies along one direction, then it is adjoined to the appropriate neighboring substructure with the requisite connectivity. The results of such a process is illustrated in Fig. 5.9. Had model symmetry been employed, the results depicted in Figs. (4.1-4) would have been achieved. This of course is highly model dependent.

To establish the improvement potential of the DECF based partitioning scheme, both the perimeter to internal node ratio and individual substructural-net computational effort are determined. These are used to quantify the comparison with the CNA scheme of Farhat and Wilson[17]. In particular, recalling the models defined in Figs. 5.6 and 5.7, Figs. 5.10-5.14 illustrate the topological arrangements for varying levels of automated substructuring, i.e. 4, 9, 16, . . partitions.

To provide a consistent basis for comparison, the reverse Cuthill-McKee scheme[24] is employed to locate the first, i.e. seed note on the diagonal. The procedure is reapplied to each succeeding reduced model generated by subtracting previously defined substructure from the original formulation. Overall the procedure lead to the effort comparisons described in Table 5.7. As can be seen, the DECF method consistently out performed the CNF. Significant factors of improvement were noted over a wide range of partitions choices.

An unexpected benefit of the Tree scheme arises for problems involving repeated-multilevel symmetries, i.e. Fig. 5-15. For such structure, the Tree reduces the burden of assembly and condensation in linear partitions. In particular, if a problem such as that depicted in Fig. 5.15 remains linear for several steps, the unit cell illustrated is all that needs to be generated and translated-rotated to yield the rest of the substructure. Hence the work load at the top level would be essentially that of a parallelized setup. The same would be true of each lower level. Hence the condensation speedup and memory reduction would be that associated with a single assignment parallelized Tree, i.e. a separate processor for each partition. During the backsubstitution phase, a similar procedure applies. Overall depending on problem topology, significant order of magnitude speedup could be achieved in large scale problems.

Acknowledgement

The first author thanks Christos Chamis of NASA Lewis for his helpful advice and lively discussions.

References

1. Zienkiewicz, O.C., The Finite Element Method, 3rd Ed. McGraw-Hill Book Co., London, England.

2. Bathe, K.J., Finite Element Procedures in Engineering Analysis, Prentice-Hall, Inc., Englewood Cliffs, N.J.

3. Cook, R.D., Malkus, D.S., and Plesha, M.E., Concepts and Applications of Finite Element Analysis, 3rd Ed., John Wiley & Sons, Inc., New York.

4. Wempner, G.A., Discrete approximations related to nonlinear theories of solids, Int. Jr. Solids and Structures, Vol. 7, pp. 1581, 1971.

5. Riks, E., An incremental approach to the solution of snapping and buckling problems, Inter. Jr. of Solids and Structure, Vol. 15, pp. 529, 1979.

6. Crisfield, M.A., A faster modified Newton-Raphson iteration, Comp. Meth. Appl. Mech. Engrg., Vol. 20, pp. 267, 1979.

7. Crisfield, M.A., A fast incremental/iterative procedure that handles snap through, Computers and Structures, Vol. 13, pp. 55, 1981.

8. Padovan, J., and Tovichakchaikul, S., Self adaptive prediction-corrector algorithms for static nonlinear structure analysis, Computers and Structures, Vol. 15, pp. 365, 1982.

9. Padovan, J., and Arechaga, T., Formal convergence characteristics of elliptically constrained incremental Newton Raphson algorithm, Inter. Jr. of Engrg. Sci., Vol. 20, pp. 1077, 1982.

10. Padovan, J., and Moscarello, R., Locally bound constrained Newton Raphson solution algorithms, Computers and Structures, Vol. 23, pp. 181, 1986.

11. Bathe, K.J., and Cimento, Some practical problem for the solution of nonlinear finite element equations, Comput. Meth. Appl. Mech. Engrg., Vol. 22, pp. 59, 1983.

12. Padovan, J., and Lackney, J., Constrained hierarchical least square non-linear equation solvers, Computers and Structures, Vol. 23, pp. 251, 1986.

13. Abel, J.F., and Han, T.Y., Adaptive substructuring techniques in Elasto-Plastic FE analysis, Computers and Structures, Vol. 20, pp. 181, 1985.

14. Padovan, J., Multiply constrained partitioned nonlinear equation solvers, Computational Mechanics, Vol. 3, pp. 255, 1988.

15. Padovan, J., and Krishna, L., On the convergence of block constrained nonlinear equation solvers, Inter. Jr. of Computer Math., Vol. 29, pp. 169, 1989.

16. DeRoeck, G., VanLaethem, M., and Shen Chyi-Horng, Multi-level substructuring in the elastic-plastic domain, Computers and Structures, Vol. 31, pp. 757, 1989.

17. Farhat, C., and Wilson, E., A parallel active column equation solver, Computers and Structures, Vol. 18, pp. 289, 1988.

18. Padovan, J., Gute, D., and Johnson, K., Hierarchical poly tree computer architects defined by computational multidisciplinary mechanics, Computers and Structures, Vol. 32, pp. 1133, 1989.

19. Padovan, J., and Gute, D., Multilevel hierarchical poly tree computer architectures, Computers and Structures, in press.

20. Gute, D., and Padovan, J., Hierarchical poly tree configurations for the solution of dynamically refined finite element models, Computers and Structures.

21. Matthies, H., and Strang, G., The solution of nonlinear finite element equations, Int. Jr. Numer. Meth. Engrg., Vol. 14, pp. 1612, 1979.

22. Fung, Y.C., Foundations of Solid Mechanics, Prentice-Hall, Englewood Cliffs, N.J.

23. Hughes, T.J.R., The Finite Element Method, Prentice-Hall, Englewood Cliffs, N.J.

24. Cuthill, E., and McKee, J., Reducing the bandwidth of space symmetric matrics, Proc. ACM National Conference, pp. 157-172, 1989.

25. Gibbs, N.E., Poole, W.G., and Stockmeyer, P.K., An algorithm for reducing the bandwidth and profile of a sparse matrix, SIAM Jr. Numer. Anal., Vol. 13, pp. 236, 1976.

26. George, A., and Lin, J., Computer Solution of Large Sparse Positive Define System, Prentice-Hall, Englewood Cliffs, N.J.

Figure and Table Captions

Fig. No.	Caption
2.1	Successive iterations of elliptically constrained Newton Raphson Scheme
2.2	2-D square and 3-D cubic regions
2.3	Sequential and partially parallel flow of control: single processor assignment
2.4	Partially parallel with multiprocessor reassignment
3.1	Multilevel tree defining linkages between successive substructural levels
3.2	Flow of control of HPT
3.3	Sequential flow of control of multilevel tree
3.4	Subparallel single assignment processor leap frogging: Multilevel
3.5	Subparallel multiple assignment processor leap frogging: Multilevel
3.6	Super/iso parallel processor reassignment process
4.1	Multiple starting points for 2-D 2 axis of symmetry model
4.2	Balanced multiple initiation point agglomeration
4.3	Successive waves of attachment: 2 axes of symmetry
4.4	Successive waves of element attachment
4.5	Potential lines and planes of symmetry in 2-D and 3-D defining constraint surfaces
4.6	Multiple levels of substructural symmetry
4.7	Typical skyline of substructured component with minimization employed or internal variables
4.8	Work load associated with elimination of ith column: Internally bandwidth minimized substructured partition
4.9	Successive shell attachment scheme for node numbering substructured components
4.10	Spiralling node connectivity generated by successive shell attachment scheme
4.11	Skyline of spiralling node connectivity model

5.1 Arch truss simulation and two level tree

5.2 Softening and hardening load deflection characteristics of truss: $\theta = 10°$

5.3 Compressive and tensile deflected shapes of truss (Fig. 5.1)

5.4 Large scale truss structure used to test parallel version of nonlinear multilevel tree

5.5 Effect of number of partition on speedup: two level tree: partition selection via Wilson-Farhat Scheme[17]: 8 processors

5.6 Multiple connected mesh: several cells: Model 1

5.7 Multiple connected mesh: multiple cells: Model 2

5.8 Mesh partitioning: Wilson-Farhat Scheme[17]

5.9 Model filling mesh partitioning
A – CCW fill – CCW speeding
B – CCW fill – CC speeding
C – Balanced fill and speeding

5.10 Partitioning of Model 1: 4 partitions

5.11 Partitioning of Model 1: 9 partitions

5.12 Partitioning of Model 2: 4 partitions

5.13 Partitioning of Model 2: 9 partitions

5.14 Partitioning of Model 2: 16 partitions

5.15 Multilevel symmetries

Table No. Caption

5.1 Load step sensitivity of various update schemes: arch 60°; compressive loading (tolerance $10^{-6}/10^{-3}$)

5.2 Load step sensitivity of various update schemes: arch 10° tensile load (tolerance $10^{-6}/10^{-3}$)

5.3 Multiload step sensitivity of various update schemes: arch 60° (No. of increments/total iterations)

5.4 Efficiency (%) of Alliant Parallel Processor

5.5 Effects of problem size on speedup and tree morphology: sequential case

5.6 Effects of problem size on speedup and tree morphology: parallel case

5.7 Comparison of DECF and CNF schemes of partitioning

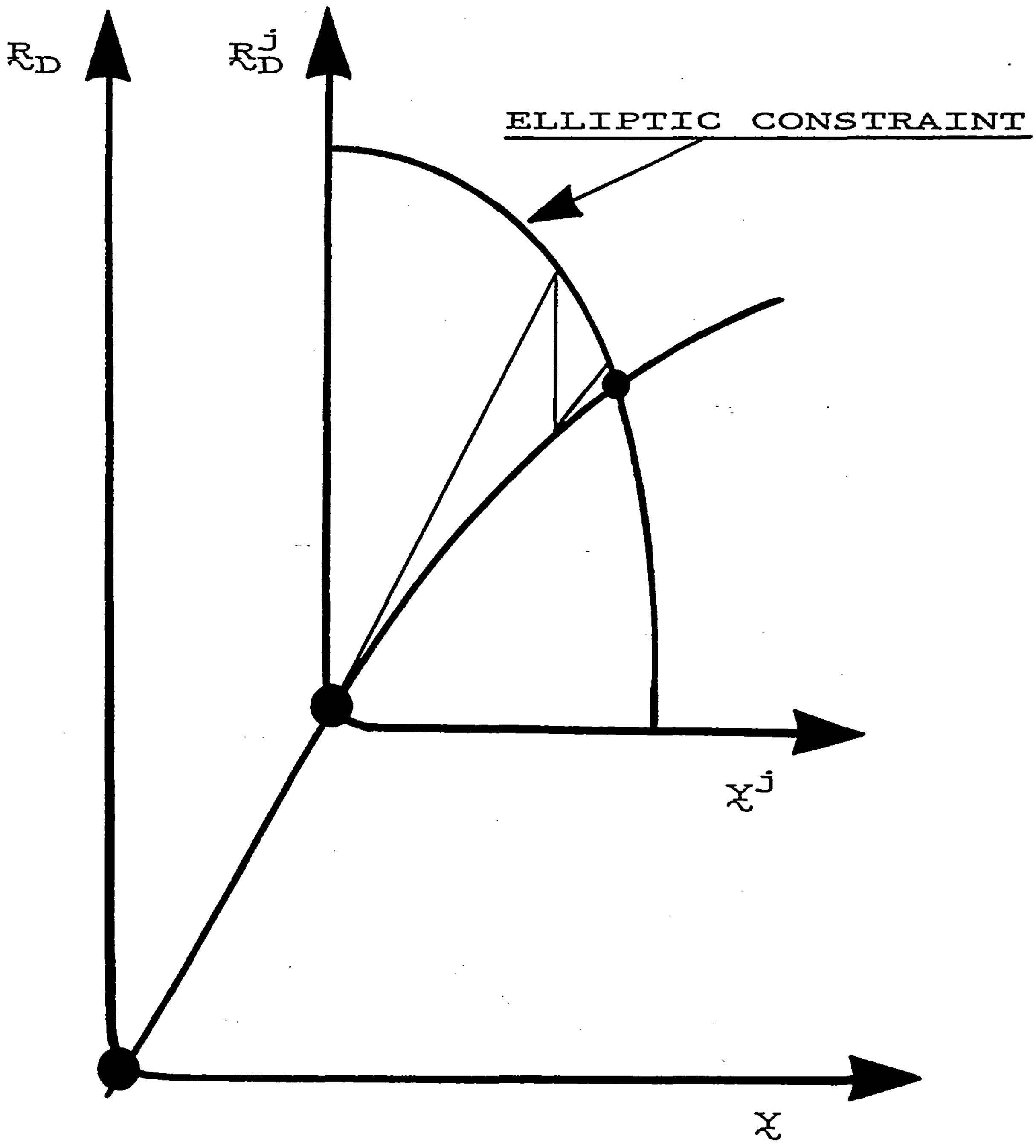
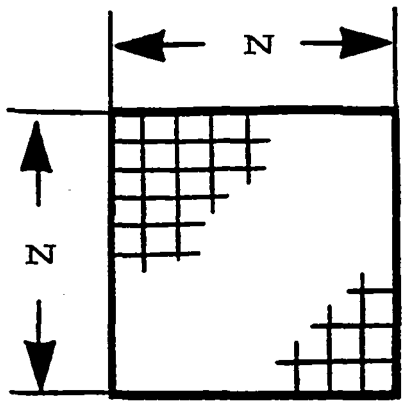
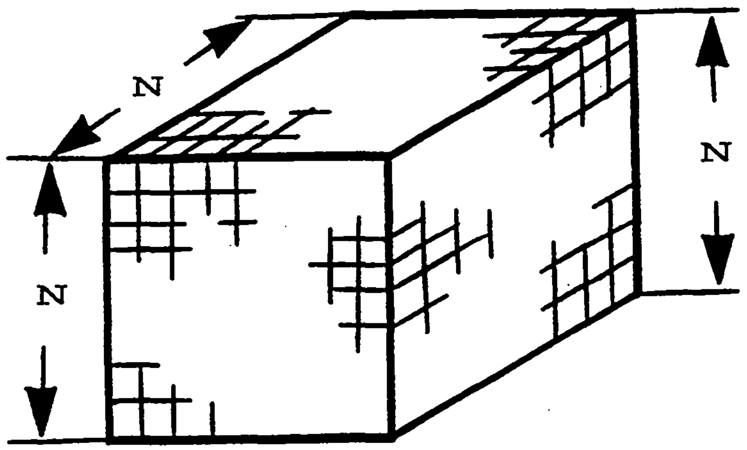


FIGURE 2.1



2-D N-N Square Regions



3-D N-N-N Cubic Regions

FIGURE 2.2

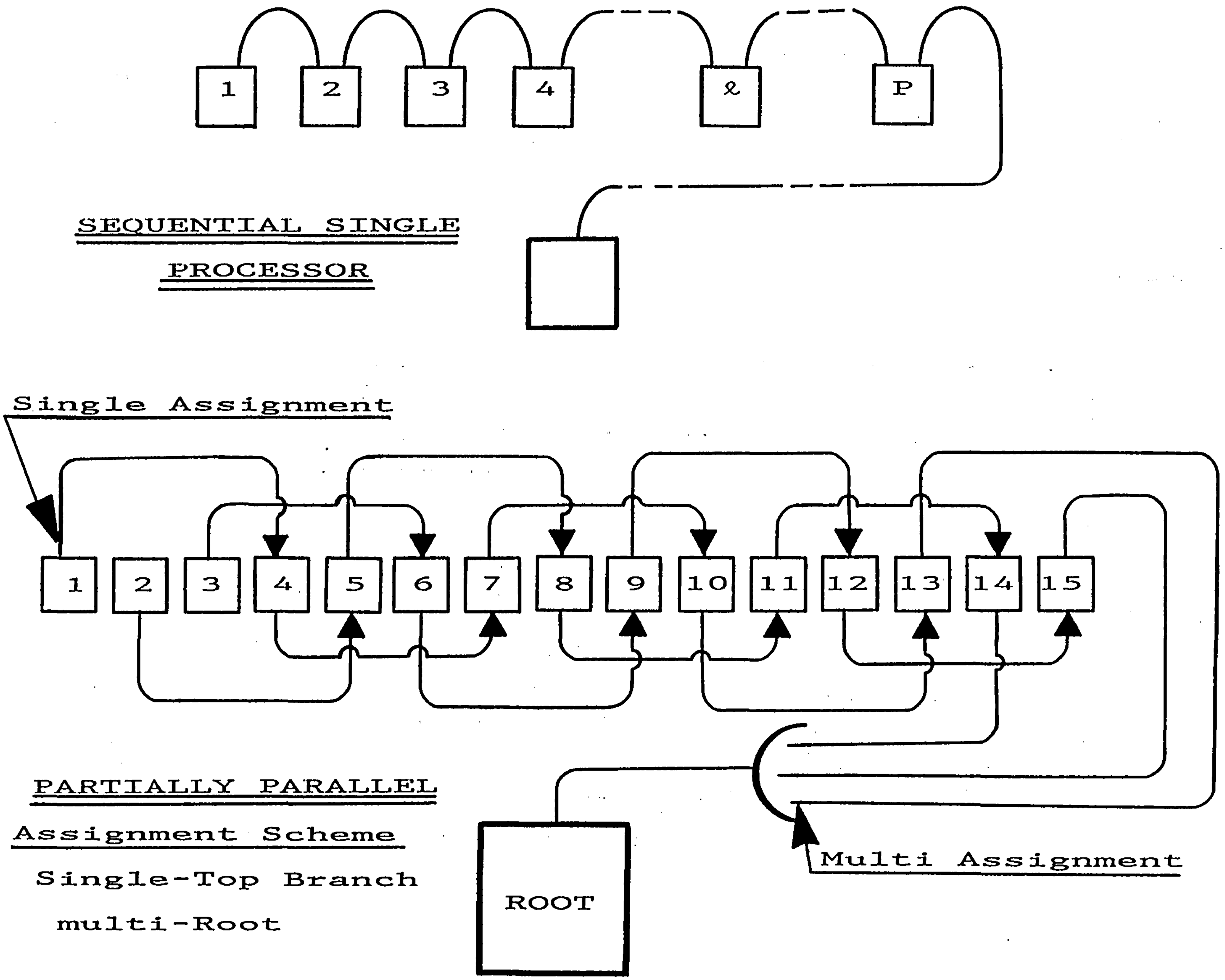

1
2
3
4
ℓ
P
SEQUENTIAL SINGLE PROCESSOR
Single Assignment
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
PARTIALLY PARALLEL
Assignment Scheme
Single-Top Branch
multi-Root
ROOT
Multi Assignment

FIGURE 2.3

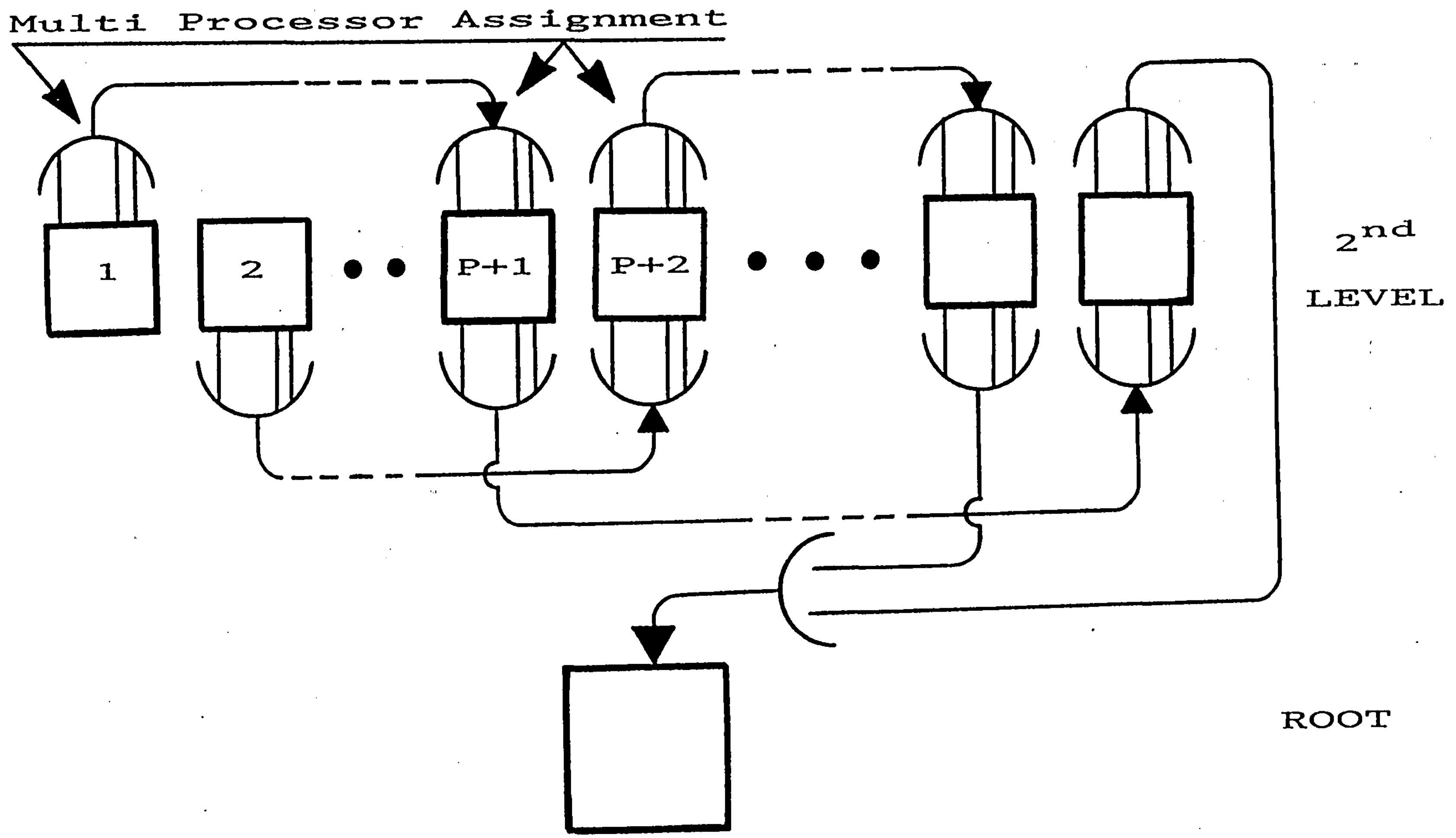


FIGURE 2.4

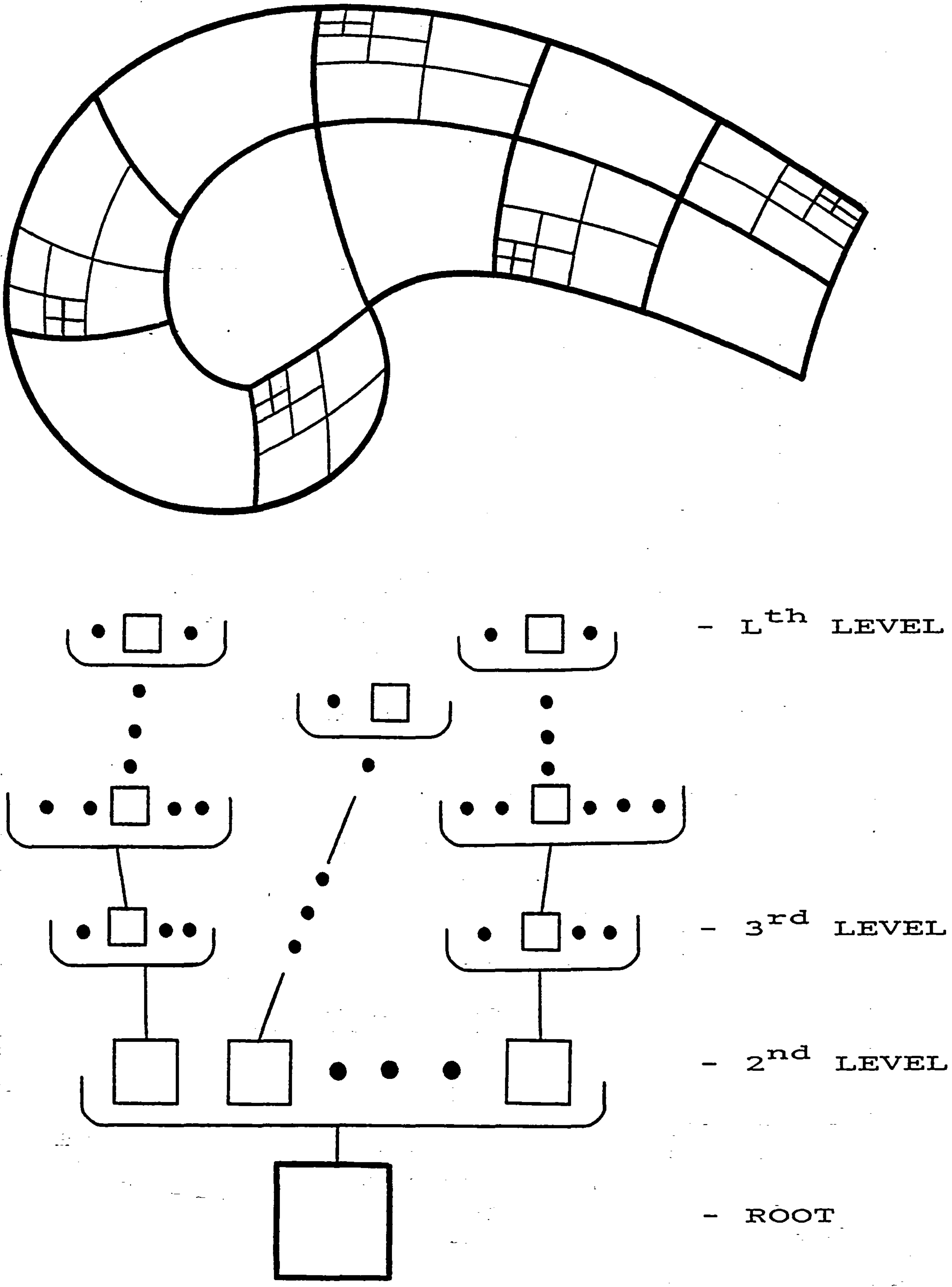


FIGURE 3.1

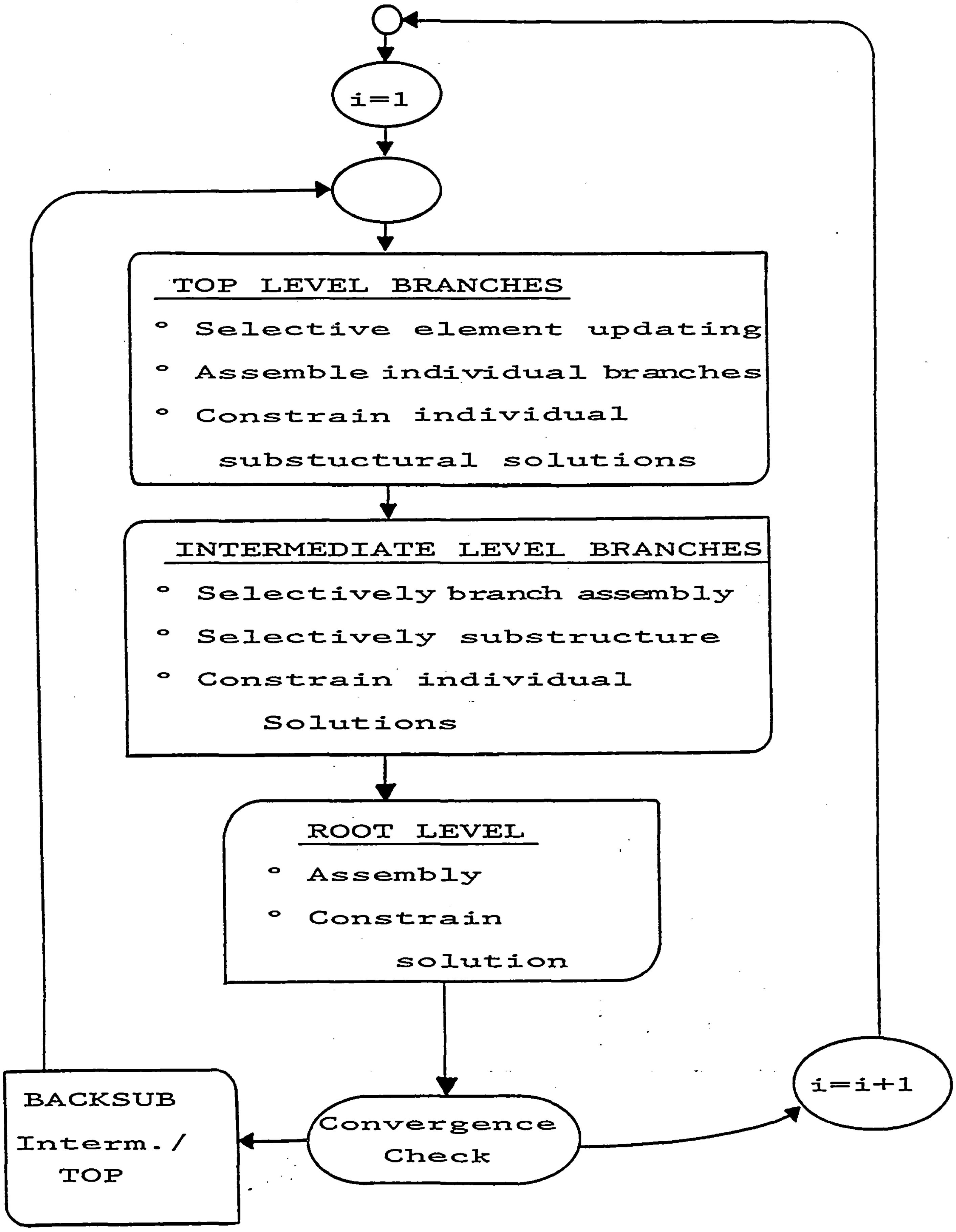

i=1
TOP LEVEL BRANCHES
° Selective element updating
° Assemble individual branches
° Constrain individual substuctural solutions
INTERMEDIATE LEVEL BRANCHES
° Selectively branch assembly
° Selectively substructure
° Constrain individual Solutions
ROOT LEVEL
° Assembly
° Constrain solution
BACKSUB Interm./ TOP
Convergence Check
i=i+1

FIGURE 3.2

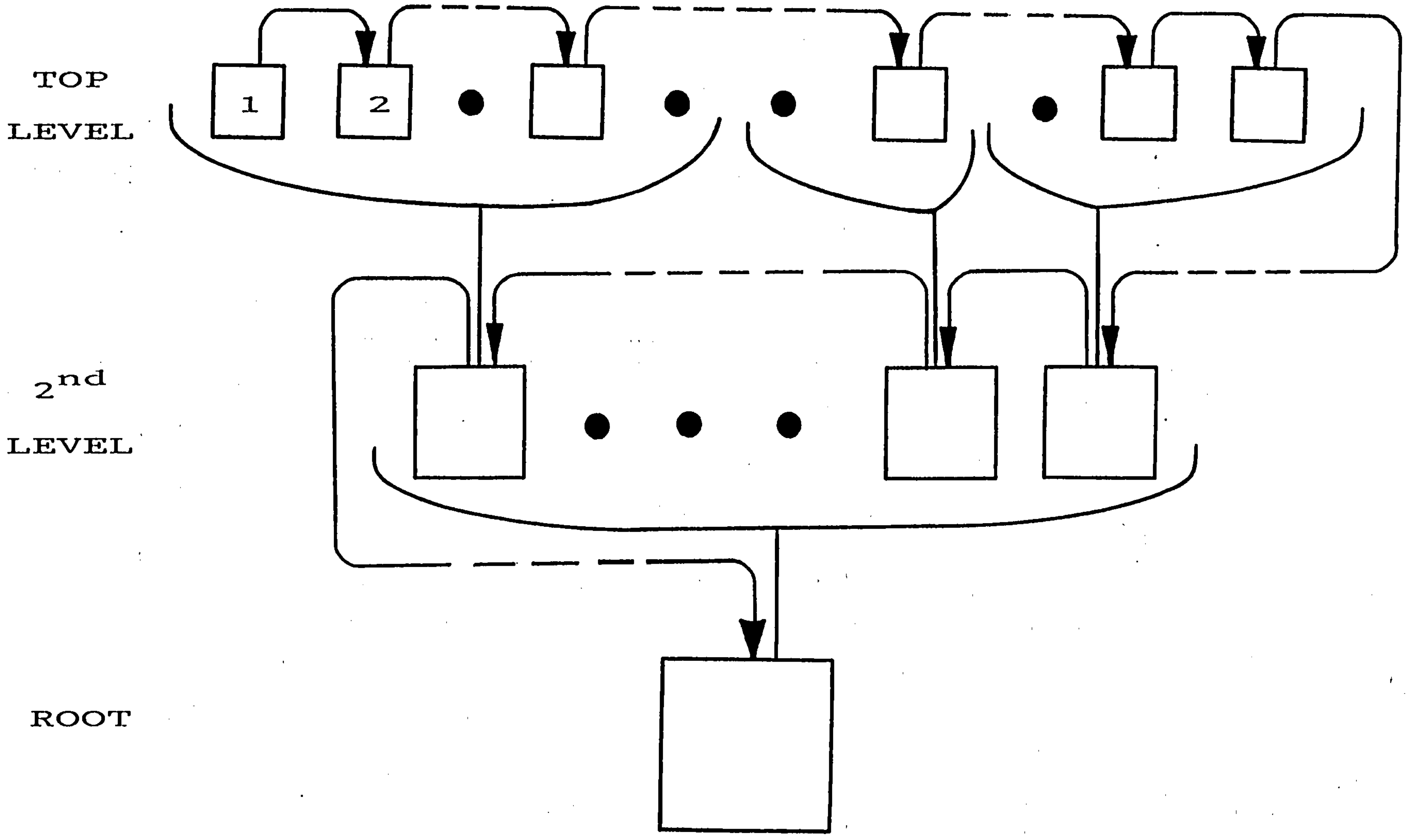


FIGURE 3.3

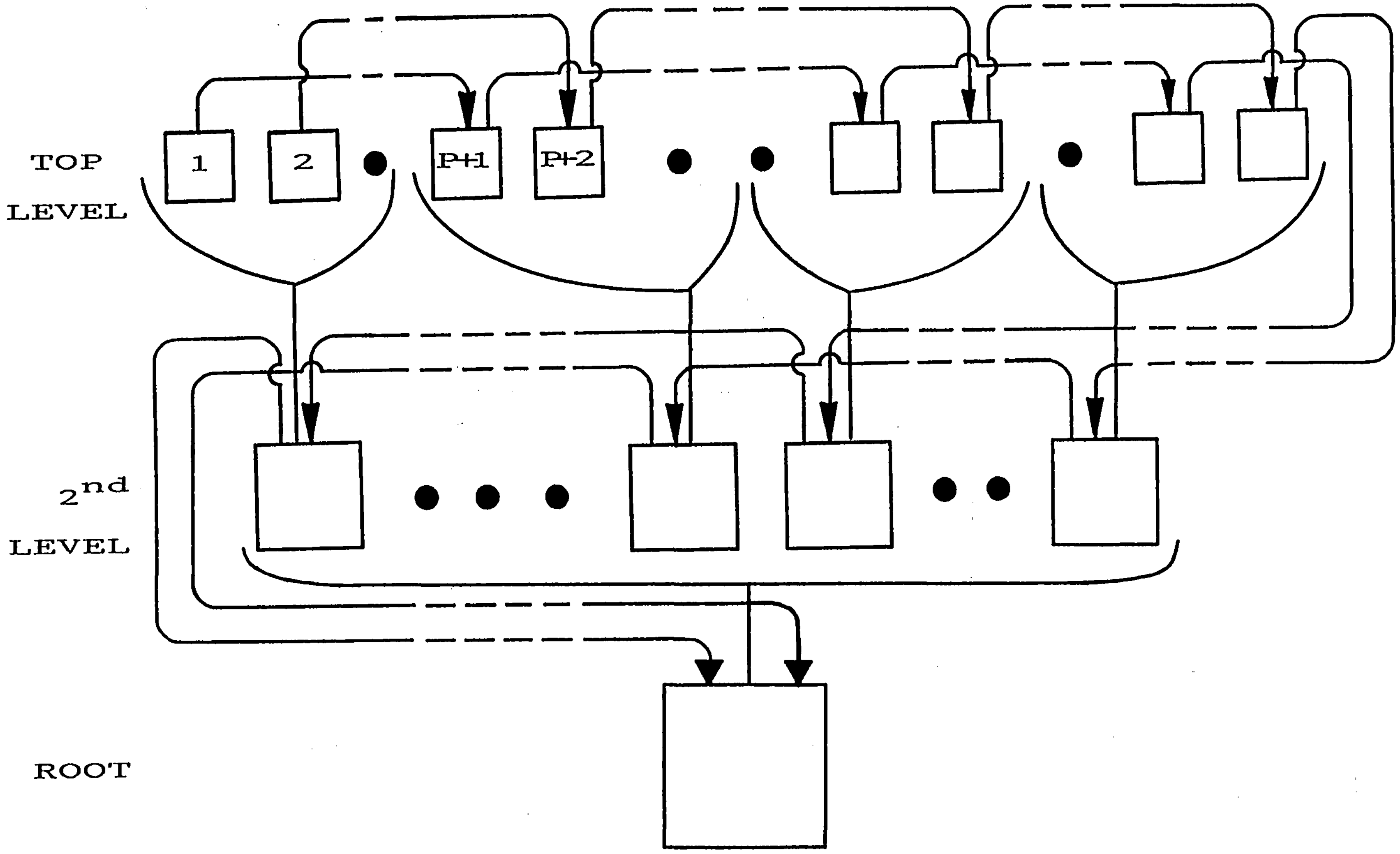


FIGURE 3.4

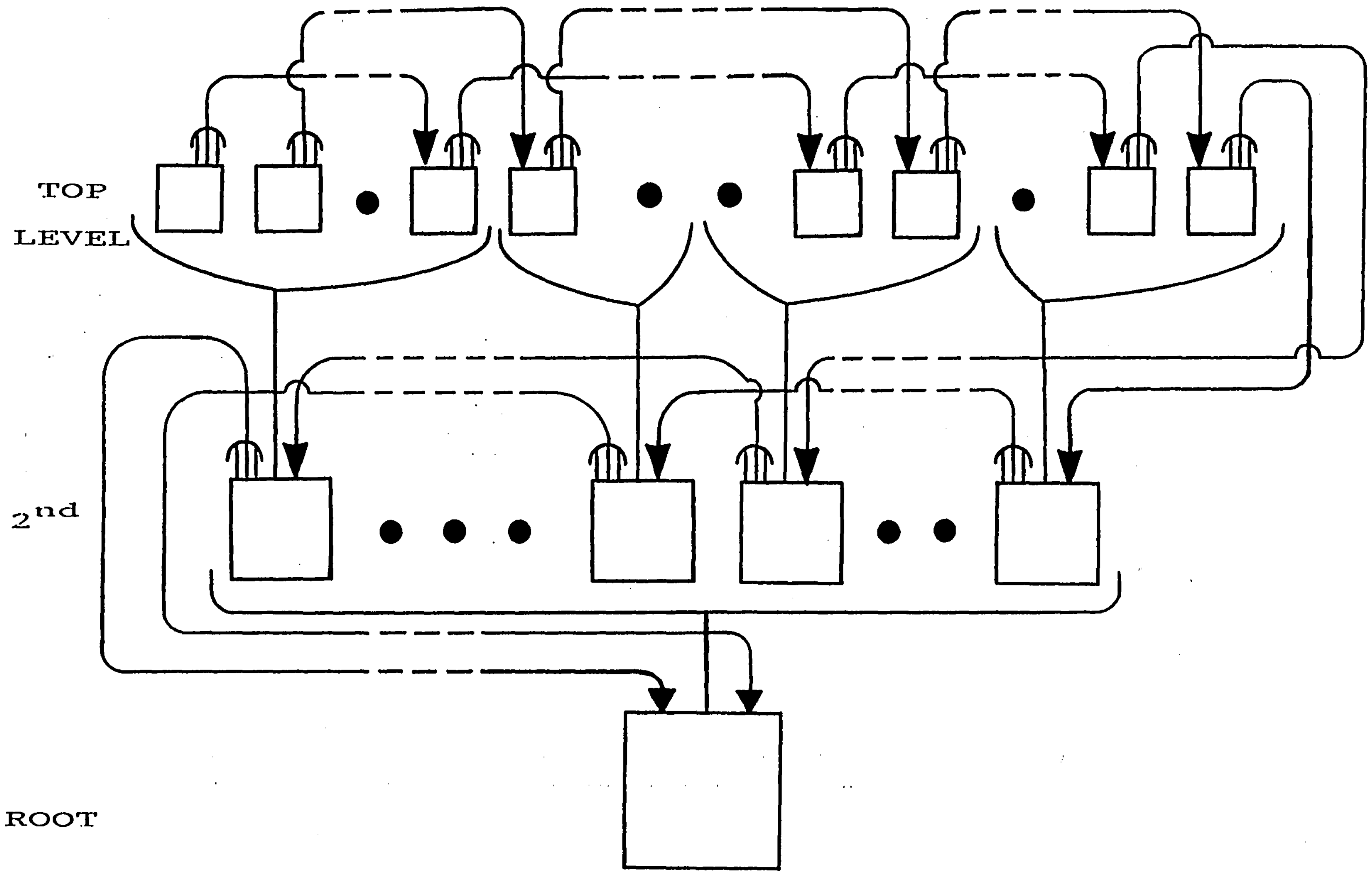


FIGURE 3.5

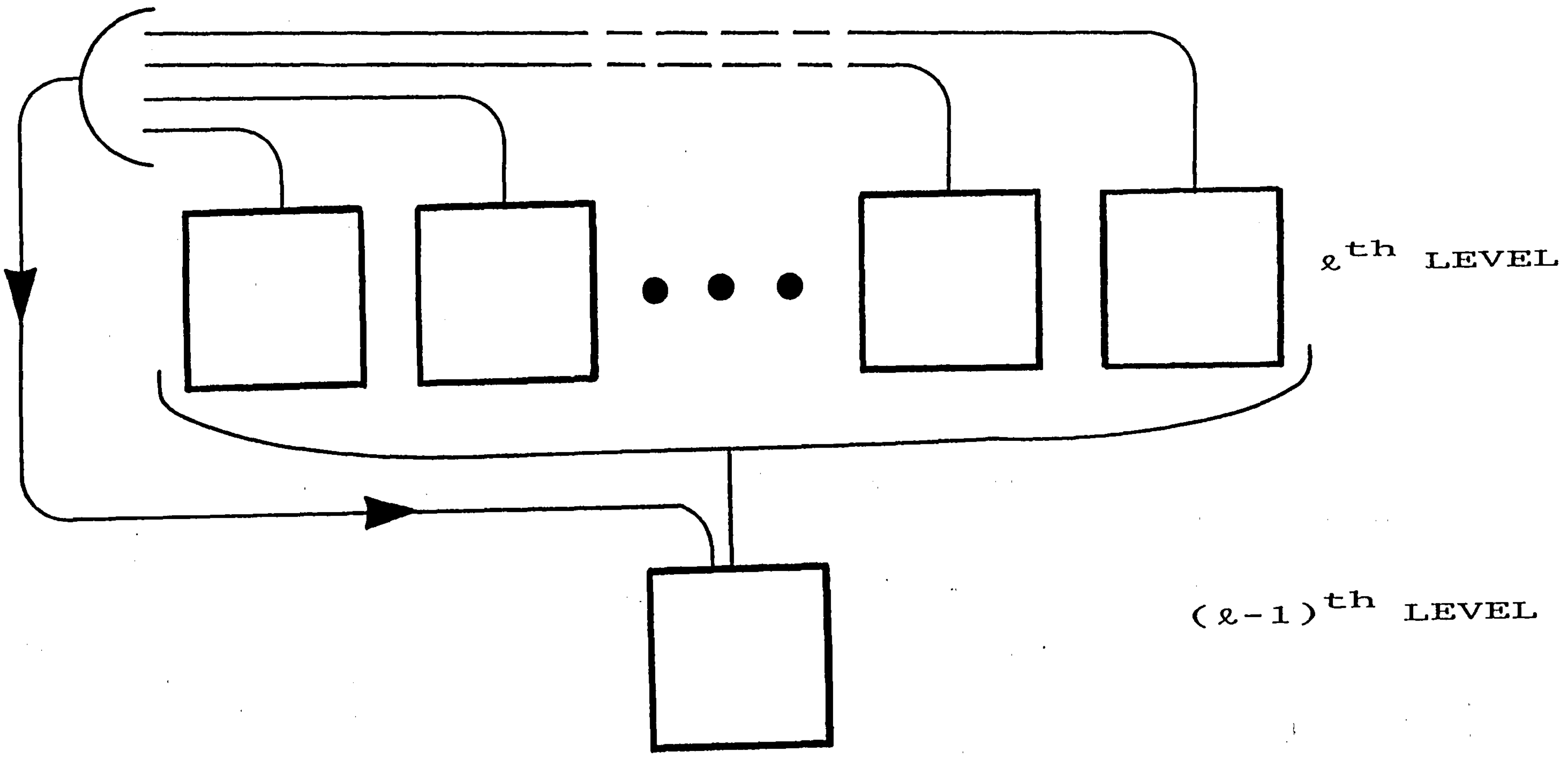


FIGURE 3.6

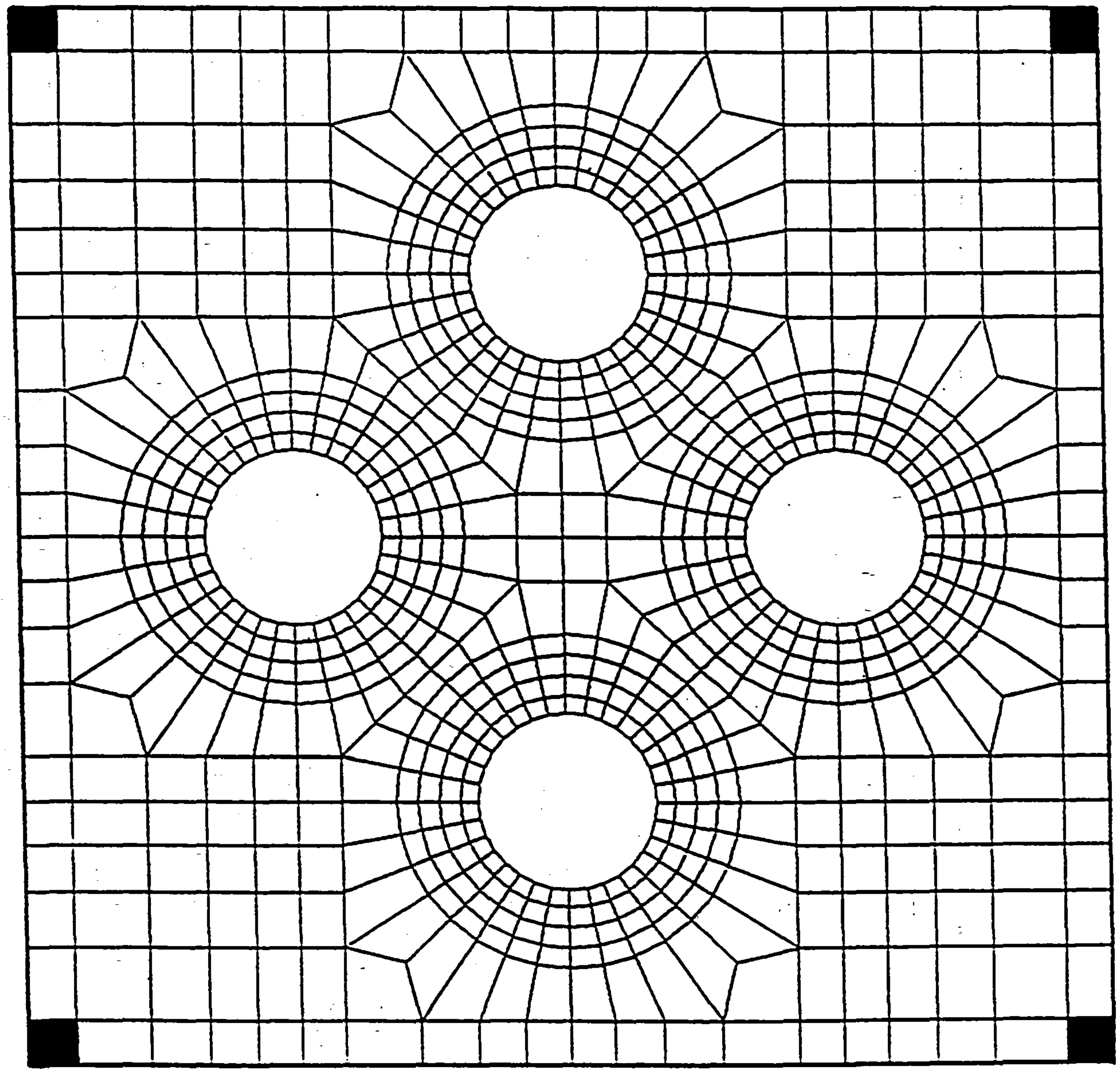


FIGURE 4.1

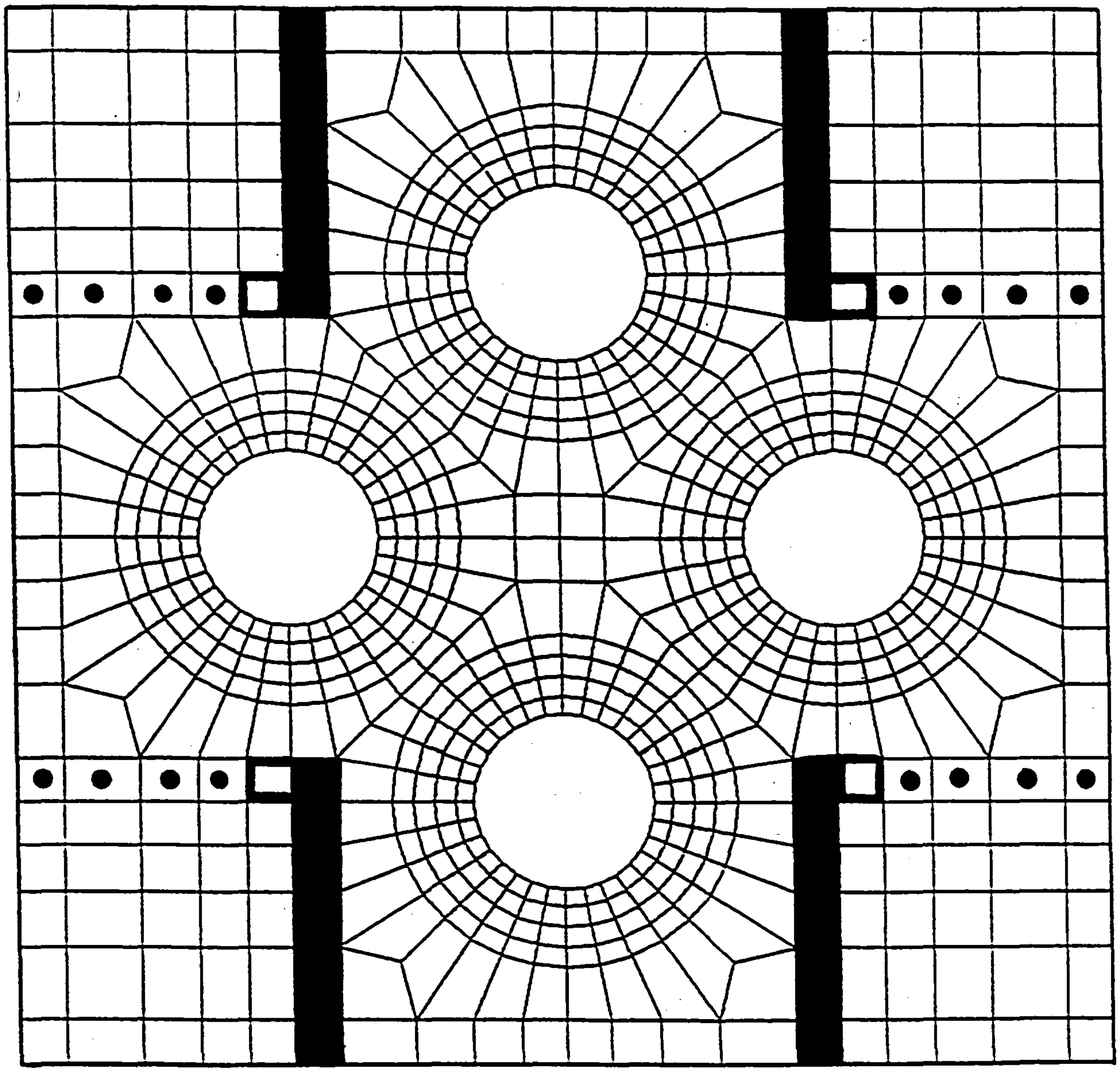


FIGURE 4.2

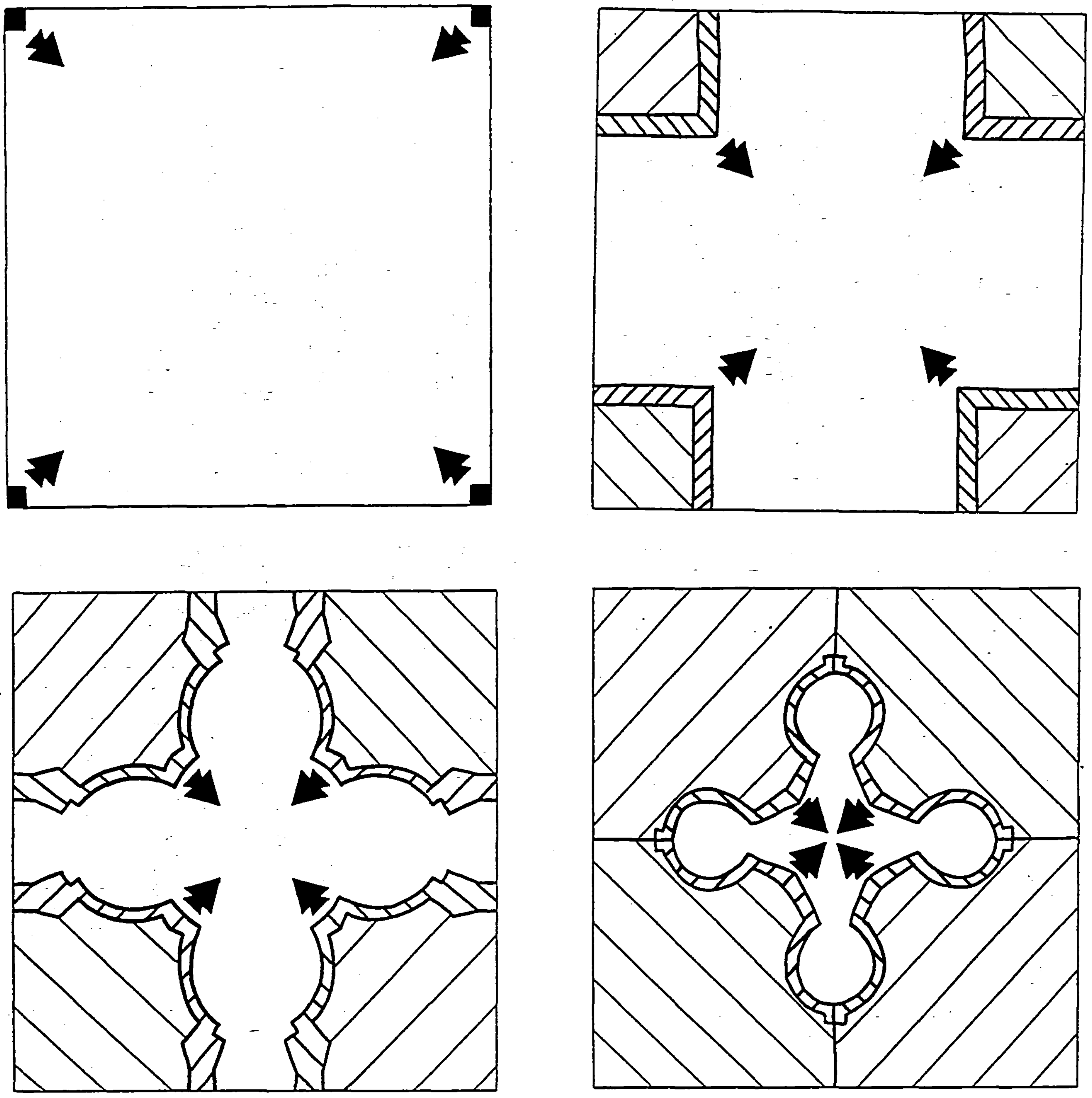


FIGURE 4.3

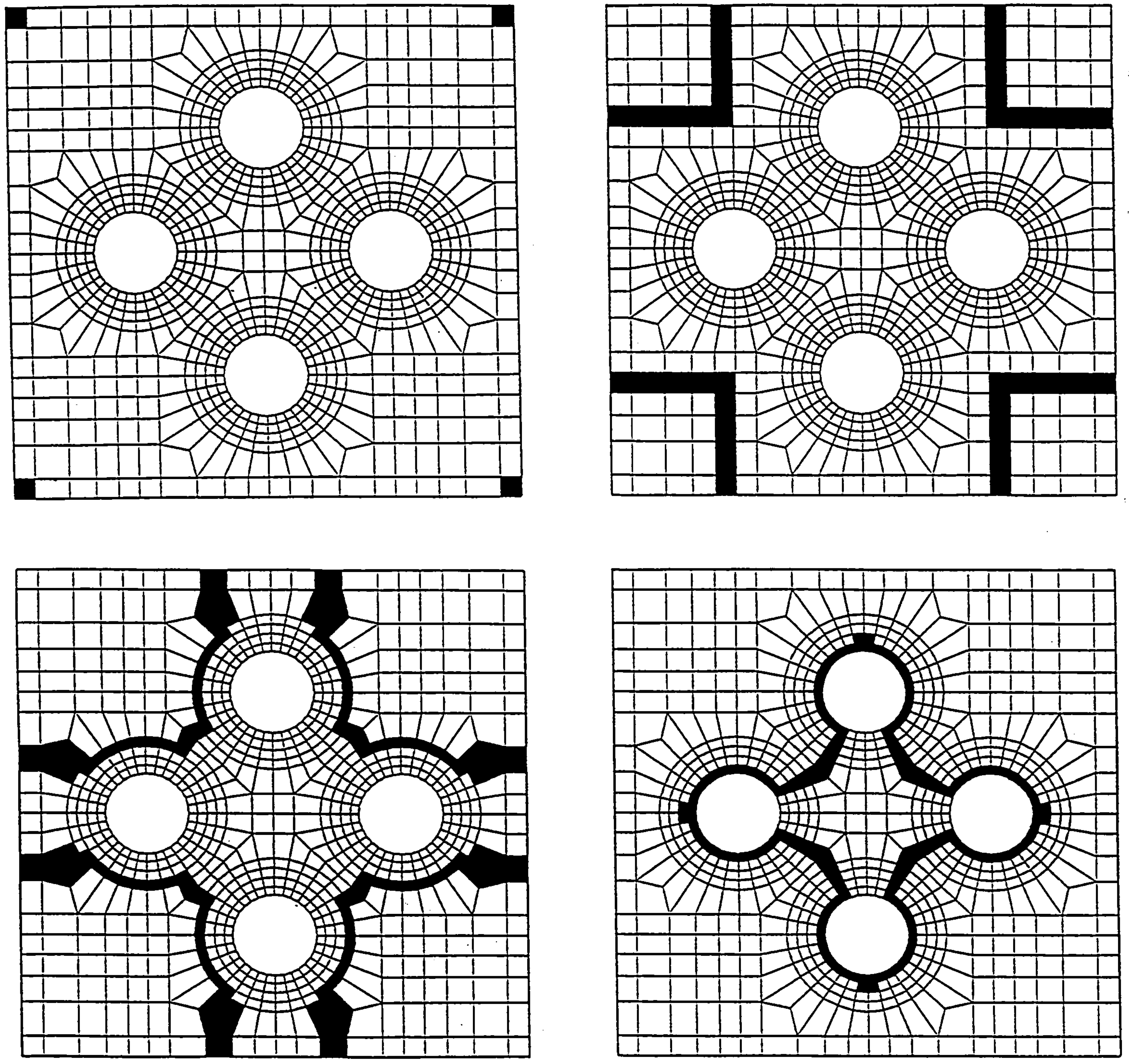


FIGURE 4.4

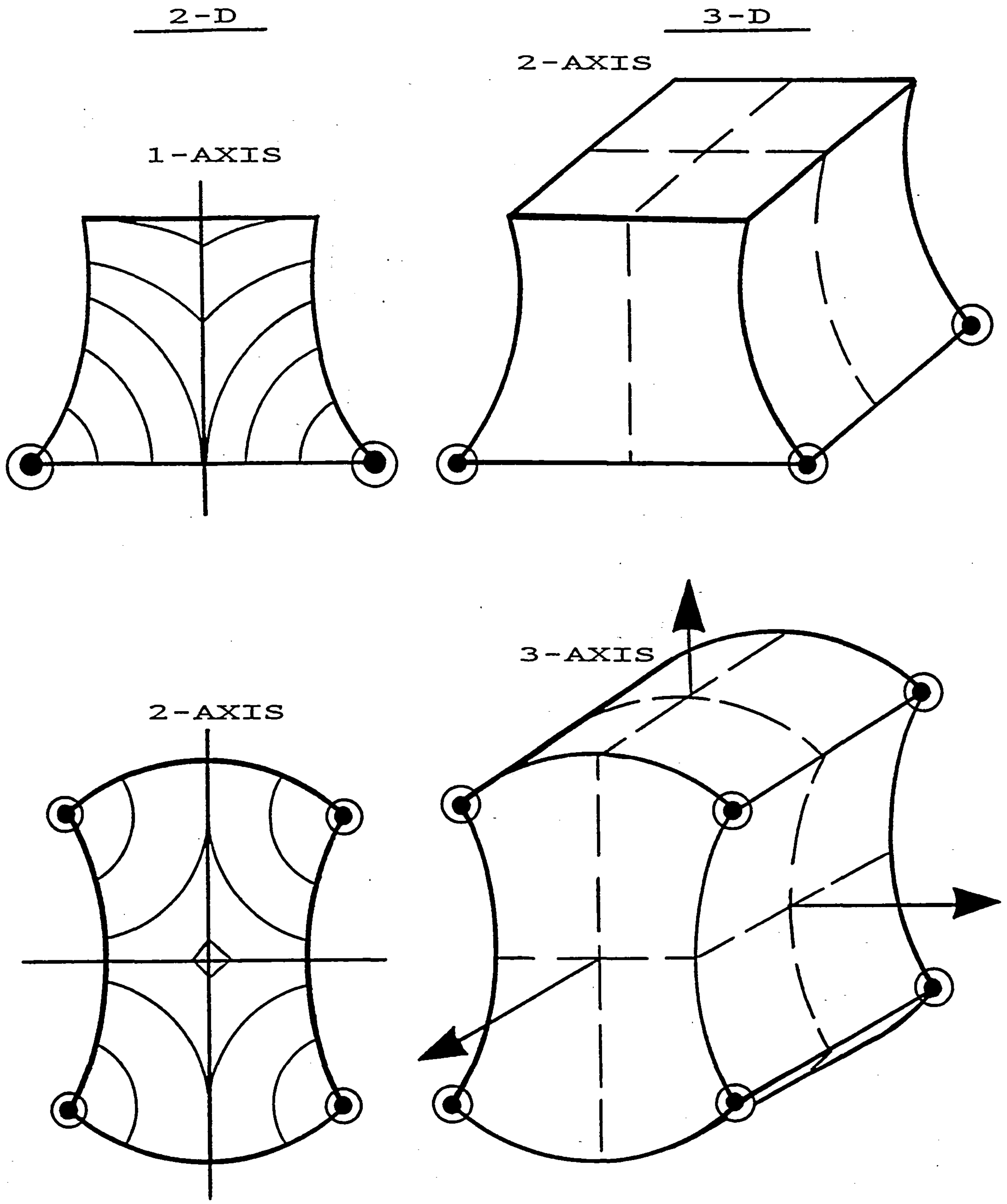


FIGURE 4.5

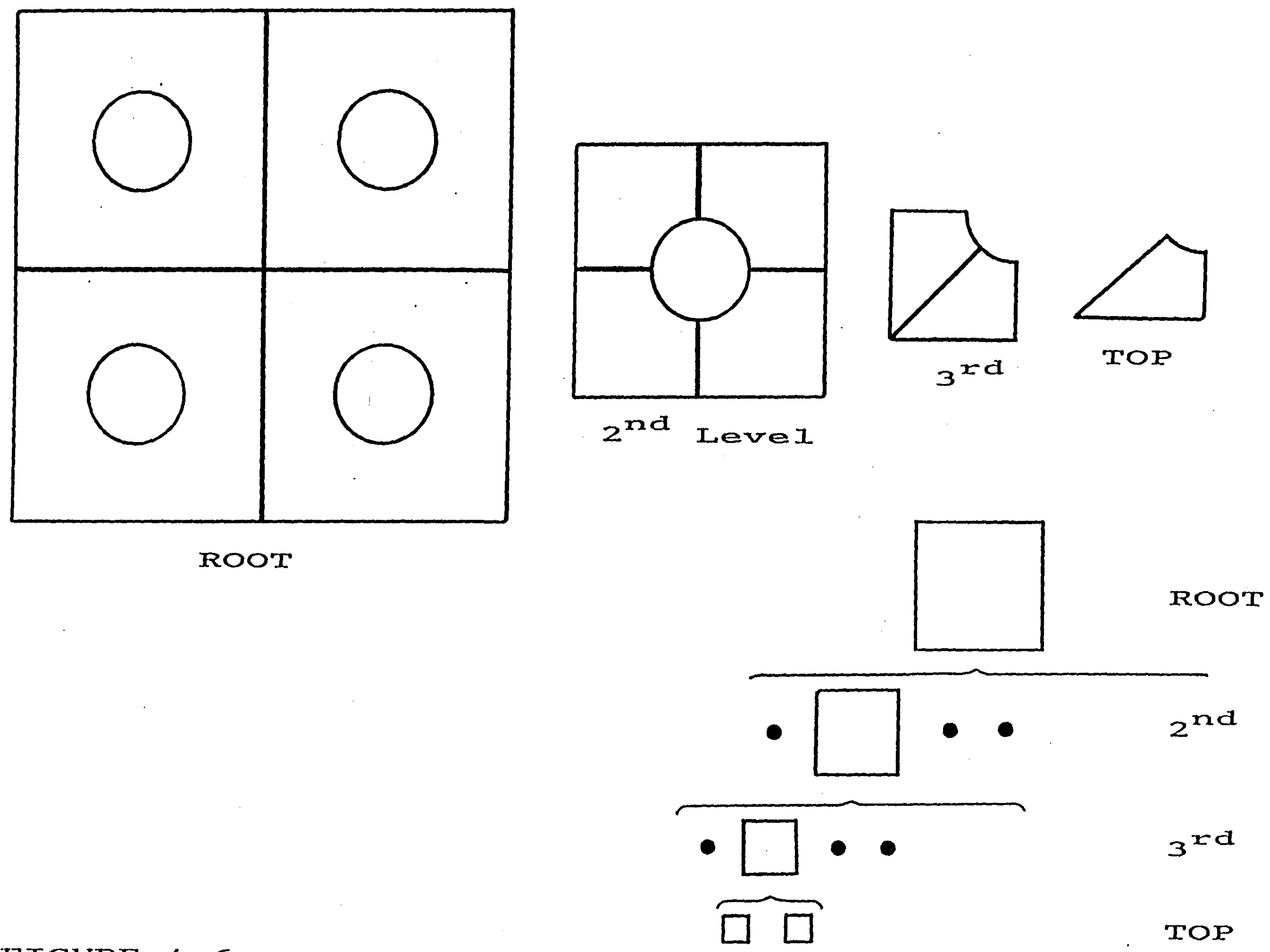
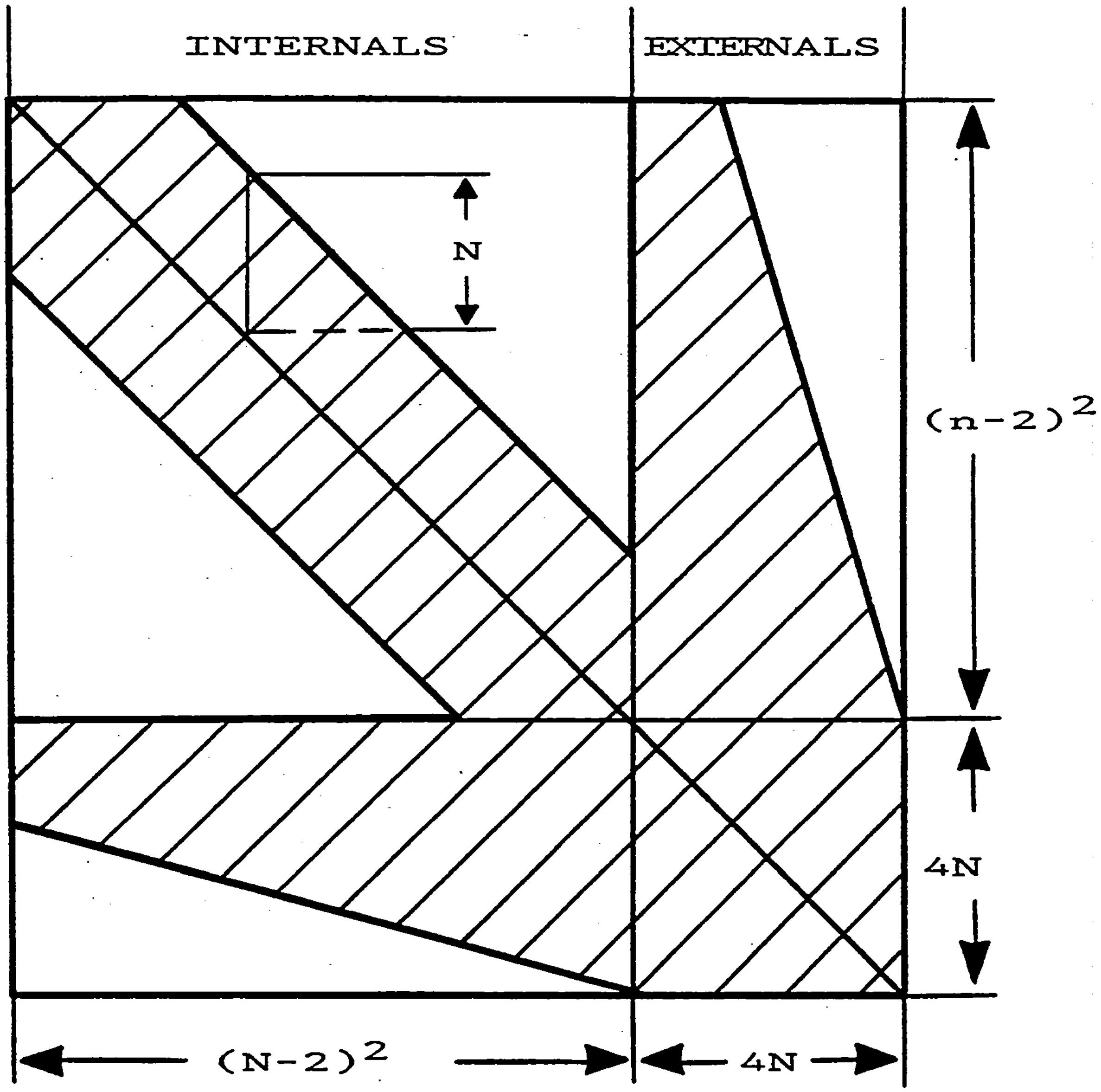


FIGURE 4.6



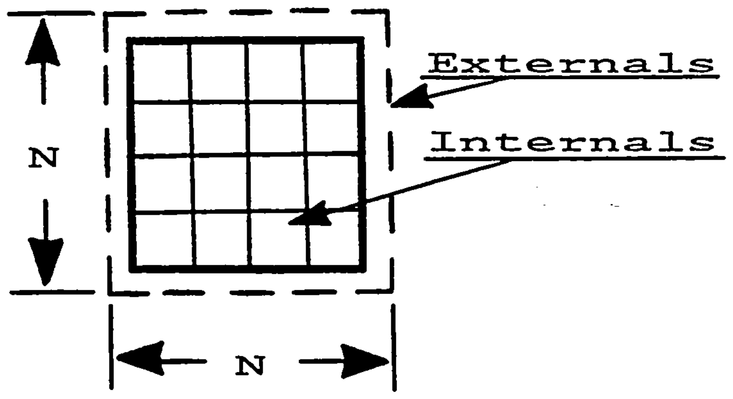


FIGURE 4.7

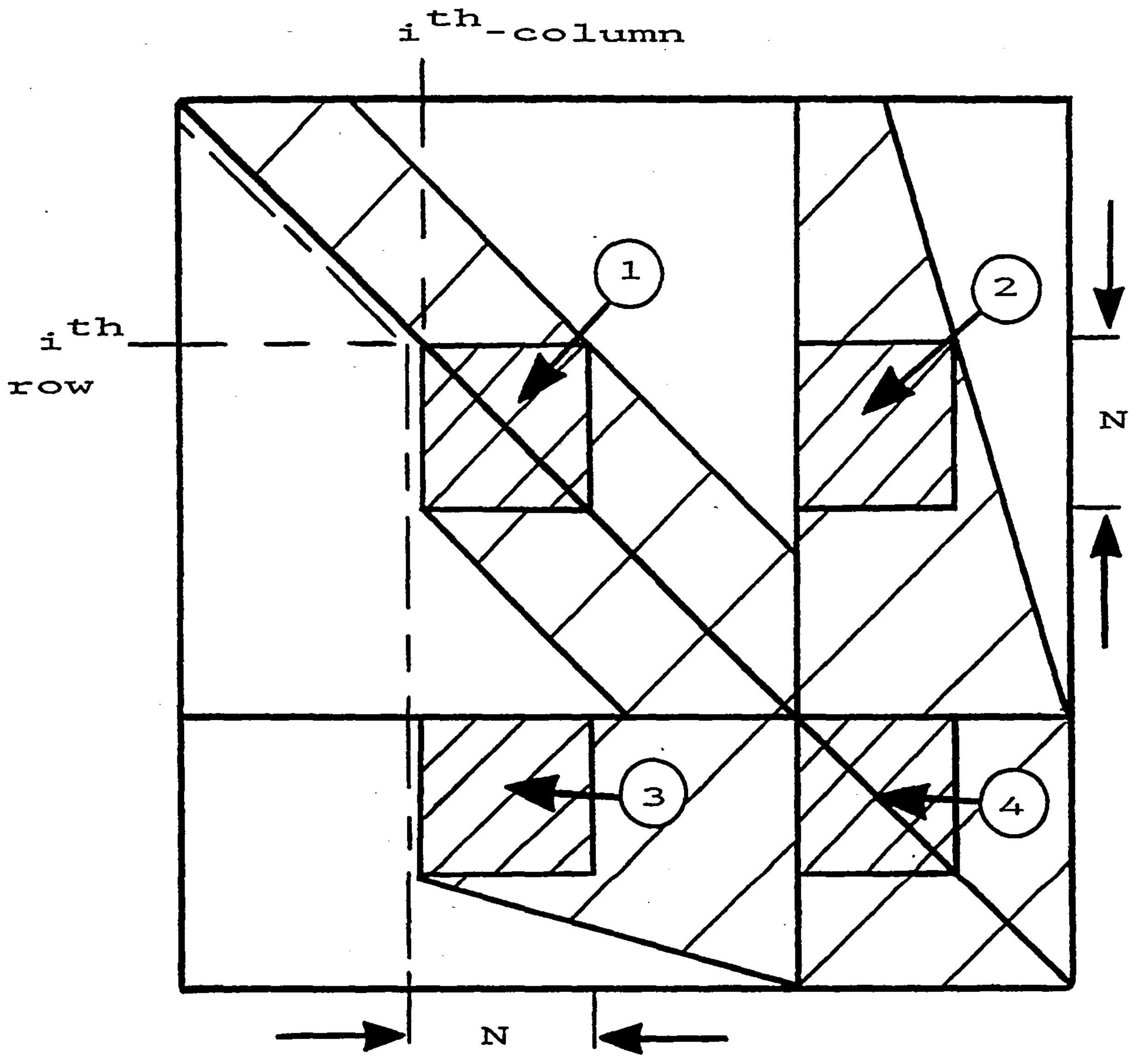


FIGURE 4.8

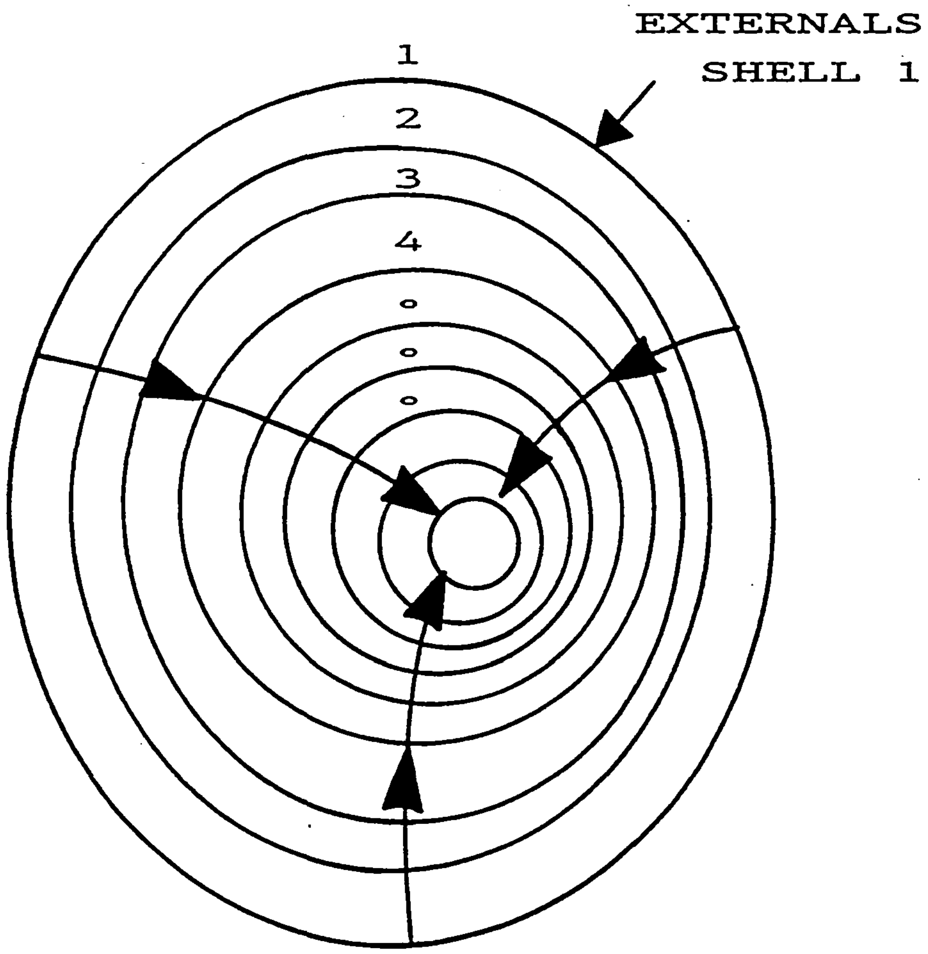


FIGURE 4.9

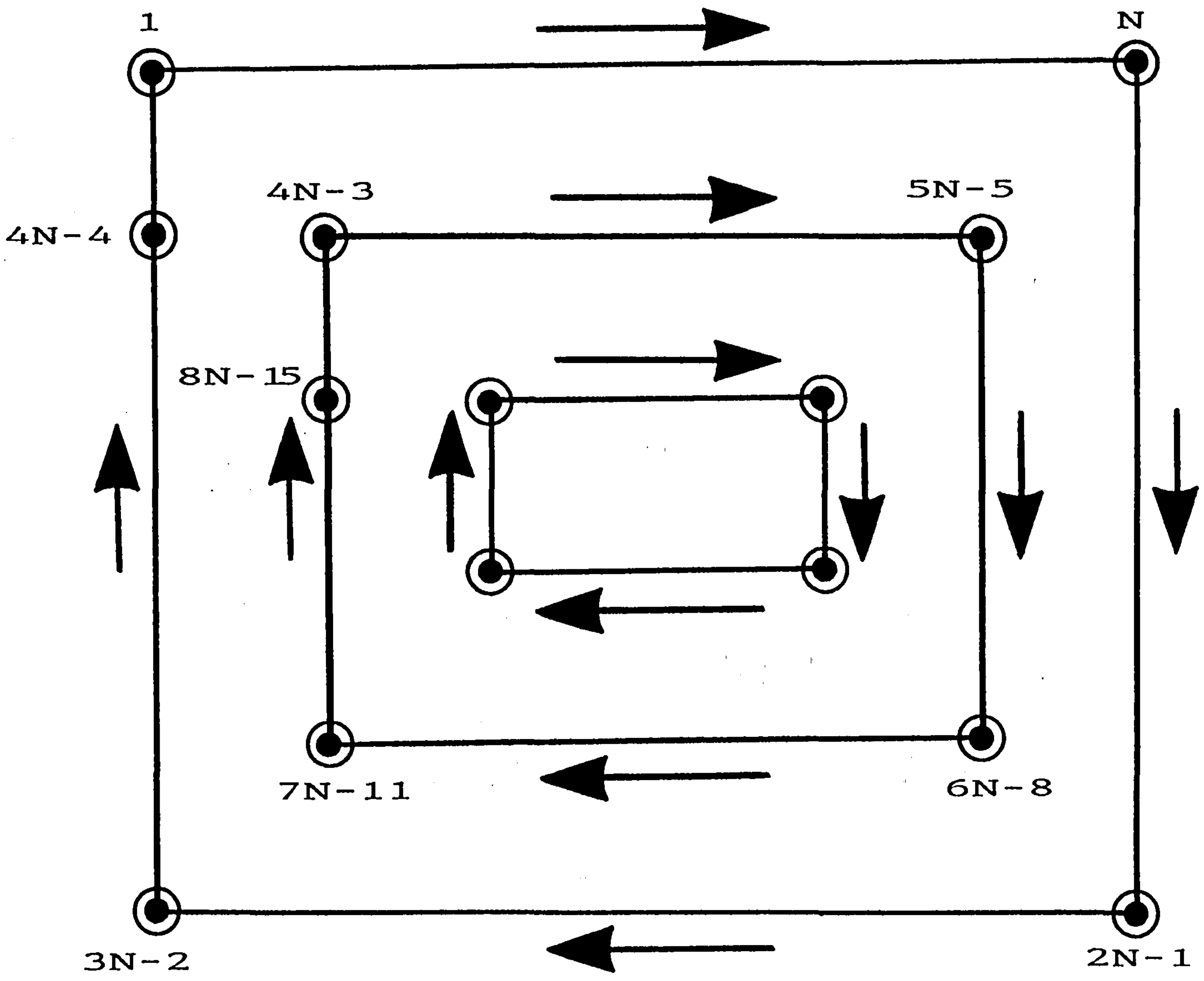


FIGURE 4.10

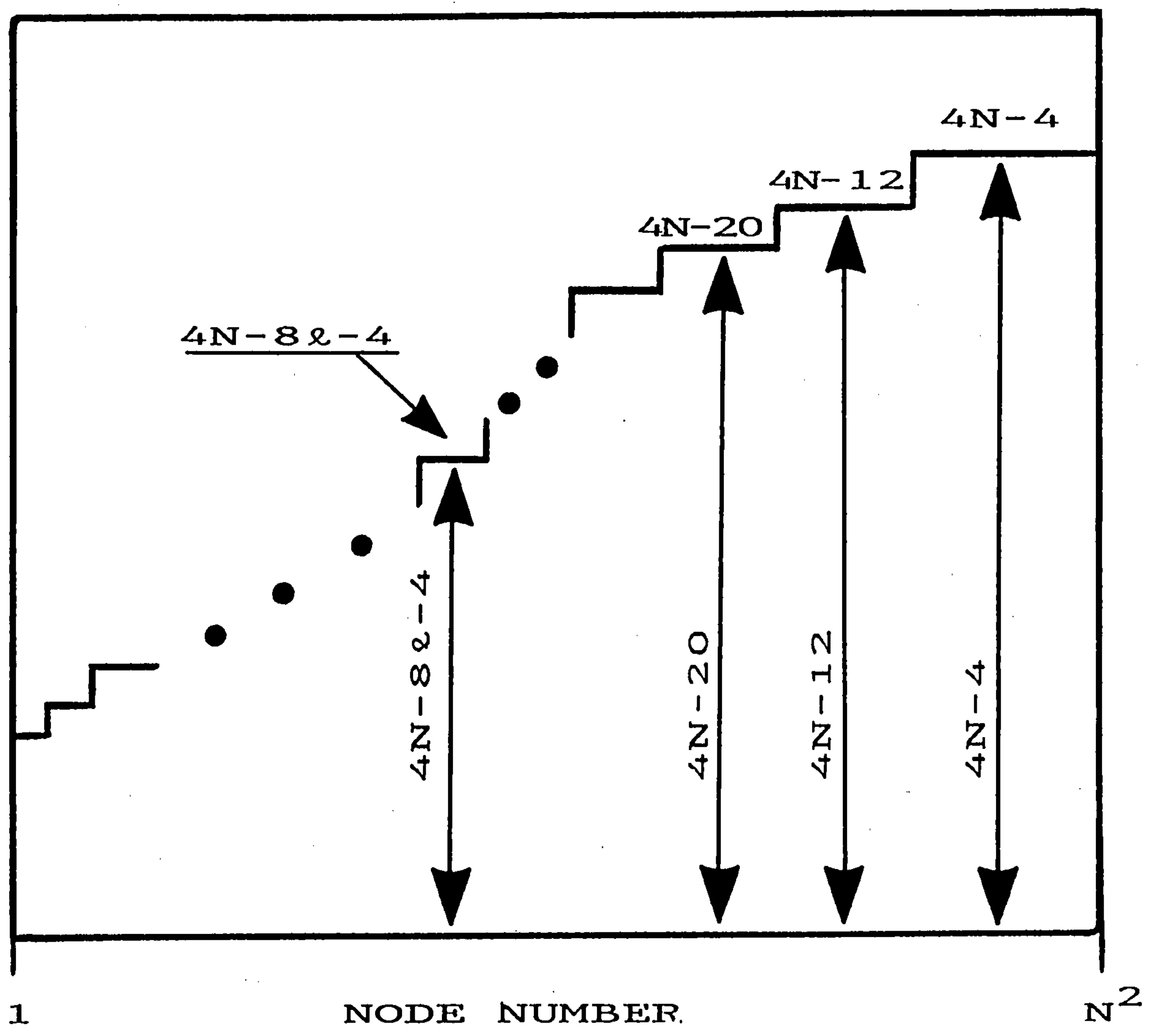


FIGURE 4.11

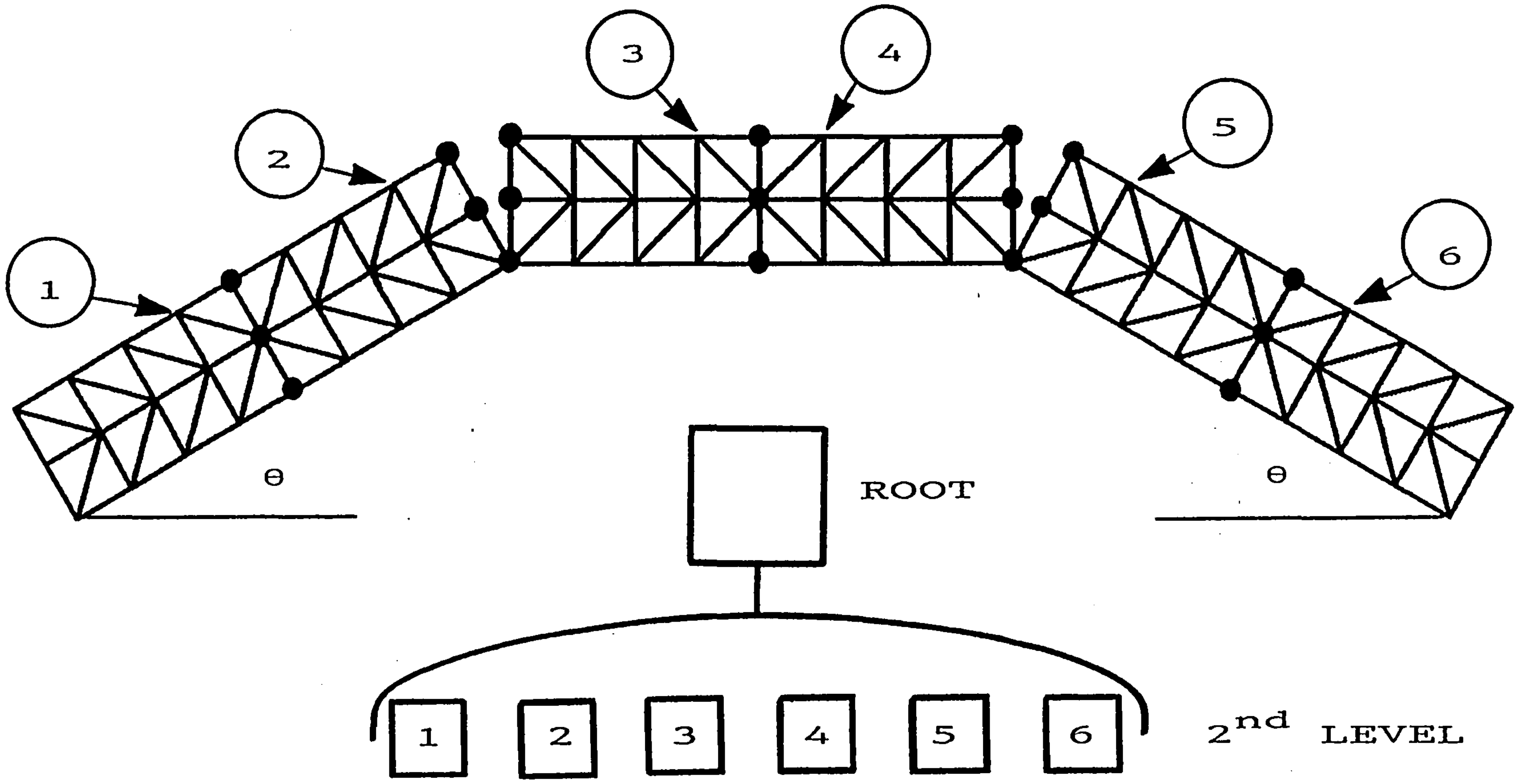


FIGURE 5.1

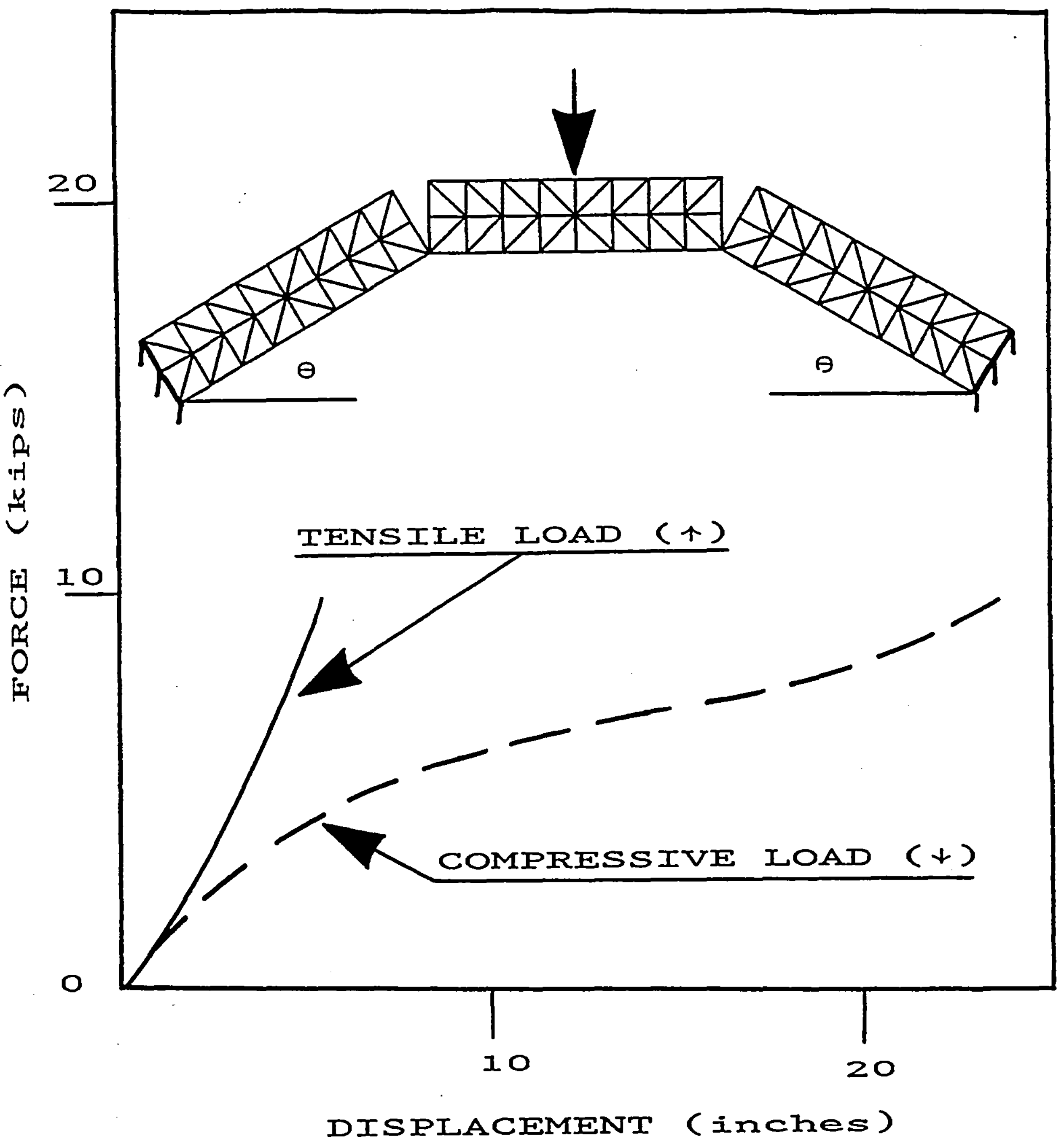

20
10
0
FORCE (kips)
θ
θ
TENSILE LOAD (↑)
COMPRESSIVE LOAD (↓)
10
20
DISPLACEMENT (inches)

FIGURE 5.2

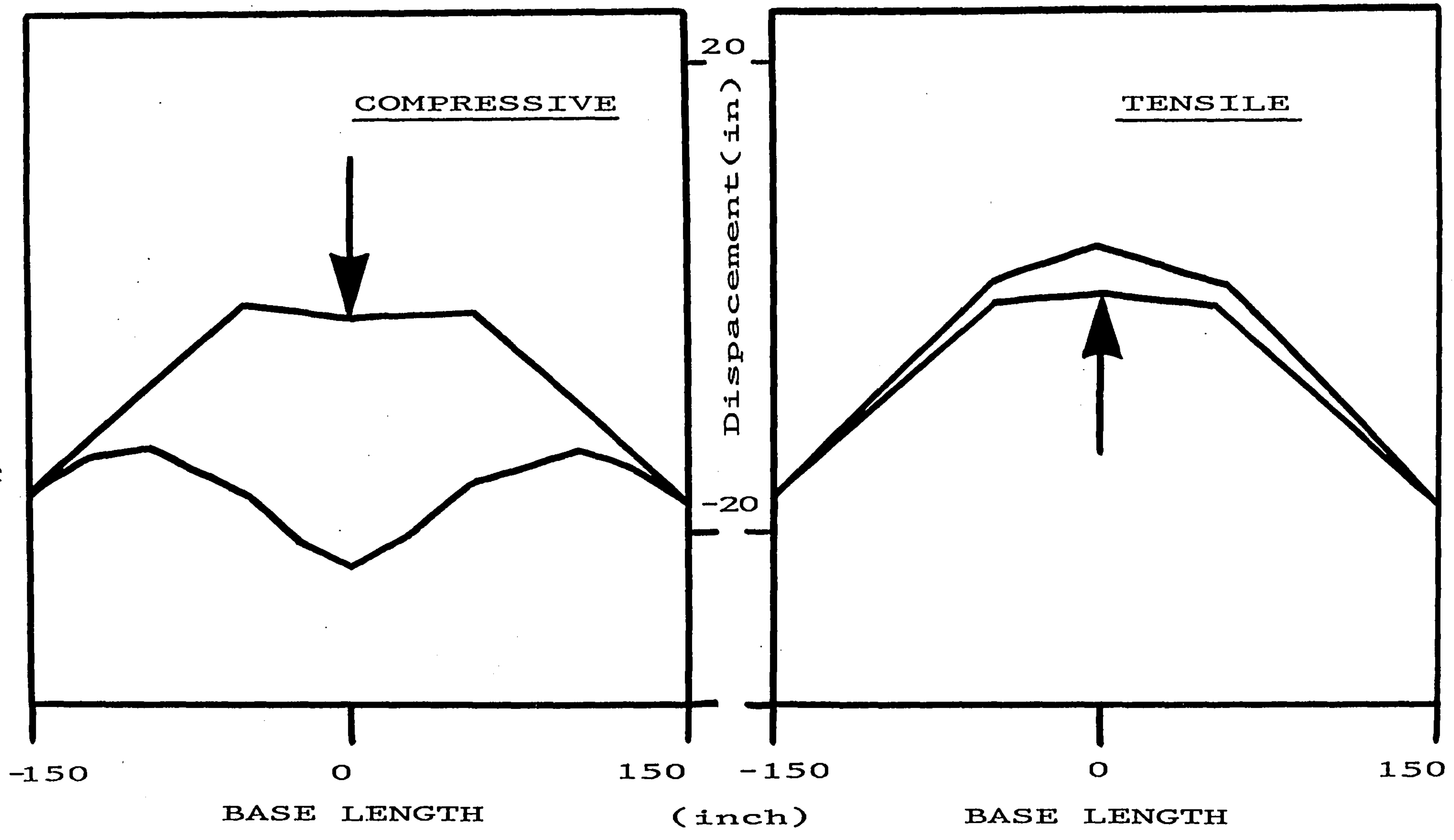

COMPRESSIVE
TENSILE
20
-20
Dispacement(in)
-150
0
150
-150
0
150
BASE LENGTH
(inch)
BASE LENGTH

FIGURE 5.3

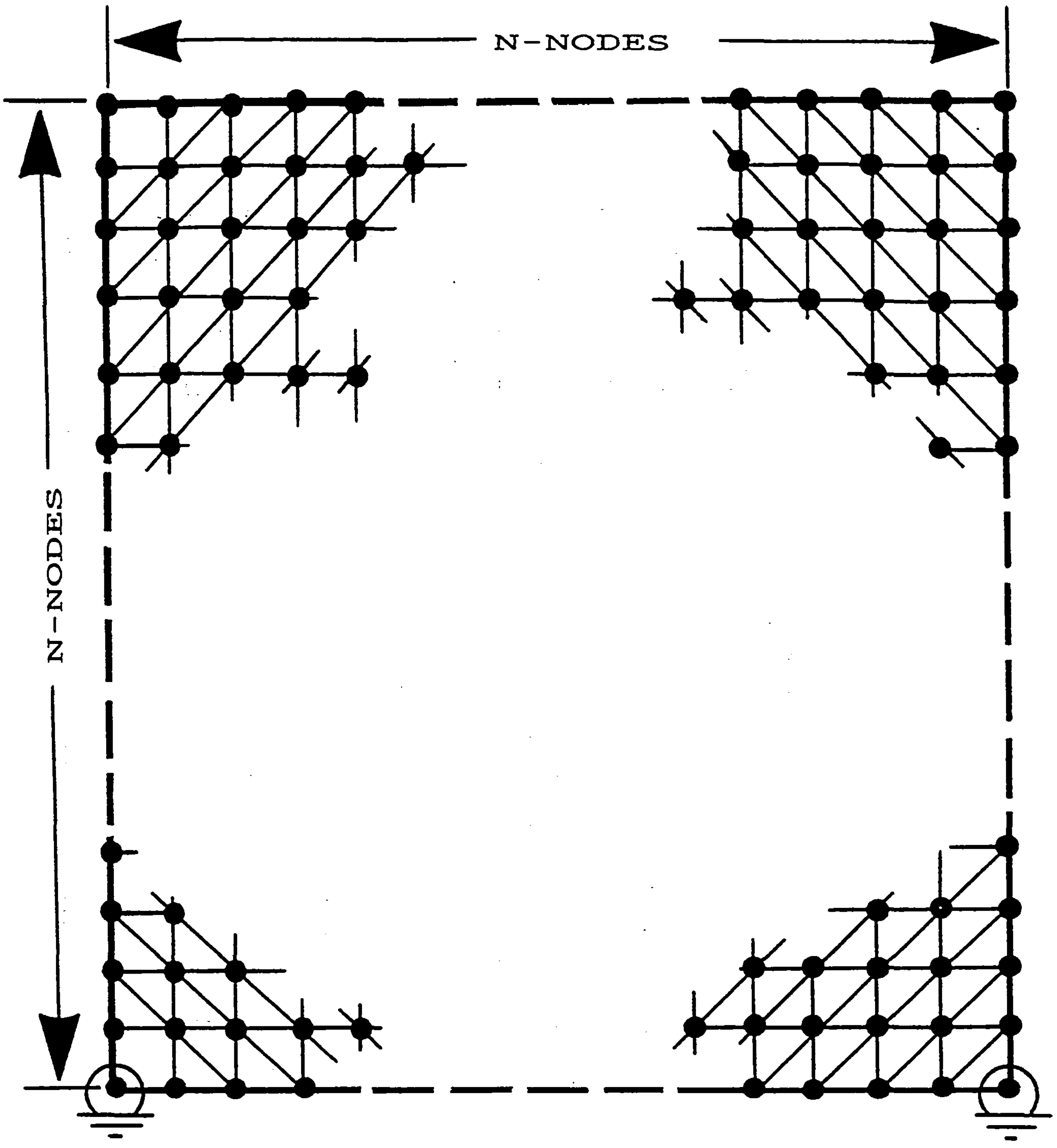


FIGURE 5.4

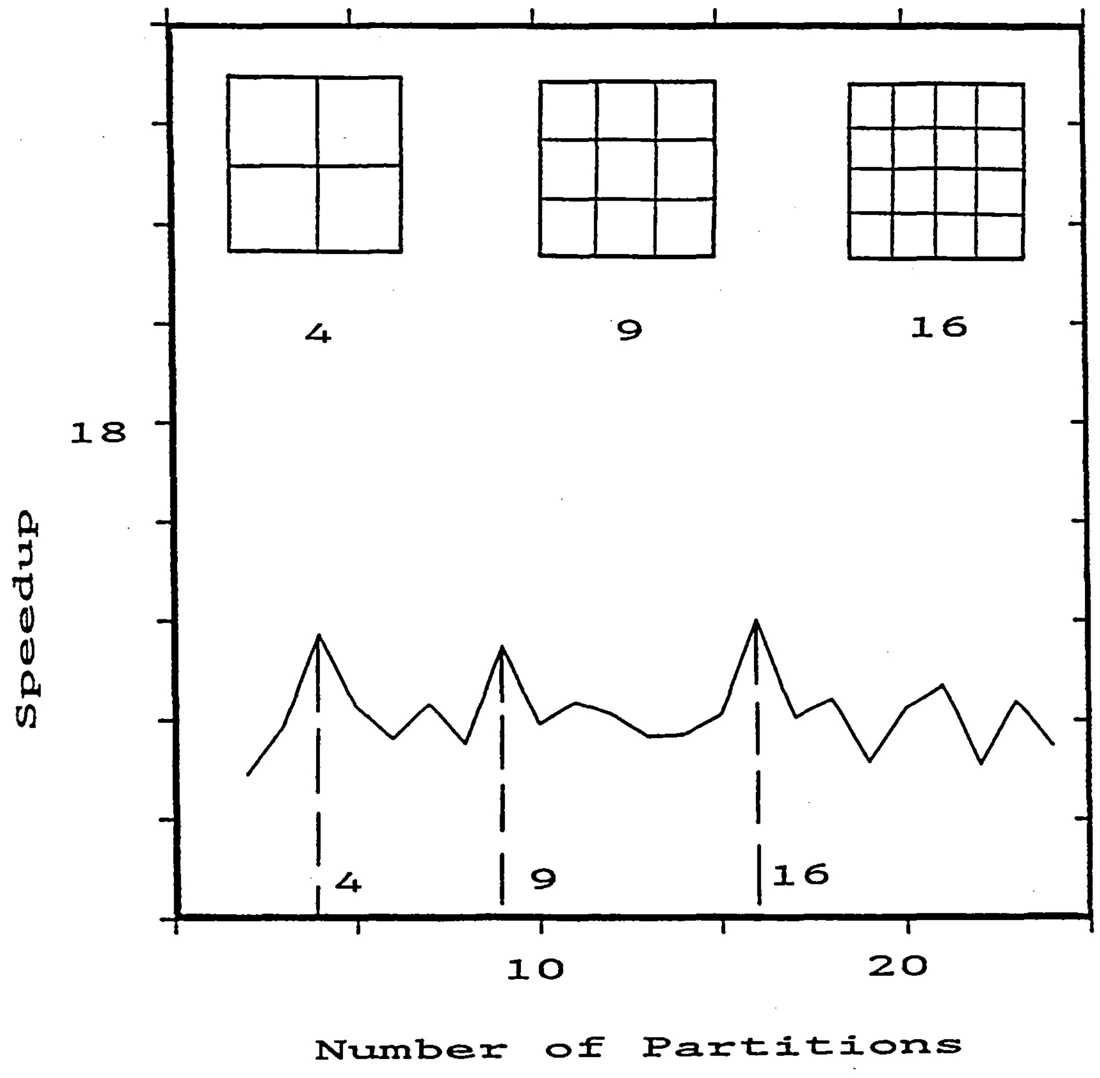


FIGURE 5.5

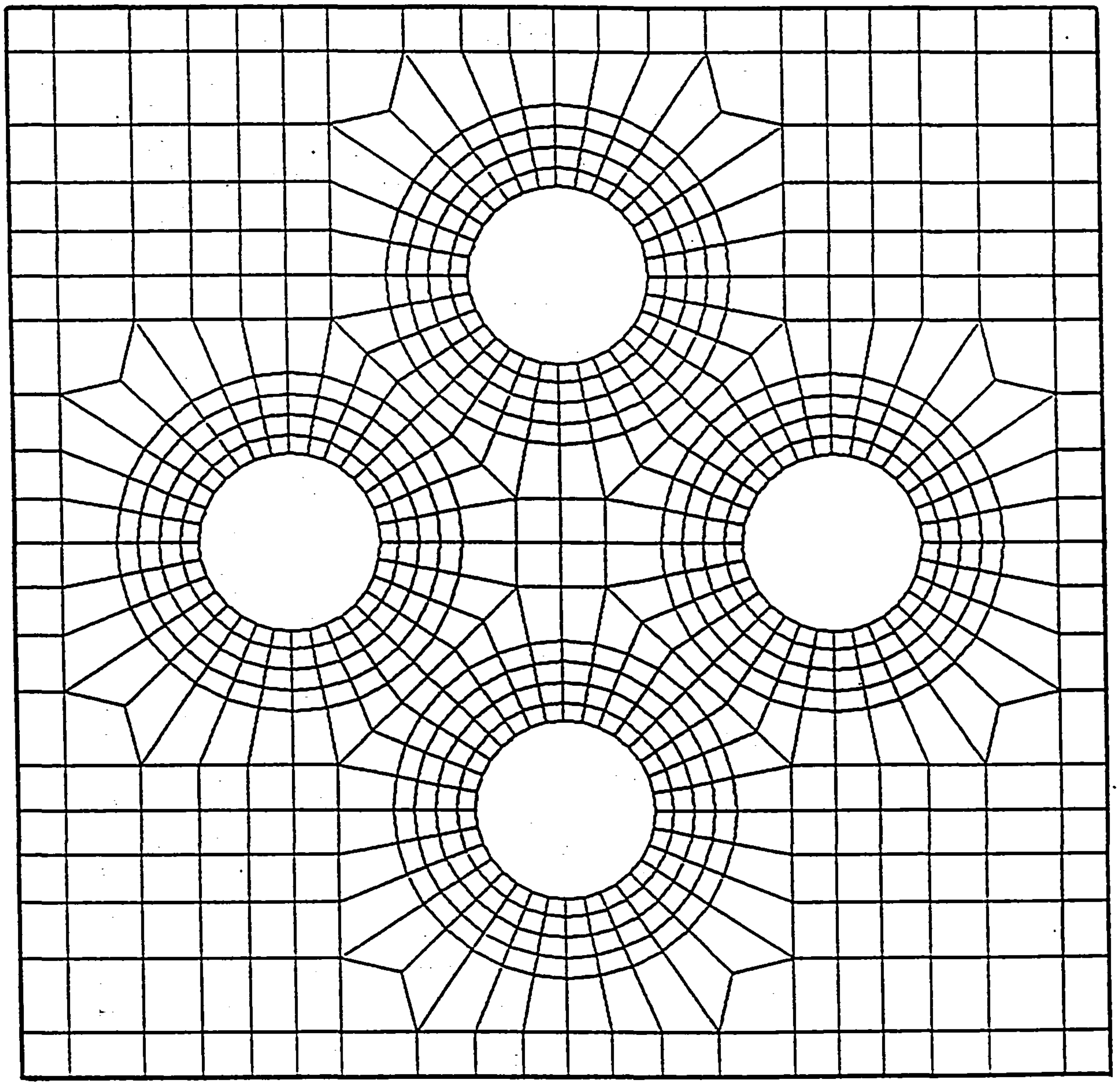


FIGURE 5.6

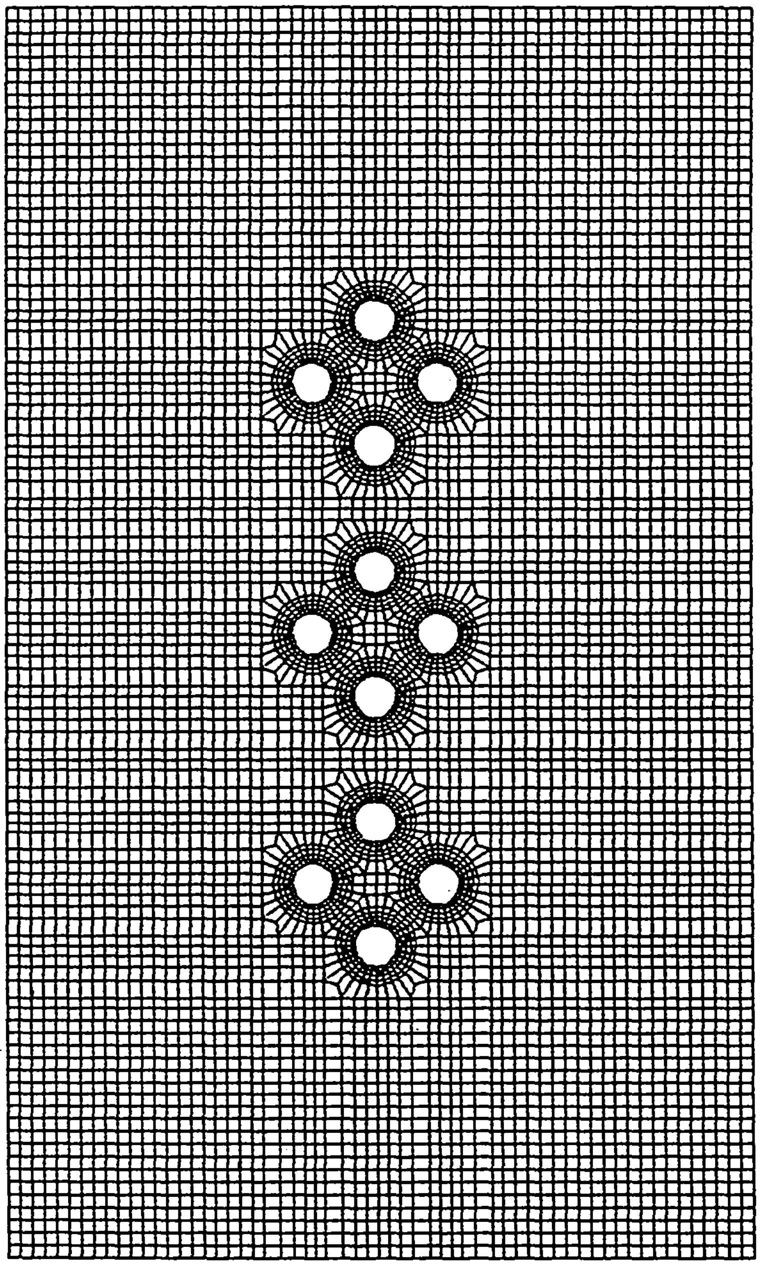
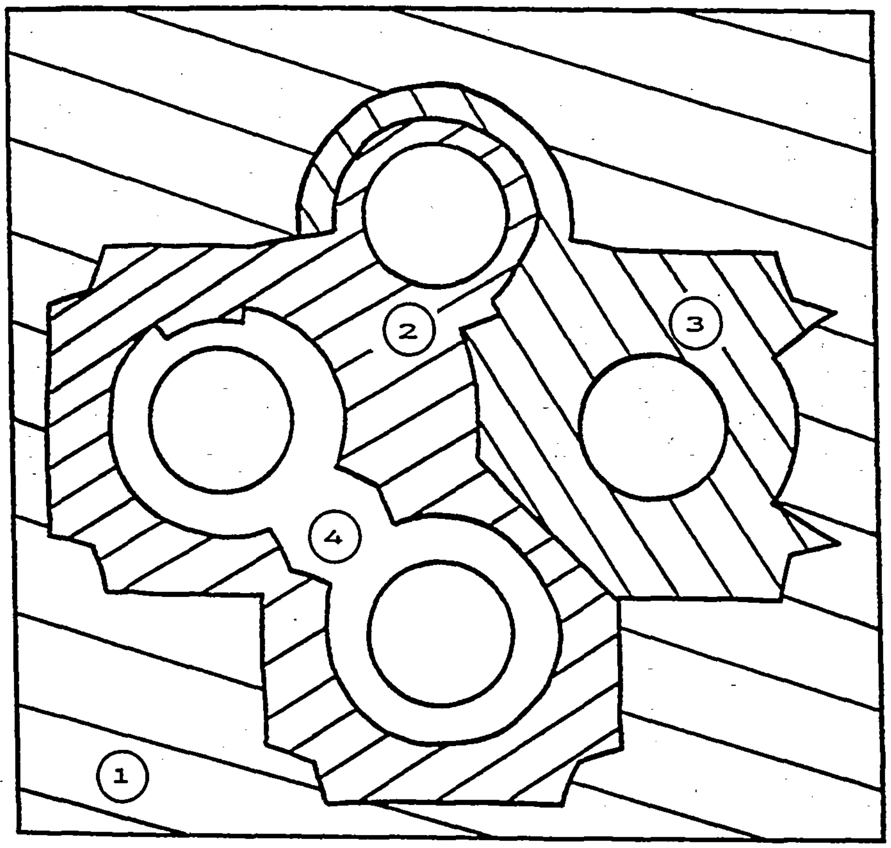


FIGURE 5.7



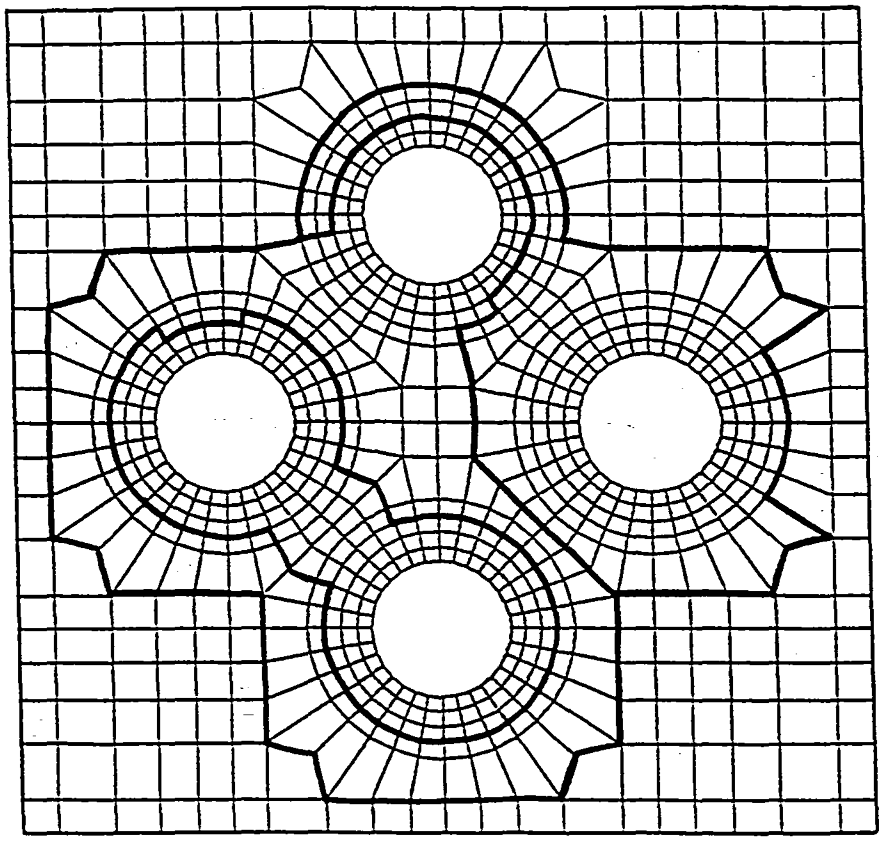


FIGURE 5.8

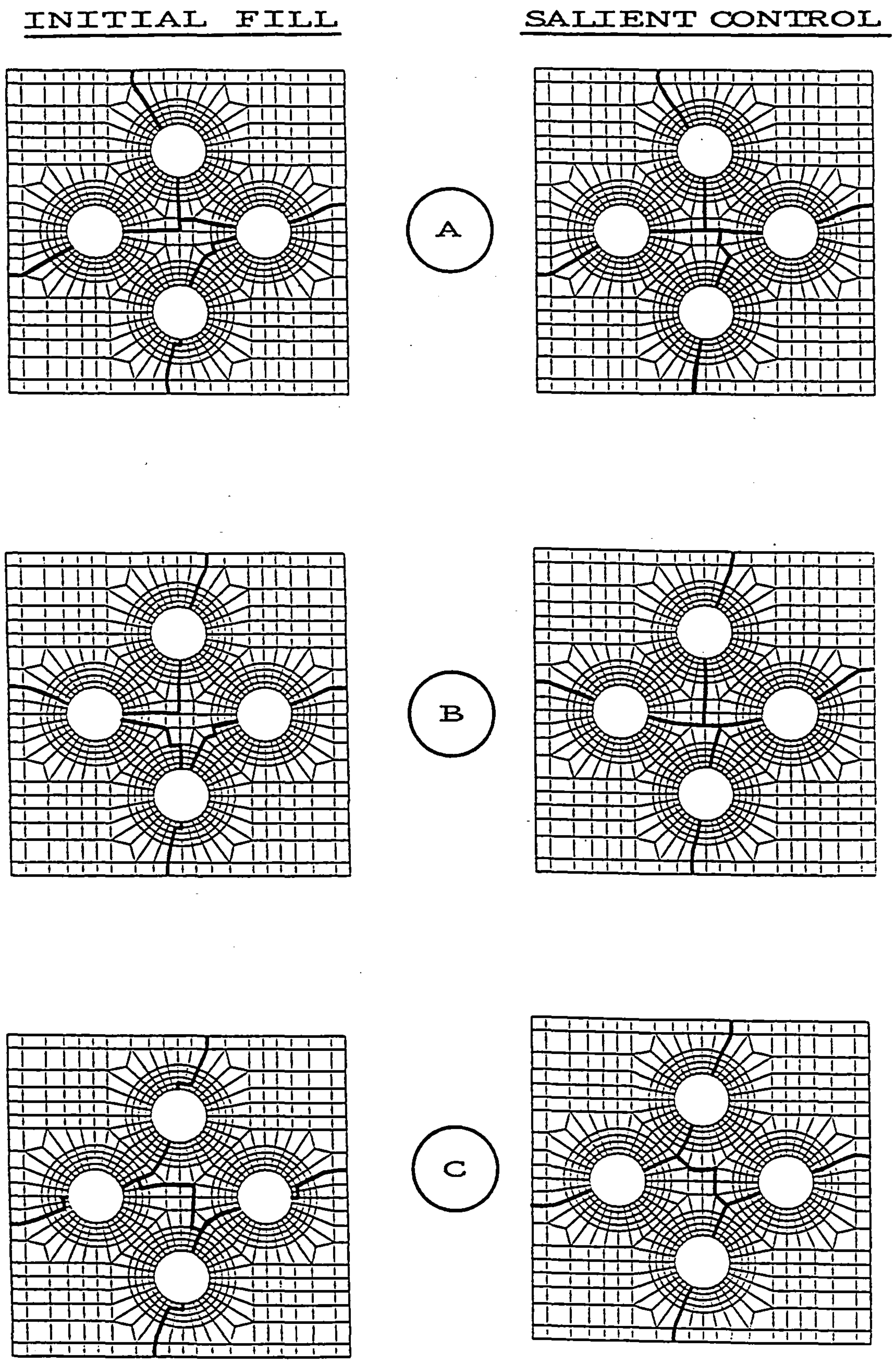
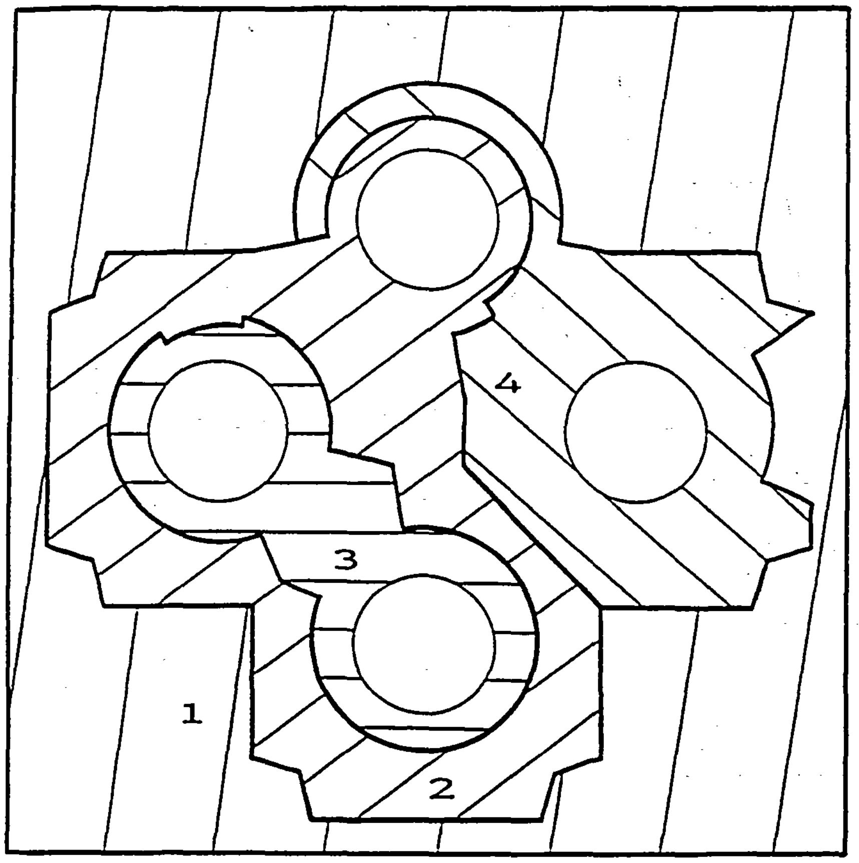
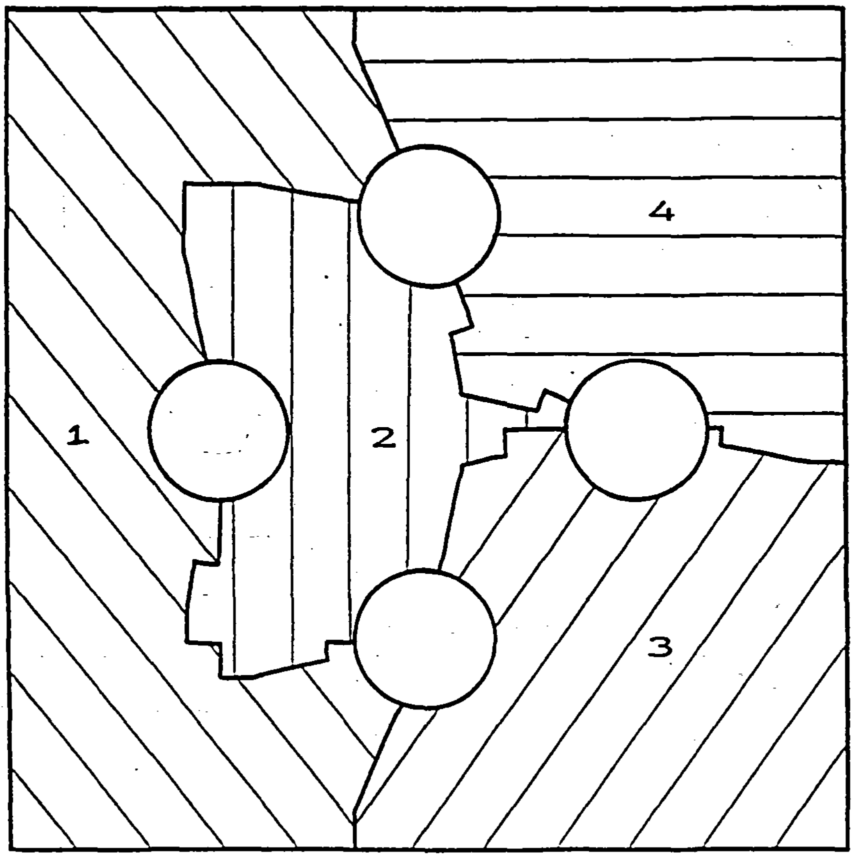


FIGURE 5.9

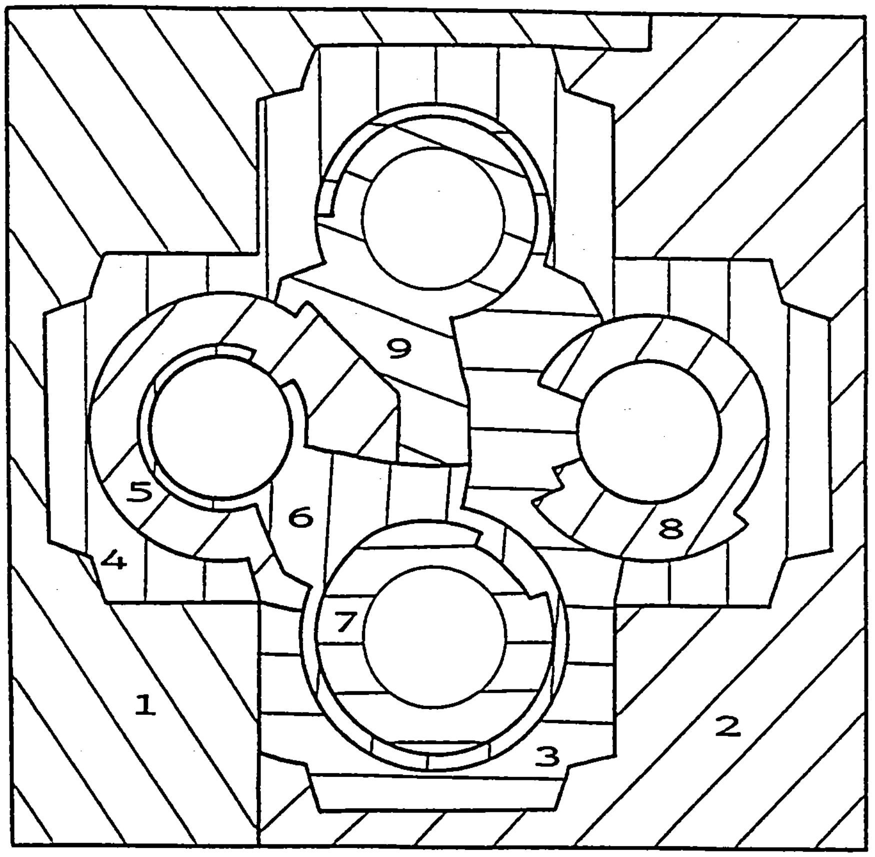


CNF



DECF

FIGURE 5.10



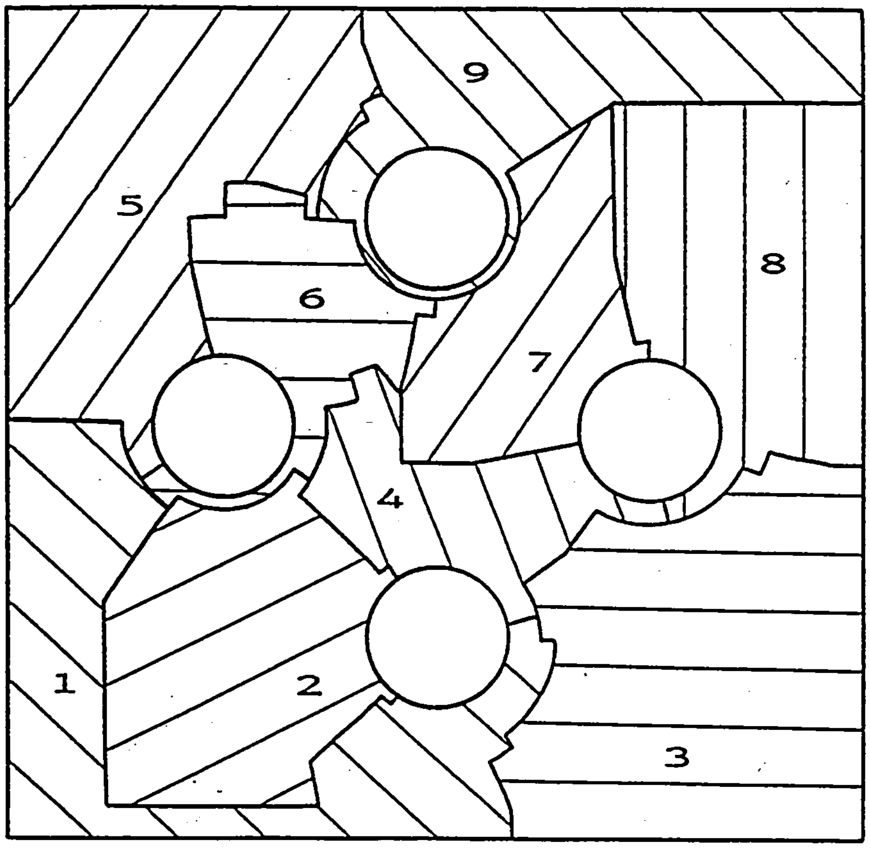


FIGURE 5.11

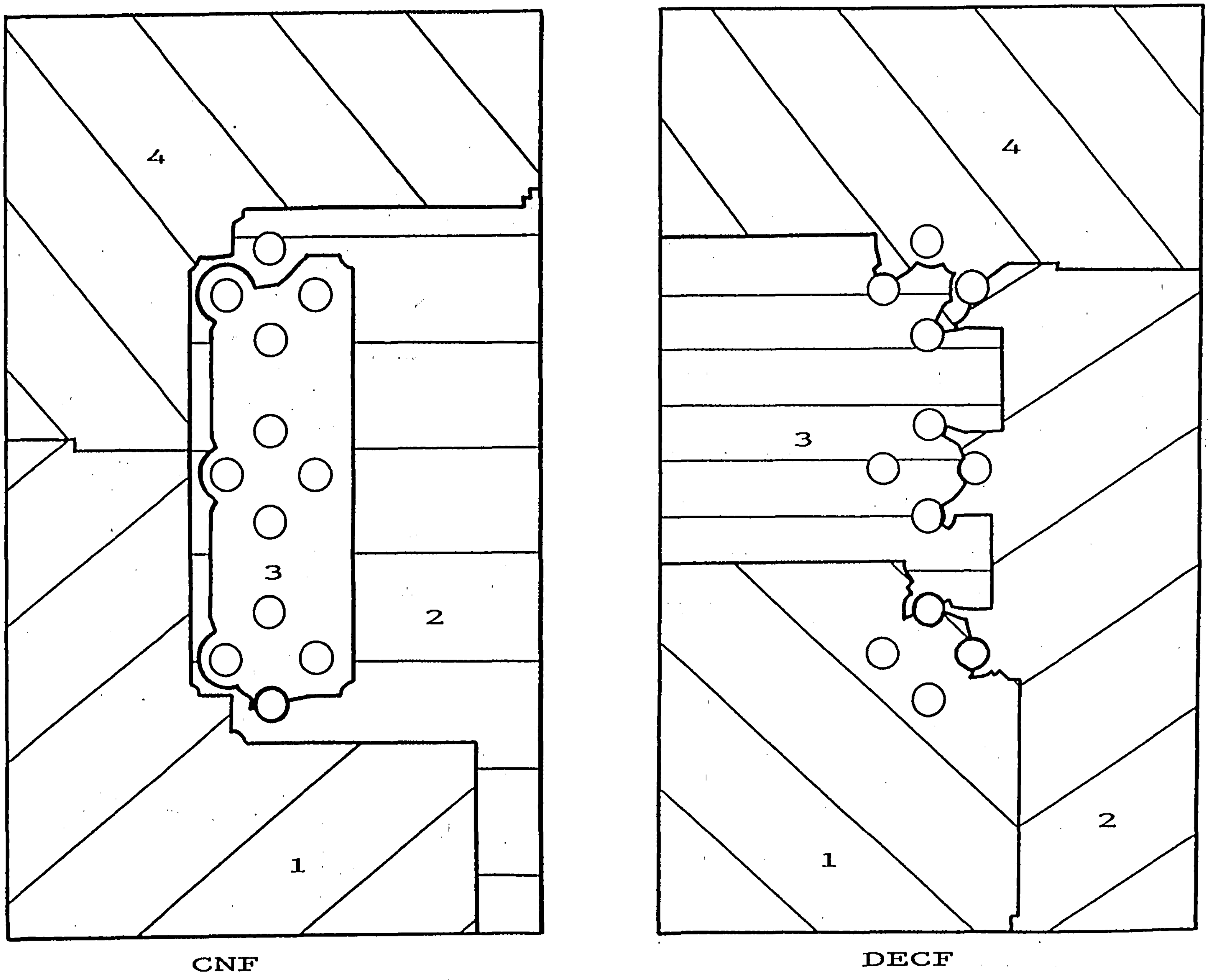
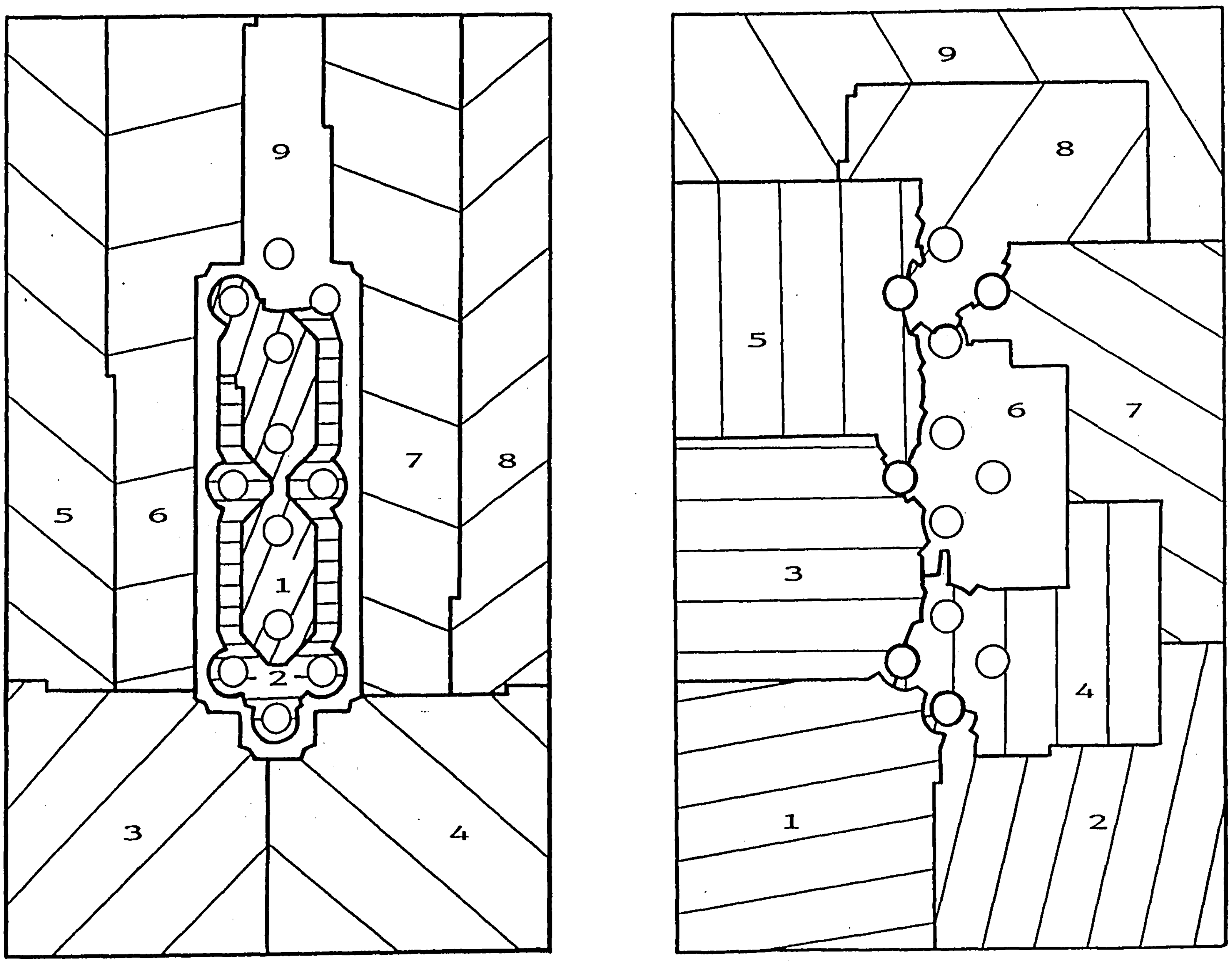


FIGURE 5.12



CNF DECF

FIGURE 5.13

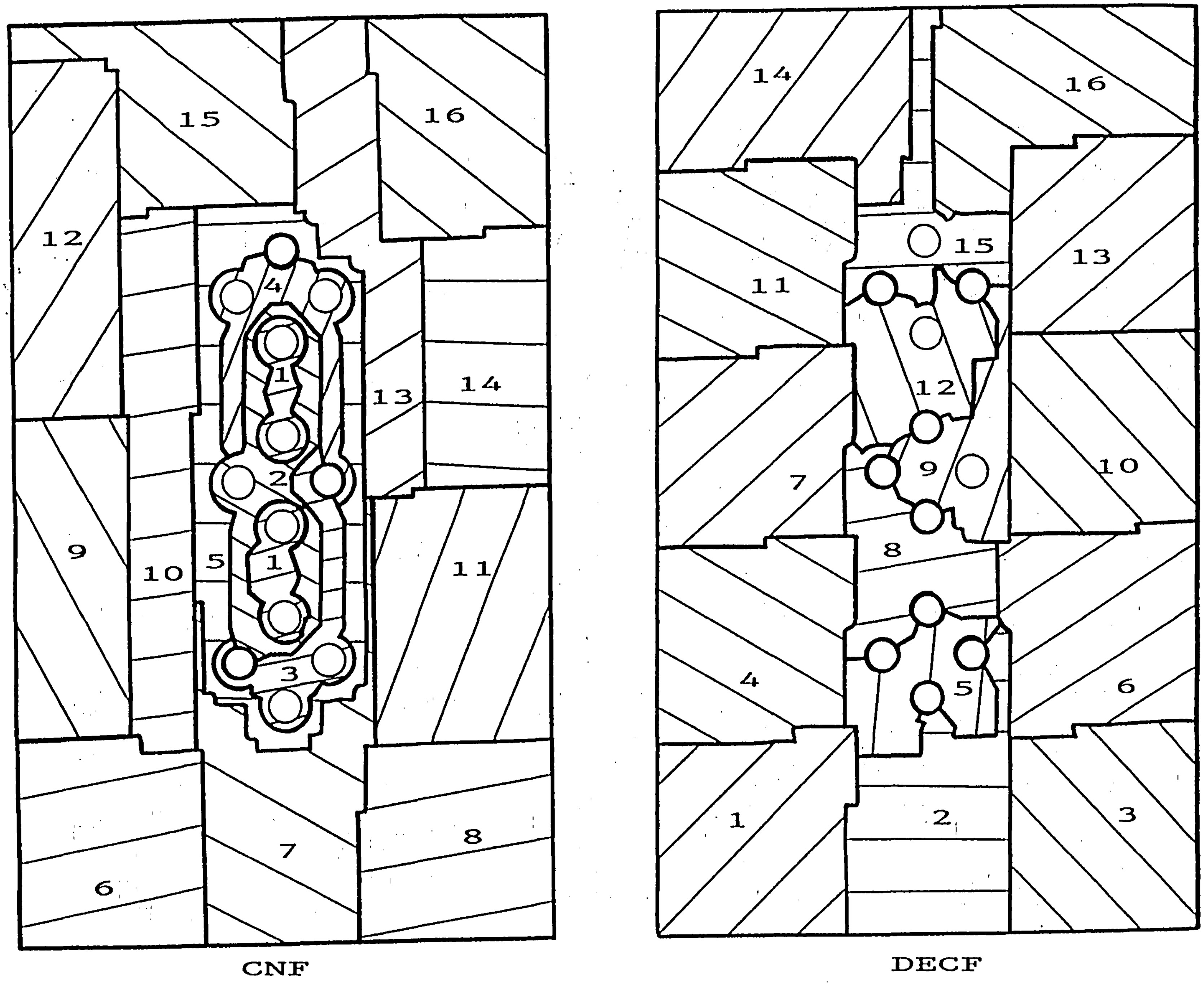

15
16
12
4
1
13
14
2
9
10
5
1
11
3
6
7
8
CNF
14
16
15
11
13
12
7
9
10
8
4
5
6
1
2
3
DECF

FIGURE 5.14

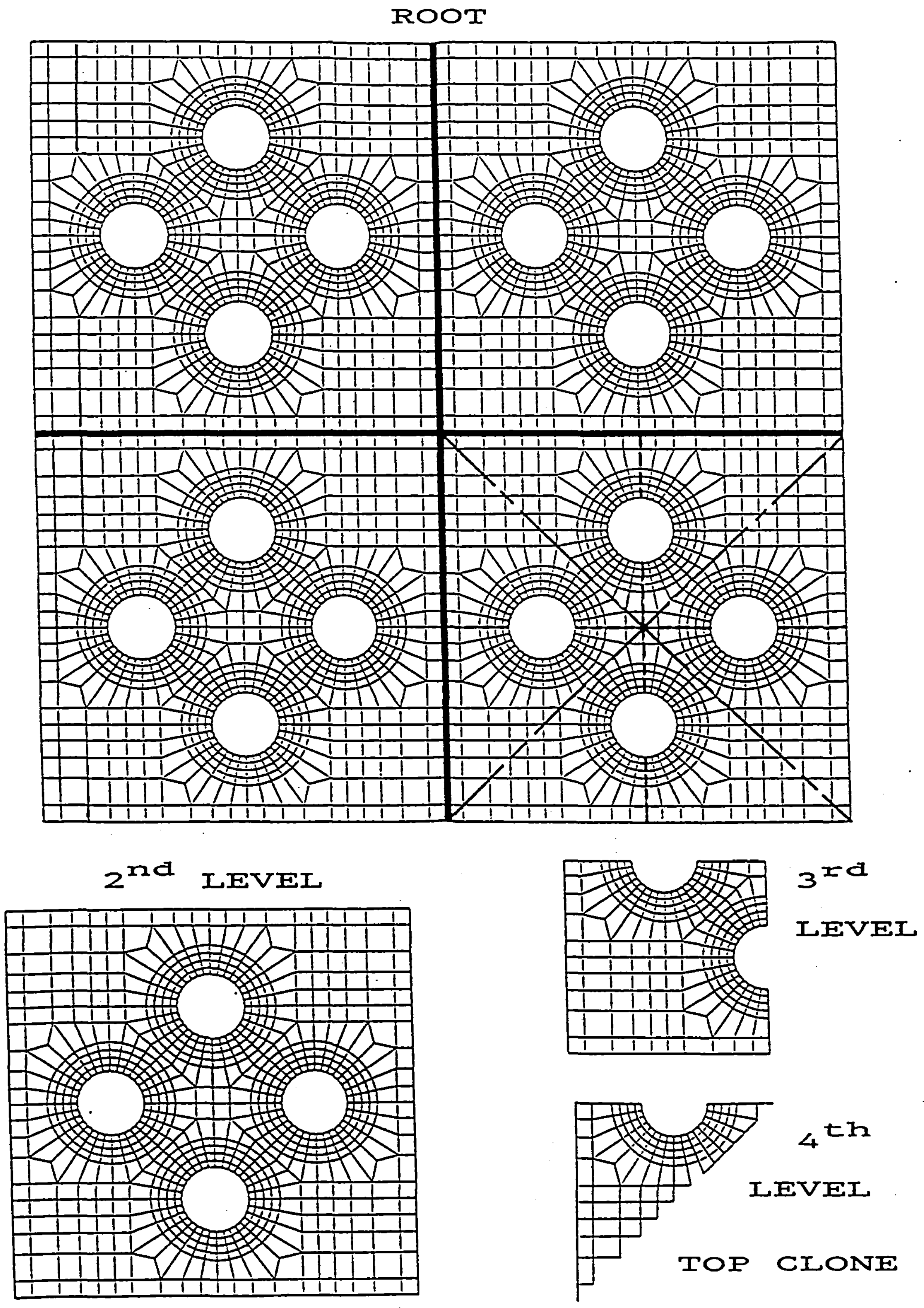


FIGURE 5.15

Load (KIPS)	1	2	4	6	8	10
U-U	4/3	5/3	5/3	6/4	6/4	6/4
BFGS-U	6/4	9/4	11/7	16/9	29/15	43/17
N-U	10/5	17/8	*	*	*	*
BFGS-N	9/5	14/8	*	*	*	*
N-N	12/5	34/15	*	*	*	*
Auto	4/3	5/3	5/3	6/4	6/4	6/4

*Failed to converge in 50 iterations

Table 5.1

Load (KIPS)	1	2	4	6	8	10
U-U	4/3	5/3	5/3	6/3	6/4	7/4
BFGS-U	7/4	9/6	12/7	16/9	27/19	30/21
N-U	9/4	17/8	*	*	*	*
BFGS-N	13/5	20/10	*	*	*	*
N-N	12/6	33/15	*	*	*	*
Auto	4/3	5/3	5/3	6/3	6/4	7/4

*Failed to converge in 50 iterations

Table 5.2

10 (KIPS) Target	Number of Load Increments					
	2	4	6	8	10	20
U	7/17	10/25	13/26	15/30	18/36	32/64
BFGS	*	10/30	13/42	15/50	18/38	32/64
N	*	*	13/61	15/52	18/64	32/64
U-U	7/17	10/25	13/26	15/30	18/36	32/64
N-U	*	10/45	13/42	15/52	18/38	32/64
N-N	*	*	13/61	15/52	18/64	32/64
BFGS-U	*	10/30	13/26	15/30	18/36	32/64
Auto	7/17	10/25	13/26	15/30	18/36	32/64

*Failed to converge in 50 iterations

Table 5.3.

Matrix Size	Number of Processors					
	2	3	4	5	6	7
225	93.66	86.22	80.28	76.58	72.36	67.88
450	95.43	90.22	87.62	79.72	79.34	73.69
500	96.84	92.30	89.24	82.20	81.96	74.93
600	96.65	93.02	87.68	81.38	82.10	76.33
700	95.99	92.05	87.25	81.21	80.81	76.52
800	97.76	92.89	88.20	86.51	81.60	80.68
900	97.12	91.68	89.15	85.65	81.31	78.02
1000	97.93	92.68	88.69	84.46	82.16	78.11
1100	96.55	92.89	86.12	84.18	81.99	78.41
1200	96.49	92.04	87.56	83.38	80.25	76.32
1500	95.69	90.90	85.21	82.54	78.82	74.58
2000	96.15	90.39	84.97	81.68	76.23	71.73

TABLE 5.4

N	L	S	M_R	K_i							
				2	3	4	5	6	7	8	9
50	2	3.75	1.98	3							
50	3	5.01	2.25	2	3						
50	4	5.43	2.16	2	2	2					
50	5	5.44	1.99	2	2	2	2				
100	2	5.88	2.77	4							
100	3	8.78	3.24	2	3						
100	4	10.28	3.60	2	2	3					
100	5	10.71	3.48	2	2	2	2				
100	6	10.71	3.27	2	2	2	2	2			
1000	2	26.10	7.45	9							
1000	3	60.14	12.57	3	6						
1000	4	82.54	16.56	2	3	5					
1000	5	95.88	19.65	2	2	3	4				
1000	6	101.87	21.31	2	2	2	3	3			
1000	7	104.58	21.58	2	2	2	2	2	3		
1000	8	105.24	21.41	2	2	2	2	2	2	2	
1000	9	105.34	20.88	2	2	2	2	2	2	2	2

N-problem size

L-number of levels

S-speedup

M_R-memory reduction

K_i-number of partitions /i^th level/ per edge

TABLE 5.5

N	L	P	S	M_R	K_i 2	3	4	5	6	7	8	9
50	2	4	15.00	1.98	3							
50	3	4	20.04	2.25	2	3						
50	4	4	21.72	2.16	2	2	2					
50	5	4	21.76	1.99	2	2	2	2				
100	2	4	23.52	2.77	4							
100	3	4	35.12	3.24	2	3						
100	4	4	41.12	3.60	2	2	3					
100	5	4	42.76	3.48	2	2	2	2				
100	6	4	42.84	3.27	2	2	2	2	2			
1000	2	4	104.40	7.45	9							
1000	3	4	240.56	12.57	3	6						
1000	4	4	330.16	16.56	2	3	5					
1000	5	4	383.52	19.65	2	2	3	4				
1000	6	4	407.48	21.31	2	2	2	3	3			
1000	7	4	418.32	21.58	2	2	2	2	2	3		
1000	8	4	420.96	21.41	2	2	2	2	2	2	2	
1000	9	4	421.36	20.88	2	2	2	2	2	2	2	2

N-problem size

L-number of levels

P-number of processors

S-speedup

M_R-memory reduction

K_i-number of partitions /i^{th} level/per
per edge

TABLE 5.6

Number of Partitions	Effort Ratio (DECF/CNF)
<u>Model 1: Fig. 5.6</u> 4	.398
9	.25
<u>Model 2: Fig. 5.7</u> 4	.78
9	.416
16	.327

Table 5.7.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE June 1994	3. REPORT TYPE AND DATES COVERED Final Contractor Report

4. TITLE AND SUBTITLE Hierarchically Parallelized Constrained Nonlinear Solvers with Automated Substructuring	5. FUNDING NUMBERS WU–505–63–5B G–NAG–3–1142
6. AUTHOR(S) Joe Padovan and Abel Kwang	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) The University of Akron Department of Mechanical and Polymer Engineering Akron, Ohio 44325	8. PERFORMING ORGANIZATION REPORT NUMBER E–8615
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135–3191	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA CR–194474

11. SUPPLEMENTARY NOTES
Project Manager, Christos C. Chamis, Structures Division, organization code 5200, NASA Lewis Research Center, (216) 433–3252.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 39	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This paper develops a parallelizable multilevel multiple constrained nonlinear equation solver. The substructuring process is automated to yield appropriately balanced partitioning of each succeeding level. Due to the generality of the procedure, both sequential, partially and fully parallel environments can be handled. This includes both single and multi processor assignment per individual partition. Several benchmark examples are presented. These illustrate the robustness of the procedure as well as its capability to yield significant reductions in memory utilization and calculational effort due both to updating and inversion.

14. SUBJECT TERMS Sequential; Partial; Full assignments; Partitioning; Benchmark examples; Multiples; Multi-level; Constraints; Progressive substructuring; Mixed iterations.	15. NUMBER OF PAGES 86
	16. PRICE CODE A05

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT