

# Advanced Analysis of Steel Frames Using Parallel Processing and Vectorization

C. M. Foley\*

Department of Civil and Environmental Engineering, Marquette University,  
1515 W. Wisconsin Avenue, Milwaukee, Wisconsin, 53233, USA

**Abstract:** *Advanced methods of analysis have shown promise in providing economical building structures through accurate evaluation of inelastic structural response. One method of advanced analysis is the plastic zone (distributed plasticity) method. Plastic zone analysis often has been deemed impractical due to computational expense. The purpose of this article is to illustrate applications of plastic zone analysis on large steel frames using advanced computational methods. To this end, a plastic zone analysis algorithm capable of using parallel processing and vector computation is discussed. Applicable measures for evaluating program speedup and efficiency on a Cray Y-MP C90 multiprocessor supercomputer are described. Program performance (speedup and efficiency) for parallel and vector processing is evaluated. Nonlinear response including postcritical branches of three large-scale fully restrained and partially restrained steel frameworks is computed using the proposed method. The results of the study indicate that advanced analysis of practical steel frames can be accomplished using plastic zone analysis methods and alternate computational strategies.*

## 1 INTRODUCTION

The two analytical methods commonly used by researchers for the inelastic analysis of structural steel frameworks are plastic hinge models and distributed plasticity (plastic zone) models. The basic difference between these two approaches lies in the manner that yielding within members is mod-

eled. In the (simple) plastic hinge technique, the member (commonly a single finite element) is assumed to remain fully elastic between its ends, and it is often assumed that full (abrupt) plastification occurs at some level of incrementally applied load. Once cross-sectional yielding has been detected, an analytical hinge (moment-free) is placed at the yielded end(s). Yielding is often defined as the combination of moment and axial forces lying on a user-defined yield surface.<sup>44</sup> In the distributed plasticity (plastic zone) method of analysis, the cross section of the member (finite element) most often is broken down into fibers to model yielding penetration within the cross section, while many finite elements are used along each frame member's length to simulate along-the-length yielding. In general, it has been recognized that distributed plasticity analysis is the most accurate means with which to analytically predict frame strength (short of a complete-shell finite-element model of the entire building structure). Therefore, the distributed plasticity model often has been chosen to benchmark other analytical techniques.<sup>55,57</sup>

Recently, an increased emphasis on the development of advanced frame-analysis methods has occurred. Excellent discussions related to advanced analysis and advanced analysis-based design procedures are available in the literature.<sup>20,23,59</sup> Advanced analysis can result in more economical building structures through explicit consideration of force redistribution within a structural framework.<sup>58</sup> In lieu of implementing member-by-member design procedures using specifications and linear elastic design analysis, advanced analysis-based design relies on advanced analytical models that include the same physical behavior assumed in the development of the member strength checks contained in specifications.<sup>16</sup> The requirements for an advanced design analysis procedure can be quite demanding. In

\*E-mail: [chris.foley@marquette.edu](mailto:chris.foley@marquette.edu).

general, an advanced analysis must satisfy the following criteria<sup>52</sup>:

1. It must consider the presence of residual stresses within the members.
2. It must accurately assess overall framework strength.
3. It must consider framework imperfections (out-of-plumb stories) and member imperfections (out of straightness).

Simple plastic hinge-based analytical models generally do not satisfy the requirements of advanced analysis. Distributed plasticity analysis, on the other hand, is considered to satisfy advanced analysis requirements, albeit the modeling of frame out of plumb and member out of straightness must be done through modified nodal locations in the typical matrix structural analysis technique.

Although distributed plasticity models for inelastic analysis have been present in the literature since the early 1970s,<sup>1,17,21,22,38,43,56</sup> their widespread use in large-scale structural engineering problems has been limited. At the time of its initial implementation, plastic zone analysis required computational resources that simply were not available. As a result, a large number of research efforts have been undertaken to develop modifications to the simple plastic hinge methods of analysis to improve their accuracy to the level needed to qualify as advanced analysis.<sup>19,37,41,42</sup> As computational power began to increase, plastic zone and plastic hinge methods of analysis began to be reconsidered as a design tool.<sup>62,64,65</sup> In these studies, however, the plastic zone implementation was relegated to small structures and was considered a benchmark for plastic hinge-based analysis using yield-surface models. Recently, interest in plastic zone analysis of three-dimensional frameworks has been rekindled.<sup>54</sup> However, as in previous studies, this most recent effort again limits results to very small structures.

Although the computational inefficiency of plastic zone analysis may have been overstated in past research efforts,<sup>54</sup> the major hurdle limiting the use of advanced analysis with distributed plasticity remains computational expense. As one may imagine (and as will be seen in this article), the plastic zone analytical model for a large framework can require extensive computational resources. Many of the research efforts undertaken subsequent to 1970 involving distributed plasticity models have involved very small structures. Use of plastic zone analysis as a practical advanced analysis/design tool has not been considered.

The requirements of linear static, nonlinear static, and transient analysis of large structural systems have motivated the development of alternate computational strategies rather than exclusive development of new analytical models. The most common and useful strategy to come of age has been the implementation of structure partitioning (substructuring) in conjunction with parallel processing

on fine- and coarse-grained multiprocessor computers. An exhaustive literature review of all the advances in computational technology in this arena is not warranted in this article. However, excellent reviews of past research efforts in applications of substructuring, parallel processing, and vectorization in computational mechanics are available.<sup>2,6,15,26,28,60</sup>

Much of the past research in applications of substructuring and parallel processing have emphasized large-structure design optimization,<sup>3,7,49,51</sup> the elastic analysis of large structures,<sup>4,5,8,9,31,32</sup> nonlinear transient analysis,<sup>35,53</sup> nonlinear finite-element analysis,<sup>36</sup> and implementation of parallel equation solvers. The neural dynamics model has been implemented recently in the optimal design of large structures.<sup>45,46</sup> This model is an extension of the neural network algorithm, and its attractiveness, with respect to high-performance computing, comes from its ability to generate a globally convergent dynamic system by casting the constraint functions in the form of a Lyapunov function.<sup>11</sup> The neural dynamics approach has been shown to be an effective tool for the optimal design of large structures when used in conjunction with parallel processing.<sup>46</sup> The integration of structural control with optimal structural design also has been a fertile area for application of high-performance computing. Parallel and vector algorithms for large-structure weight and control force minimization have been developed.<sup>12,48</sup> Several additional sources of information summarizing the most recent advances in high-performance computing are available.<sup>10,11,13,14</sup>

Although an abundance of research has been performed in a wide variety of areas related to high-performance computing in structural engineering, the inelastic nonlinear analysis of large structural steel frameworks using parallel processing, vectorization, and substructuring has yet to receive attention. Application of the divide and conquer analytical paradigm in conjunction with parallel processing shows promise in the realm of inelastic stability analysis. These advanced computational techniques have yielded significant speedup and efficiency for the elastic stability analysis of multistory steel frames on a Cray Y-MP supercomputer.<sup>31,32</sup> The benefits of parallel processing and vectorization that are attained in nonlinear elastic analysis most certainly can be realized in the inelastic analysis of steel-frame structures. As a result of the state determination (e.g., current connection stiffness, member yielding) required at each increment (or iteration) in loading, parallel processing perhaps can result in even more efficient computation and speedup in the inelastic analysis of large partially and fully restrained steel frameworks.

Some of the most promising (in terms of design office) applications of concurrent (parallel) processing come in the application of distributed computing on networked workstations using parallel virtual machine technology.<sup>8,9</sup> Furthermore, at the time of this writing, multiprocessor computers

are available to the public (not just researchers at supercomputing centers). In addition, there are C, C++, and Fortran compilers capable of creating multithreaded code for multiprocessor PCs. Unfortunately, the same cannot be said about vector processing on personal computers. Therefore, research in applications of parallel processing is needed to ensure that advanced analytical techniques such as distributed plasticity can be implemented by both researchers and practicing engineers on structures of practical size and complexity. This, in turn, will allow the design of safe, reliable, and economical structures, as well as more accurate evaluation of existing structures in need of retrofit or repair.

This article seeks to present an advanced frame-analysis algorithm with which parallel processing and vector computation can be implemented in the advanced analysis of large-scale structural steel frames. To this end, a finite element capable of modeling spread of plasticity, residual stresses, shift in the neutral axis of the elastic core, and partially restrained connections is described. A method of nonlinear solution using the work-control technique<sup>47,61</sup> is discussed relative to the computation of ultimate loads (including postcritical response) of partially and fully restrained steel frames. The computer program developed in this study uses substructuring in conjunction with parallel processing and vectorization and is implemented on a Cray

Y-MP C90 multiprocessor supercomputer. Segments of the nonlinear solution algorithm where parallel processing and vectorization is applied are discussed. Speedup and efficiency of the computer program are evaluated. In this manner, it is hoped that this article will foster a renewed effort in the examination of new computational strategies for widespread implementation of advanced structural analysis in addition to the present ongoing development of new analytical representations to handle the ever-increasing demands of more complicated analysis procedures.

### 2 FINITE-ELEMENT MODEL

Much of the past research discussed in the preceding section involved frame and truss analysis that assumed elastic material behavior. One of the goals of this study is to demonstrate the feasibility of collapse load analysis for large structures using distributed plasticity. A brief outline of the present finite-element formulation will give the reader an appreciation for the level of computational complexity contained in this analysis and a more complete understanding of the algorithm used in the solution.

A planar beam-column finite element is used as the basis for this study. The typical 6-degree-of-freedom element is shown in Figure 1(a), where  $d_{se,i}$  are the displacements

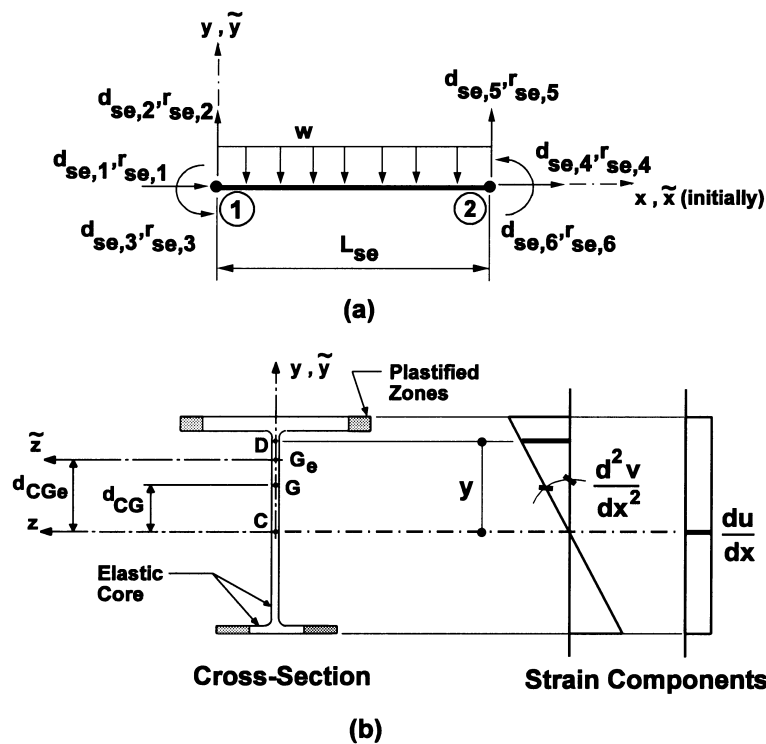


Fig. 1. Beam or column finite-element modeling: (a) typical finite element used for framework analysis; (b) partial plastification within cross section and strain components.

at the ends (1 or 2) of the subelement, and  $r_{se,i}$  are the subelement end forces (e.g., consistent end forces; fixed end actions are present as well). It should be noted that the finite element given in Figure 1(a) is one of many subelements used to model any beam or column member. The length of this subelement is given by  $L_{se}$ . Member loading is considered to be uniformly distributed for the present discussion, although concentrated subelement member loads are possible.<sup>30</sup>

Assuming elastic–perfectly plastic material behavior (unloading occurs with the initial modulus, and strain hardening is neglected), the strain energy for a subelement can be written as

$$U = \frac{E}{2} \int_{L_{se}} \int_{A_e} \epsilon^2 dA_e dx + \sigma_y \int_{L_{se}} \int_{A_p} \epsilon dA_p dx - \frac{1}{2} \sigma_y \epsilon_y \int_0^{L_{se}} \int_{A_p} dA_p dx \quad (1)$$

where  $A_e$  and  $A_p$  are the elastic and plastified portions of the subelement cross-sectional area, respectively,  $\epsilon(\sigma)$  is the current strain (stress) at any fiber within the cross section, and  $\epsilon_y(\sigma_y)$  is the yield strain (stress) for a fiber. Normal (and incremental) strain, with reference to the non-moving  $x, y, z$  coordinate system shown in Figure 1(b), may be written using superposition of axial and bending strains (assuming elastic behavior within a load increment):

$$\epsilon = \frac{du}{dx} + \frac{1}{2} \left( \frac{dv}{dx} \right)^2 - y \frac{d^2v}{dx^2} \quad (2)$$

where  $u$  is a function describing longitudinal displacement along the subelement,  $v$  is a function describing transverse displacement, and  $dx$  is an infinitesimal length in the longitudinal direction. Substituting Equation (2) into Equation (1) and integrating over the area give the strain energy for the partially plastified element:

$$U = \frac{E}{2} \int_0^{L_{se}} \left[ A_e \left( \frac{du}{dx} \right)^2 - 2S_{ze} \left( \frac{d^2v}{dx^2} \right) \left( \frac{du}{dx} \right) + I_{ze} \left( \frac{d^2v}{dx^2} \right)^2 \right] dx + \frac{E}{2} \int_0^{L_{se}} \left[ A_e \left( \frac{du}{dx} \right) \left( \frac{dv}{dx} \right)^2 - S_{ze} \left( \frac{d^2v}{dx^2} \right) \left( \frac{dv}{dx} \right)^2 + \frac{A_e}{4} \left( \frac{dv}{dx} \right)^4 \right] dx + \int_0^{L_{se}} \left[ \int_{A_p} \sigma_y \left( \frac{du}{dx} \right) dA_p + \int_{A_p} \frac{1}{2} \sigma_y \left( \frac{dv}{dx} \right)^2 dA_p \right] dx - \int_0^{L_{se}} \left[ \int_{A_p} \sigma_y y \left( \frac{d^2v}{dx^2} \right) dA_p + \frac{1}{2} \int_{A_p} \sigma_y \epsilon_y dA_p \right] dx \quad (3)$$

The first moment of area, second moment of area, and cross-sectional area terms are based on the remaining elastic core with reference to the fixed coordinate system

shown in Figure 1(b). The nonmoving coordinate system allows for unsymmetric yielding within the cross section to be tracked and the eccentricity of applied loads (whose point of application remains in the fixed system) with respect to the remaining elastic core to be included. The cross-sectional properties may be defined with the aid of Figure 1(b) and can be expressed as the following integrals:

$$A_e = \int_{A_e} dA_e \quad S_{ze} = \int_{A_e} y dA_e = A_e d_{CG_e}$$

$$I_{ze} = \int_{A_e} y^2 dA_e \quad A_p = \int_{A_p} dA_p$$

where  $d_{CG_e}$  is the distance that locates the centroid of the elastic core, point  $G$  denotes the initial centroid location for an initially unsymmetric cross section, and point  $G_e$  denotes the centroid of the remaining elastic core. For a symmetric cross section, points  $G$  and  $C$  are coincident. The generalized stress relations for the plastified areas may be written as

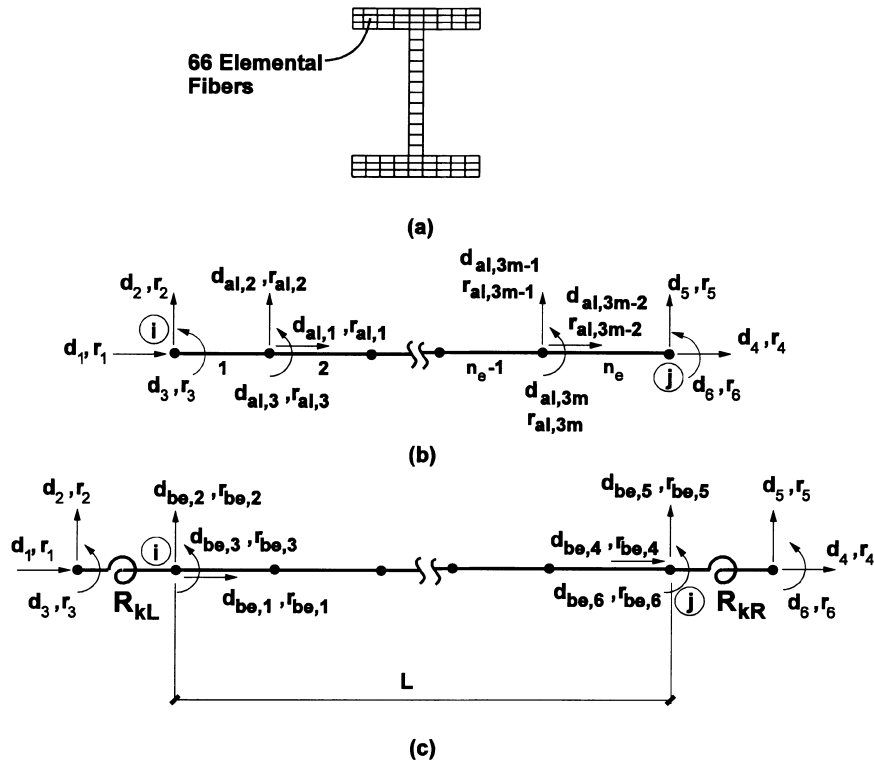
$$P_{A_p} = \int_{A_p} \sigma_y dA_p \quad M_{A_p} = \int_{A_p} \sigma_y y dA_p$$

The total potential energy for the subelement indicated in Figure 1(a) can be written as

$$\Pi = \frac{E}{2} \int_0^{L_{se}} \left[ A_e \left( \frac{du}{dx} \right)^2 - 2S_{ze} \left( \frac{d^2v}{dx^2} \right) \left( \frac{du}{dx} \right) + I_{ze} \left( \frac{d^2v}{dx^2} \right)^2 \right] dx + \frac{E}{2} \int_0^{L_{se}} \left[ A_e \left( \frac{du}{dx} \right) \left( \frac{dv}{dx} \right)^2 - S_{ze} \left( \frac{d^2v}{dx^2} \right) \times \left( \frac{dv}{dx} \right)^2 + \frac{A_e}{4} \left( \frac{dv}{dx} \right)^4 \right] dx \quad (4)$$

$$+ \int_0^{L_{se}} \left[ P_{A_p} \left( \frac{du}{dx} \right) + \frac{P_{A_p}}{2} \left( \frac{dv}{dx} \right)^2 - M_{A_p} \left( \frac{d^2v}{dx^2} \right) - \frac{P_{A_p}}{2} \epsilon_y \right] dx + \int_0^{L_{se}} wv dx - \{r_{se}\}^T \{d_{se}\}$$

The cross-sectional properties and generalized stress relations represented in Equation (4) are numerically evaluated using a fiber-element model. Sixty-six fiber elements as shown in Figure 2(a) are used within the cross section of each subelement. This fiber configuration analytically allows for accurate modeling of cross-sectional yielding in addition to assignment of initial stress states corresponding to a variety of residual stress patterns.<sup>30</sup> All cross-sectional properties are computed using a fiber-element



**Fig. 2.** Distributed plasticity modeling: (a) fiber-element model; (b) subfinite elements used for along-the-length distributed plasticity modeling; (c) attachment of partially restrained connections.

model as shown in Figure 2(a) and the fixed coordinate system shown in Figure 1(b) as a reference.

The Rayleigh-Ritz method and the principle of stationary potential energy are used in the development of incremental equilibrium equations for an inelastic subelement. Although the subelement lengths can be relatively short, cubic shape functions (Hermite-type) are used for transverse deformations. Axial deformations are interpolated using linear functions. The secant elemental stiffness equations that result are

$$\{r_{se}\} = [K_{se}]\{d_{se}\} + \{FEA_{se}\} + \{r_P\}_{se} \quad (5)$$

where  $\{FEA_{se}\}$  is a vector of consistent subelement end loads,  $\{r_P\}_{se}$  is a vector of nodal loads resisted by the plastified portions of the subelement cross section,  $[K_{se}]$  is the subelement stiffness, and  $\{d_{se}\}$  is a vector of nodal displacements. The secant stiffness matrix  $[K_{se}]$  for the subelement is written as

$$[K_{se}] = [K_0] + \frac{EA_e}{2}[K_1] + \frac{EA_e}{3}[K_2] + [K_3] \quad (6)$$

Each component stiffness matrix is defined as follows:  $[K_0]$  is a stiffness matrix containing linear terms,  $[K_1]$  and  $[K_2]$  are nonlinear stiffness contributions as a result of the equilibrium equations being written for the deformed configuration, and  $[K_3]$  is a nonlinear stiffness matrix that results

from the interaction of the axial force resisted by the plastified portions of the cross section acting in conjunction with the nodal deformations. All terms for the subelement stiffness matrices and further details of the derivation are available.<sup>30</sup>

A truncated Taylor series expansion of the subelement equilibrium equations can be used to develop the subelement tangent stiffness, given by

$$[K_T]_{se} = [K_0] + EA_e[K_1] + EA_e[K_2] + [K_3] \quad (7)$$

where the component matrices are the same as those given in Equation (6). It should be noted that the vectors of fixed end actions and plastification forces drop out of the incremental equations as a result of the partial differentiation. This implies that these quantities do not change over an incremental load step.

Along-the-length yielding of the frame members (beams and beam columns) is simulated by modeling each beam and column member with many of the formerly described subelements. This modeling technique is shown schematically in Figures 2(b) and 2(c). Both beam and column members are modeled using  $n_e$  subelements, and  $m$  denotes the number of the final internal subelement node. Internal member deformations  $d_{al}$  and internal consistent nodal loads  $r_{al}$  are also indicated. Figure 2(b) illustrates the analytical element used for the typical column member. It

should be noted that column end displacements  $d_i$  and column consistent nodal loads  $r_i$  correspond to the beam-to-column joints in the analytical model. The beam analytical model is shown in Figure 2(c). Consistent member end forces and displacements at the beam ends are denoted as  $r_{be,i}$  and  $d_{be,i}$ , respectively. It should be noted that these forces and displacements are consistent with  $r_i$  and  $d_i$  for the column members. Therefore, within nodes  $i$  and  $j$ , both beams and columns are treated in the same manner during the analysis. A condensed stiffness and condensed consistent nodal load vector for the beam and column members is created. After condensation, a 6-degree-of-freedom element with two nodes ( $i$  and  $j$ ) is created.

An added manipulation to beam members during the analysis is now discussed with reference to Figure 2(c). Assembly of the condensed beam element (member end forces  $r_{be,i}$ , member end displacements  $d_{be,i}$ ) with two nonlinear spring elements ( $R_{kL}$  and  $R_{kR}$ ) is carried out. The nonlinear connections assumed in this study are modeled using a trilinear model that includes unloading.<sup>30,33,34</sup> This second condensation procedure results in a 6-degree-of-freedom beam element suitable for assembly into the building analytical model. It is up to the user to define the number of subelements for modeling beams and columns. It has been found that 10 elements for each beam and column are sufficiently accurate for a wide variety of member loadings.<sup>30</sup> This study uses more elements so that a detailed distribution of yielding along the members is modeled. Residual stresses within the member cross sections are assumed to be uniform tension in the web and a linear variation from flange tip compression to tension at the web-to-flange junction.<sup>39</sup>

The subelement assembly, condensation, and assembly of the linearized spring elements for the beam members within the analytical model can be summarized as follows. The reader should consult Figures 2(b) and 2(c) for notation. An element tangent stiffness matrix is formed using a user-determined number of subelements  $n_e$ :

$$[K_T]_m = \sum_{i=1}^{n_e} [K_T]_{se,i} \quad (8)$$

Omitting consistent nodal loads for this discussion, the incremental equations of equilibrium for a column or beam member can be written as

$$\{\Delta r_{al}\} = [K_T]_m \{\Delta d_{al}\} \quad (9)$$

The system implied in Equation (9) is (in general) much larger than the traditional six equations found in the typical planar beam-column finite element. It necessarily becomes larger because more refinement in the modeling of along-the-length yielding is desired. Static condensation is used to develop a reduced 6-degree-of-freedom system. Condensation of degrees of freedom for the column member

results in

$$\{\Delta r_i\} = [K_{T,C}]_m \{\Delta d_i\} = [K_{T,C}]_c \{\Delta d_i\} \quad (10)$$

while the condensation procedure for the beam elements gives

$$\{\Delta r_{be}\} = [K_{T,C}]_m \{\Delta d_{be}\} \quad (11)$$

where  $[K_{T,C}]_m$  is the condensed member tangent stiffness. The condensed incremental stiffness for the beam member given by Equation (11) must then be assembled with the two linearized spring elements having incremental connection stiffness  $R_{kL}$  and  $R_{kR}$  at the left and right ends, respectively. Condensation is again used to form a 6-degree-of-freedom combined beam-connection element. The incremental equations of equilibrium for this element are therefore

$$\{\Delta r_i\} = [K_{T,C}]_{bc} \{\Delta d_i\} \quad (12)$$

The tangent stiffness for the column element contained in Equation (10) and that of the beam element contained in Equation (12) are then used in the assembly of the structure stiffness as follows:

$$[K_T] = \sum_{i=1}^{\text{no. col.}} [K_{T,C}]_{c,i} + \sum_{i=1}^{\text{no. beam}} [K_{T,C}]_{bc,i} \quad (13)$$

The preceding discussion neglected to mention the consistent nodal loads with reference to the condensation. It should be noted that all consistent internal nodal loads  $r_{al}$  are also involved in the condensation procedures. In this study, only the beams are subjected to loading applied between the beam-to-column connections. These loads are also manipulated in the condensation procedure(s). The condensed consistent incremental nodal load vector for the beam elements is denoted  $\{\Delta r_C\}_m$  in this algorithm. These forces naturally would have to be added to Equation (12) to complete the incremental equations of equilibrium for the beam member.

### 3 SUBSTRUCTURING AND INELASTIC ANALYSIS

Substructuring is a means of breaking up an analytical model (in this case a steel building frame) into smaller components. It can be conceptualized as a divide and conquer strategy and has been found to be very popular in implementing parallel processing on coarse-grained parallel machine architectures. A schematic example of the method is given in Figure 3. A domain-decomposition algorithm<sup>27,29</sup> is used to partition the framed structure into subdomains. The subdomains and nodes are then numbered in a judicious manner, allowing parallel condensation of the structure stiffness equations.<sup>4,5,29</sup> The internal nodes within each substructure are numbered consecutively first, and

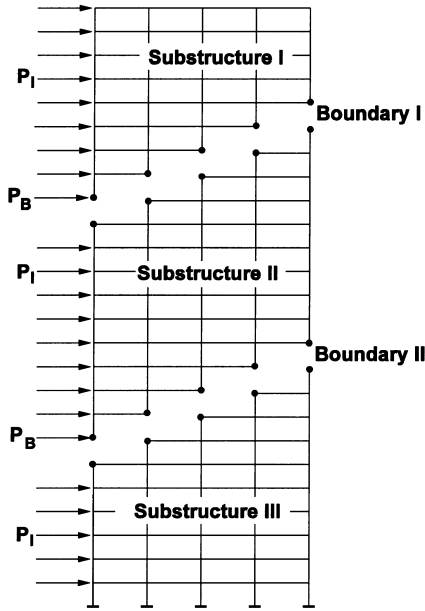


Fig. 3. Typical partitioning of multistory framework using substructuring.

the nodes along the substructure boundaries are numbered last. This node-numbering scheme (and therefore degree-of-freedom numbering) allows the internal degrees of freedom within subdomains to become independent (decoupled) from one another. As a result, parallel condensation of the internal subdomain degrees of freedom to the boundary degrees of freedom gives a reduced set of equilibrium equations to solve. This set of equilibrium equations is given by

$$\{R_B\}_{re} = [K_{T,B}]\{D_B\}_{re} \quad (14)$$

where  $\{R_B\}_{re}$  is a vector consisting of the reference nodal forces condensed to the subdomain boundaries,  $[K_{T,B}]$  is the condensed tangent stiffness matrix (at boundary), and  $\{D_B\}_{re}$  is the condensed reference nodal displacement vector for the subdomain boundaries. The *reference* label results from implementation of the constant-work algorithm to be discussed in greater detail later in this article.

Equation (14) requires that the condensed stiffness matrix and the condensed reference nodal load vector be formulated. The condensed tangent stiffness matrix is computed via

$$[K_{T,B}] = \sum_{j=1}^N \{ [K_{bb,T}] - [K_{ib,T}]^T [K_{ii,T}]^{-1} [K_{ib,T}] \}_j \quad (15)$$

where  $[K_{bb,T}]$  is the tangent stiffness matrix containing boundary terms for a substructure  $j$ ,  $[K_{ib,T}]$  is the tangent stiffness matrix representing the influence of the internal nodes on the boundary for substructure  $j$ ,  $[K_{ii,T}]$  is the tangent stiffness matrix containing terms that result from

the internal nodes in subdomain  $j$ , and  $N$  is the number of subdomains. The condensed reference nodal load vector for the structure is computed using

$$\{R_B\}_{re} = \sum_{j=1}^N (\{R_b\}_{re} - [K_{ib,T}]^T [K_{ii,T}]^{-1} \{R_i\}_{re}) \quad (16)$$

where  $\{R_i\}_{re}$  is the internal (off-boundary) reference nodal forces applied to subdomain  $j$ , and  $\{R_b\}_{re}$  is the reference nodal load vector at the boundary nodes for substructure  $j$ .

Once Equation (14) is solved for the reference displacements along the subdomain boundaries, the reference displacements at the nodes within each subdomain (off-boundary) need to be determined. The expansion of the subdomain reference displacements to the internal nodes of subdomain  $j$  is performed using

$$\{D_i\}_{re} = [K_{ii,T}]^{-1} (\{R_i\}_{re} - [K_{ib,T}]\{D_b\}_{re}) \quad (17)$$

where  $\{D_b\}_{re}$  are reference displacements located at the boundary of subdomain  $j$ .

The numbering scheme described means that the condensation of the subdomain stiffness matrices and nodal load vectors depicted in Equations (15) and (16) can take place concurrently on  $N$  processors. The final summation of the condensed stiffness matrix and nodal load vector shown in Equations (15) and (16) must be *synchronized* and, unfortunately, cannot be done in parallel. Therefore, bottlenecks can result in the algorithm depending on the number of nodes contained at the subdomain boundaries.<sup>31,32,35</sup> Lastly, the evaluation of the reference nodal displacements within each subdomain (off-boundary) as given in Equation (17) also can be performed in parallel. Therefore, the expansion phase of the equation solution procedure is fully parallel.

One should keep in mind that the condensation procedure necessarily incurs many additional operations when forming the subdomain stiffness and load vectors. These additional operations involved in Equations (15) through (17) approach (and may marginally exceed) the number of operations needed to solve the original unpartitioned system of equations. However, one also must keep in mind that multiple operations are being carried out on a parallel-architecture computer. Therefore, the number of operations carried out in the partitioned system will take less time on a parallel computer. The original motivation for subdomain partitioning procedures was computer memory limitations. In this analysis, partitioning is used to divide up the operations needed to solve the system of equations among different processes.

The inelastic nonlinear analysis of partially restrained structural steel building frames allows the incorporation of parallel processing on a much wider scale than pure linear elastic analysis. There are several advantageous locations to implement parallel processing in nonlinear finite-element

analysis.<sup>28,36</sup> In addition to formation of the substructure stiffness matrices, the method of substructuring allows parallel processing to be incorporated into the following areas, termed in this study *state determination* for the elements and connections. In the inelastic analysis of PR frameworks, state determination consists of (1) computation of the displacements at the end(s) of all beam members in the finite-element model, (2) computation of the displacements along the length of each member using the computed displacements at the structure nodes, (3) determination of the states of stress and strain in the fibers for each finite element contained in the substructures, (4) updating the cross-sectional properties for each of the finite elements in a substructure based on the current state of stress and strain, and (5) computation of the partially restrained connection stiffness at the ends of the beam members in the substructures. All the preceding items can be computed in parallel on multiple processors for each of the substructures in the analytical model. As a result, the time required for the state determination at each load step in the nonlinear solution algorithm can be greatly reduced for large structural analysis models.

#### 4 INELASTIC ANALYSIS ALGORITHM

The preceding discussion of the finite-element formulation implies that the resulting equations of equilibrium for the assembled structure are nonlinear. Furthermore, past research in parallel and vector computing has been focused overwhelmingly on elastic analysis of large structures. In this study, the entire load-deformation response of a large inelastic framework is to be traced. As a result, a nonlinear solution algorithm capable of reliably surpassing limit points is needed. A simple Newton-type incremental algorithm was used to compute the nonlinear load-deformation response. In lieu of iterations at constant load, the constant-work control method<sup>47,61</sup> of load incrementation is used in the solution. Additional details of the nonlinear algorithm can be found elsewhere,<sup>30,33</sup> but it is prudent to highlight some important aspects to give the reader an understanding of the computational procedure.

The incremental work done by an incremental load step  $\{\Delta R_n\}$  in going from displaced (structure) configuration  $\{D_n\}$  to configuration  $\{D_{n+1}\}$  is given by

$$\Delta W_n = \{\Delta D_n\}^T \{\Delta R_n\} \quad (18)$$

An incremental nodal load vector can be written as

$$\{\Delta R_n\} = \Delta \lambda_n \{R_{re}\} \quad (19)$$

where  $\{R_{re}\}$  is a reference nodal load vector (usually the applied nodal loads), and  $\Delta \lambda_n$  is the incremental load factor (a scalar).

The incremental displacements can be determined using the tangent stiffness matrix for the structure in the usual manner:

$$\{\Delta D_n\} = [K_T]^{-1} \{\Delta R_n\} \quad (20)$$

Equation (19) can be substituted into Equation (20), giving

$$\{\Delta D_n\} = \Delta \lambda_n [K_T]^{-1} \{R_{re}\} \quad (21)$$

which can be rewritten as

$$\{\Delta D_n\} = \Delta \lambda_n \{D_{re}\} \quad (22)$$

where the displacements resulting from the reference loads at a point on the load-deformation response is given by

$$\{D_{re}\} = [K_T]^{-1} \{R_{re}\} \quad (23)$$

The incremental work for the  $n$ th load increment can now be written in terms of the incremental nodal load vector using Equations (18) and (22):

$$\Delta W_n = \Delta \lambda_n^2 \{D_{re}\}^T \{R_{re}\} \quad (24)$$

Finally, an incremental load factor for the  $n$ th load increment can be determined using a predetermined incremental work magnitude as well as reference nodal loads and displacements at a given point on the load deformation response. This incremental nodal load factor is given by

$$\Delta \lambda_n = \sqrt{\frac{\Delta W_n}{\{D_{re}\}^T \{R_{re}\}}} \quad (25)$$

Equation (25) forms the basis of the nonlinear solution algorithm used in this study. The simple Euler stepping algorithm with load increments controlled by constant work will help to control the level of drift from the equilibrium response. If the initial load increment is small, the drift also will be small. The accuracy of the proposed algorithm has been demonstrated on both partially and fully restrained frames.<sup>33</sup> Since the full nonlinear load deformation response is traced, a stopping point must be defined by the user in the analysis. In this study, the structure is loaded (and unloaded as required beyond the limit) until a user-defined deformation is reached at a control node within the structure.

Computation of the full nonlinear load deformation response of a large planar structure using distributed plasticity modeling requires a significant amount of computational effort. To combat this and render the advanced analysis of large structures using distributed plasticity tractable, parallel processing and vector computations were incorporated into the computer program. The computer program developed was implemented on a Cray Y-MP C90 multiprocessor supercomputer at the Pittsburgh Supercomputing Center using Fortran. The program uses parallel

**Step 1:** Assemble  $\{R\}_{re}$   $\{r_c\}_{re}$   $\{r_{al}\}_{re}$   
**Step 2:** Assemble  $[K_{T,B}]$   
**Step 3:** Compute  $\{R_B\}_{re}$   
**Step 4:** Solve for  $\{D_B\}_{re}$  using parallel / vector solver - LAPACK  
**Step 5:** Expand  $\{D_B\}_{re}$  and compute  $\{D\}_{re}$   
**Step 6:** Compute the work done (vector);  $W = \{D\}_{re}^T \{R\}_{re}$   
**Step 7:** Check to see if the limit load has been reached or surpassed:  
     **IF** limit has NOT been reached (limit\_flag = 0) **THEN**  
         **IF** work is less than zero **THEN**  
             establish negative incremental load factor:  
             limit\_flag = 1  
         **ENDIF**  
     **ELSE**  
         **IF** work greater than zero, **STOP**.  
     **ENDIF**  
**Step 8:** Compute the incremental work and incremental load factor:  
     **IF** limit\_flag = 0 **THEN**  
         **IF** this is the first load step **THEN**  
              $\Delta W = \Delta \lambda^2 W$   
         **ELSE**  
              $\Delta \lambda = \sqrt{\frac{\Delta W}{W}}$   
         **ENDIF**  
     **ENDIF**  
**Step 9:** Update (vector);  $\{\Delta D\} = \Delta \lambda \{D\}_{re}$      $\{\Delta r_c\} = \Delta \lambda \{r_c\}_{re}$      $\{\Delta r_{al}\} = \Delta \lambda \{r_{al}\}_{re}$   
**Step 10:** Compute  $\{\Delta d_{be}\}$   
**Step 11:** Compute  $\{\Delta d_{al}\}$   
**Step 12:** Update fiber stress and strain  
**Step 13:** Update the section properties  
**Step 14:** Compute current state of connection stiffness.  
**Step 15:** Update (vector - where arrays are involved):  
      $\lambda = \lambda + \Delta \lambda$        $\{d_{al}\} = \{d_{al}\} + \{\Delta d_{al}\}$   
      $\{D\} = \{D\} + \{\Delta D\}$      $\{d_{be}\} = \{d_{be}\} + \{\Delta r_{be}\}$   
**Step 16:** If current reference displacement exceeds limit, **Stop**, otherwise, go to **Step 1**.

Fig. 4. Pseudocode for nonlinear solution algorithm.

processing and vectorization throughout. In general, vectorization is used in as many loops as possible (without significant algorithmic overhaul) through compiler directives. However, loop unrolling is used where applicable.<sup>24,25,40</sup> Parallel processing is implemented at the DO LOOP level using the Cray method of autotasking through microtasking directives. A general statement as to where vectorization and parallel processing are used in the computer program is as follows: Parallel processing is limited to large do loops over all the elements contained in the substructures, whereas any and all inner do loops are vectorized (if possible).

A complete explanation of the total program and locations where vectorization and parallel processing are implemented is not warranted in this article. However, one can get an understanding of the algorithm and the locations where parallel and vector computations occur through

examination of the pseudocode for the nonlinear solution algorithm, found in Figures 4 through 13. Several important aspects to the nonlinear algorithm should be noted at the present time. First of all, step 4 (Figure 4) mentions use of a parallel vector solver from the LAPACK library of linear algebra subroutines. The linear solvers contained in the LAPACK library housed on the Pittsburgh Supercomputing Center Cray machine are fully optimized for parallel and vector computation. Second, if a loop in the pseudocode is executed in parallel, it is denoted by (parallel), and any computations performed in vector mode are indicated as (vector). Loops denoted with (vector-shortloop) were vectorized loops carried out using the Cray *shortloop* compiler directive.<sup>24,25</sup>

Figures 5 through 8 contain algorithm components pertaining to development of the reference nodal load vector (Figure 5), the condensed stiffness matrix (Figure 6),

**Loop 1 - All Subdomains** (parallel)  
**Loop 2 - All Beam Elements in Subdomain**  
 Assemble the beam end displacement vector:  $\{d_{be}\}$   
 Assemble the member stiffness matrix:  $[K_T]_m$   
 Assemble the nodal load vector:  $\{r_{al}\}_{re}$   
 Form  $[K_{T,C}]_m$  and  $\{r_C\}_{m,re}$   
 Modify  $\{r_C\}_{m,re}$  to account for the presence of connections  
**GUARDED REGION**  
 Add  $\{r_C\}_{m,re}$  into  $\{R\}_{eq,re}$   
**END GUARDED REGION**  
**End Loop 2**  
**End Loop 1**  
**Loop 3 - All Degrees of Freedom** (vector)  
 Assemble  $\{R\}_{re} = \{R\}_{eq,re} + \{R\}_{jt,re}$   
**End Loop 3**

**Fig. 5.** Pseudocode for step 1 in the nonlinear algorithm: reference nodal load assembly.

the condensed reference nodal load vector (Figure 7), and computation of the reference displacement vector (expansion phase, Figure 8). Several vectors in Figure 5 should be defined at the present time. Consistent reference nodal forces along the member length condensed to the member ends (nodes  $i$  and  $j$  in Figure 2) are denoted  $\{r_C\}_{m,re}$  in the pseudocode.  $\{R\}_{eq,re}$  and  $\{R\}_{jt,re}$  are equivalent reference

nodal load vectors and reference loads applied directly to the structure joints, respectively. The guarded regions indicate locations where multiple processors are synchronized prior to the operation(s). With respect to formation of the structure condensed tangent stiffness matrix (Figure 6), this ensures that each processor that computes a condensed subdomain stiffness adds this stiffness into the condensed structure stiffness matrix appropriately. Guarded regions frequently are necessary on shared-memory multiprocessor computers to prevent multiple processes from accessing a common memory space.

Figure 9 contains the pseudocode for the reverse condensation process needed to determine the incremental displacements at the ends of the beam members given the current displacements at the beam-to-column joints. Incremental beam end displacements are needed to accurately compute the incremental rotations within the connections. A reverse condensation procedure is also contained in Figure 10. The pseudocode in this figure outlines the procedure used to develop incremental displacements along the member lengths based on the displacements at the ends of the member.

Once all displacements are known, the state of stress within all fibers can be computed using the procedure outlined in Figure 11. It should be emphasized that the rather simplistic finite element used in this study requires that an average curvature be used for any subelement. Part of the

**Loop 1 - All Subdomains** (parallel)  
**Loop 2 - All Elements in Subdomain**  
 Extract member displacements (local coordinates):  
 $\{d\}$  - for column member  
 $\{d_{be}\}$  - for beam member  
**Loop 3 - All Subelements within Element**  
 Form  $[K_T]_{se}$   
 Add sub-element contribution to  $[K_T]_m$  (vector - shortloop)  
**End Loop 3**  
 Form  $[K_{T,C}]_m$   
 Form  $[K_{T,C}]_{bc}$  if beam member  
 Add element contribution to:  $[K_{bb,T}]$ ,  $[K_{ib,T}]$ ,  $[K_{ii,T}]$   
**End Loop 2**  
 Compute:  $[K_{ib,T}]^T [K_{ii,T}]^{-1} [K_{ib,T}]$   
**GUARDED REGION**  
 Add to condensed structure stiffness (vector)  

$$[K_{B,T}] = [K_{B,T}] + \left[ [K_{ii,T}] - [K_{ib,T}]^T [K_{ii,T}]^{-1} [K_{ib,T}] \right]$$
  
**END GUARDED REGION**  
**End Loop 1**

**Fig. 6.** Pseudocode for step 2 in the nonlinear algorithm: formation of condensed tangent stiffness matrix.

Extract  $\{R_B\}_{re}$  from  $\{R\}_{re}$  (vectorization)

**Loop 1 - All Subdomains** (parallel)

**Loop 2 - All Elements in Subdomain**

Extract member displacements (local coordinates):

$\{d\}$  - for column member

$\{d_{be}\}$  - for beam member

**Loop 3 - All Subelements within Element**

Form  $[K_T]_{se}$

Add sub-element to  $[K_T]_m$  (vector - shortloop)

**End Loop 3**

Form  $[K_{T,C}]_m$

Form  $[K_{T,C}]_{bc}$  if beam member

Add element contribution to;  $[K_{bb,T}]$ ,  $[K_{ib,T}]$ ,  $[K_{ii,T}]$

**End Loop 2**

Compute;  $[K_{ib,T}]^T [K_{ii,T}]^{-1}$

Extract  $\{R_i\}_{re}$  from  $\{R\}_{re}$  (vectorization)

**GUARDED REGION**

Add contribution to condensed structure stiffness (vector)

$$\{R_B\}_{re} = \{R_B\}_{re} + \left[ \{R_b\}_{re} - [K_{ib,T}]^T [K_{ii,T}]^{-1} \{R_i\}_{re} \right]$$

**END GUARDED REGION**

**End Loop 1**

Fig. 7. Pseudocode for step 3 in the nonlinear algorithm: formation of condensed reference nodal load vector.

motivation for using such a fine subelement discretization is to ensure that this average curvature is acceptable. Step 12 in the algorithm does nothing more than compute the state of stress in all fibers within any given subelement cross section. It should be noted that if unloading of a fiber is encountered, the initial material modulus of elasticity is used for the increment in stress. This unloading increment is then added to the current state of stress in that fiber. In this fashion, unloading at any yielding locations within the cross section can be considered easily in the analysis.

After the state of stress in each and every fiber of the model is determined, the cross-sectional properties  $A_e$ ,  $I_{ze}$ ,  $S_{ze}$ ,  $d_{CGe}$ , and  $A_p$  are computed. Furthermore, the axial loading resisted by plastified portions of the cross section  $P_{A_p}$  is also determined. These are the tasks outlined in the pseudocode contained in Figure 12. It should be noted that if the cross-sectional area that remains elastic becomes very small (i.e., less than the area of two fibers located in the member web at the original cross-sectional centroid), a minimum area and second moment of area are assigned. These minimums correspond to the two fiber areas just mentioned.

The final state determination needed in the nonlinear analysis is determination of the current connection rotation and stiffness. The pseudocode for this stage in the algorithm is contained in Figure 13. The incremental rotations

**Loop 1 - All Subdomains** (parallel)

**Loop 2 - All Elements in Subdomain**

Extract member displacements (local coordinates):

$\{d\}$  - for column member

$\{d_{be}\}$  - for beam member

**Loop 3 - All Subelements within Element**

Form  $[K_T]_{se}$

Add sub-element to  $[K_T]_m$  (vector - shortloop)

**End Loop 3**

Form  $[K_{T,C}]_m$

Form  $[K_{T,C}]_{bc}$  if beam member

Add element contribution to;  $[K_{ib,T}]$ ,  $[K_{ii,T}]$

**End Loop 2**

Extract  $\{D_b\}_{re}$  from  $\{D_B\}_{re}$  (vector)

Extract  $\{R_i\}_{re}$  from  $\{R\}_{re}$  (vector)

Compute subdomain internal reference displacements;

$$\{D_i\}_{re} = [K_{ii,T}]^{-1} \left[ \{R_i\}_{re} - [K_{ib,T}] \{D_b\}_{re} \right]$$

**GUARDED REGION**

Place  $\{D_i\}_{re}$  into  $\{D\}_{re}$

**END GUARDED REGION**

**End Loop 1**

Add  $\{D_B\}_{re}$  to  $\{D\}_{re}$  (vectorization)

Fig. 8. Pseudocode for step 5 in the nonlinear algorithm: computation of structure reference displacement vector (substructure displacement expansion phase).

**Loop 1 - All Subdomains** (parallel)

**Loop 2 - All Beam Elements in Subdomain**

Extract incremental displacements at beam joints:  $\{\Delta d_i\}$

**Loop 3 - All Subelements within Element**

Form subelement stiffness:  $[K_T]_{se}$  using current  $\{d_{ai}\}$

Assemble  $[K_T]_{se}$  into  $[K_T]_m$  (vector - shortloop)

**End Loop 3**

Form  $[K_{T,C}]_{bc}$

Compute  $\{\Delta d_{be}\}$  using  $[K_{T,C}]_{bc}$  and  $\{\Delta r_C\}$

**End Loop 2**

**End Loop 1**

Fig. 9. Pseudocode for step 10 in the nonlinear algorithm: computation of incremental displacements at the beam ends.

in the connections are first determined. This is the difference between the incremental rotation at the beam end and the incremental rotation at the beam-to-column joint. This incremental rotation is then multiplied by the current connection rotation to determine if unloading occurs. If unloading of the connection does occur, the stiffness is changed

```

Loop 1 - All Subdomains (parallel)
  Loop 2 - All Elements in Subdomain
    Extract current displacements at member ends;
    { $d_i$ } - for column member
    { $d_{be}$ } - for beam member
    Loop 3 - All Subelements within Element
      Extract current subelement displacements: { $d_{ai}$ }
      Assemble portions of  $[K_T]_m$  needed for reverse condensation
    End Loop 3
    Establish incremental displacements at the member ends;
    { $\Delta d_i$ } - for column member
    { $\Delta d_{be}$ } - for beam member
    Compute { $\Delta d_{ai}$ } using { $\Delta r_{ai}$ } and portions of  $[K_T]_m$ 
  End Loop 2
End Loop 1

```

Fig. 10. Pseudocode for step 11 in the nonlinear solution algorithm: computation of the incremental displacements along members.

```

Loop 1 - All Subdomains (parallel)
  Loop 2 - All Elements in Subdomain
    Extract incremental displacements at member ends
    { $\Delta d_i$ } - for column member
    { $\Delta d_{be}$ } - for beam member
    Loop 3 - All Subelements within Element
      Extract current incremental displacements: { $d_{ai}$ }
      Compute the incremental axial strain and incremental
      average curvature for sub-element (remove rigid
      body subelement displacements).
    Loop 4 - Over Cross-Section Fibers (vector)
      Update fiber stress and strain
      IF  $\sigma_{curr} \geq \sigma_y$  THEN  $\sigma_{curr} = \sigma_y$ 
    End Loop 4
  End Loop 3
End Loop 2
End Loop 1

```

Fig. 11. Pseudocode for step 12 in the nonlinear algorithm: computation of current fiber stress state.

to the first linear stiffness in the trilinear model, and the current connection rotation is reset to zero. If the connection loads, the incremental connection stiffness is set to the appropriate magnitude in the trilinear model. Further details regarding the connection models can be found elsewhere.<sup>30</sup>

## 5 PROGRAM PERFORMANCE MEASURES

Program performance can be measured in a variety of ways. One intuitive measure is wall-clock time. However, this tends to be a misleading measure of program performance on nondedicated multiprocessor machines such as

```

Loop 1 - All Subdomains (parallel)
  Loop 2 - All Elements in Subdomain
    Loop 3 - All Subelements within Element
      Loop 4 - Over Cross-Section Fibers (vector - shortloop)
        Compute elastic core properties
      End Loop 4
      Set minimum elastic core properties (as required)
      Loop 5 - Over Cross-Section Fibers (vector - shortloop)
        Compute plastic axial loading
      End Loop 5
    End Loop 3
  End Loop 2
End Loop 1

```

Fig. 12. Pseudocode for step 13 in the nonlinear algorithm: update of cross-sectional properties corresponding to current fiber stress state.

```

Loop 1 - All Subdomains (parallel)
  Loop 2 - All Beam Elements in Subdomain (vector)
    Compute incremental connection rotations
    Compute products;  $\Delta \theta_L * \theta_{L,curr}$  and  $\Delta \theta_R * \theta_{R,curr}$ 
    IF a product is less than ZERO THEN
      Connection unloads, therefore, set  $\theta_{curr} = 0$ 
    ELSE
      Connection loads, therefore, add increment
    ENDIF
  End Loop 2
End Loop 1

```

Fig. 13. Pseudocode for step 14 in the nonlinear algorithm: update of current beam connection stiffness based on current connection rotation.

the Cray Y-MP C90 used in this study. The computer program written as part of this research was required to be run in batch mode with many competing jobs. Therefore, processors often were swapped between jobs by the operating system, even though the program explicitly requested a user-defined number of processors at compile time. As a result, the processing can occur piecemeal such that the wall-clock time domain may be measuring noncontiguous process time. Therefore, a much better set of measures for assessing program performance are floating-point operations and processor (analogous to CPU) connect time.

The Cray Y-MP C90 multiprocessor computer used in this research allows scalar, vector, and parallel processing to be defined using compiler switches. The program was run in three distinct modes: (1) scalar mode, (2) vector mode, and (3) autotasked (parallel-vector) mode. The scalar mode was used to simulate a typical von Neuman computer (e.g., a typical desktop PC). It is recognized that the CPU clock speeds on a Y-MP C90 are not typical of those found in PCs, but the scalar mode will give qualitative indication of typical computer performance. The vector mode included only vector processing (i.e., only one processor was requested). Therefore, in this mode, the Cray autotasking directives related to parallel processing were ignored because only one process was requested. This mode would allow the speedup directly attributable to vector computation to be quantified. The final mode was autotasked mode. In these runs, the Cray compiler was instructed to consider both the vectorization and parallelization compiler directives. Furthermore, a number of processors equal to the number of subdomains present in the model was requested. This resulted in full realization of the speedup associated with both vector and parallel processing.

The parallel-vector mode performance data will help to quantify the speedup attributable to purely parallel processing. This speedup can then be used to gauge program performance within the context of expected performance of a similar program in a distributed (parallel) workstation environment. It is understood that this may be an overly simplistic analogy because the efficiency of message passing and memory communication in distributed workstation environments is significantly different from that on the shared-memory MIMD Cray C90. However, it is felt that analogous performance between the two environments will be seen qualitatively.

An added consideration in measuring program performance is the level at which the data are reported. The first may be considered reporting overall program performance, and the other can be considered as reporting task performance. This study used runs as part of a production research effort. As such, the computer program was writing significant amounts of output data at each step in the incremental nonlinear algorithm (e.g., nodal displacements, fiber

strains, fiber stresses, effective cross-sectional areas, connection rotations, and load factors). This resulted in overall program performance being unduly impeded by extreme output frequency. Therefore, this study reports program performance data at the task level. The tasks in the nonlinear algorithm were defined as the steps in Figure 4. If it is assumed that incremental load steps within a nonlinear solution algorithm cannot be carried out in parallel due to recursion (very realistic, since each load step depends on data determined in the previous load step), one can make the argument that if each task within the nonlinear algorithm is sped up, the entire nonlinear algorithm will speed up. Therefore, this is the rationale behind reporting program performance at the task level.

It should be noted that a large amount of research has been devoted to improved performance of equation solvers through parallel processing and vectorization over the last two decades. This study used the LAPACK library of equation solvers resident on the Cray C90. These solvers are extremely fast and efficient and are optimized for the C90 machine. It was felt that trying to create or implement an alternate set of parallel-vector factorization and backsubstitution routines would not be prudent. Therefore, any discussion related to the performance of solving routines (tasks) is ignored.

The two measures of task performance used in this study are speedup and efficiency. As eluded to in previous discussion, this article will focus on performance increases due to parallel and vector processing. The first performance measure to be considered is *speedup*. Within the context of this study, speedup is quantified three ways, reflecting the type of processing used in the computer run. The first is speedup resulting from vector processing alone (i.e., moving from scalar to vector processing):

$$SU_{S \rightarrow V} = \frac{PCT_{\text{scalar}}}{PCT_{\text{vector}}} \quad (26)$$

where  $PCT_{\text{vector}}$  is the process connect time for vector computations, and  $PCT_{\text{scalar}}$  is the process connect time for scalar computations. Therefore, the speedup given in Equation (26) attempts to quantify the improvement resulting from vector processing over that anticipated using a machine with a scalar processor. A second speedup measures the improvement going from vector processing to parallel-vector processing. This measure is given by

$$SU_{V \rightarrow P} = \frac{PCT_{\text{vector}}}{APCT_{\text{autotask}}} \quad (27)$$

where  $APCT_{\text{autotask}}$  is the average processor connect time for the parallel (autotasked) run. The speedup expressed in Equation (27) attempts to quantify the speedup due to parallel processing alone. The final speedup measures the performance enhancement moving from scalar processing

to vector/parallel processing. This speedup is defined as

$$SU_{S \rightarrow P} = \frac{PCT_{\text{scalar}}}{APCT_{\text{autotask}}} = SU_{S \rightarrow V} SU_{V \rightarrow P} \quad (28)$$

The multiplicative effect illustrated in Equation (28) implies that superlinear speedups (when compared with a scalar processing run) can be attained if vector and parallel processing are used. This results from each of the 16 processors on the Cray C90 being a vector processor. Therefore, the machine is capable of creating parallel-vector processes, and superlinear speedups over scalar execution should be expected.

The speedup expressions contained in Equations (26) through (28) mask the difficulty encountered when attempting to achieve significant speedup using vector and parallel processing. A theoretical upper limit on speedup can be determined using Amdahl's law. On the Cray C90, the theoretical speedup via vectorization alone may be written as

$$SU_V = \frac{1}{f_S + (f_V/R_V)} \quad (29)$$

where  $f_V$  is the fraction of code executed with vector processing,  $f_S$  is the fraction of computer code executed in scalar mode ( $1 - f_V$ ), and  $R_V$  is a machine-dependent value denoting the ratio of scalar processing time to vector processing time.  $R_V$  ranges from 10 to 20 on Cray machines.<sup>24,25</sup> Therefore, assuming that 50 percent of the code is executed in vector mode and assigning  $R_V = 20$ , the speedup expected through vectorization (over scalar execution) is only 1.9. If 75 percent of the code is executed in vector mode, the expected speedup is 3.48. Therefore, significant speedups (2 or greater) are not expected until over 50 percent of the computer code is executed in vector mode. Achieving a 50 percent vector code many times requires a significant overhaul of the computational algorithm, and therefore, significant speedup due to vector processing can be limited in finite-element computations.

Amdahl's law is also applicable to parallel processing and is expressed mathematically in a similar manner to that for vector processing:

$$SU_P = \frac{1}{f_S + (f_P/N)} \quad (30)$$

where  $f_P$  is the fraction of code executed in parallel. Examination of Equation (30) allows qualifying statements to be made with respect to the expected speedup due to parallel processing. First of all, linear speedup will not be attained until 100 percent of the code is executed in parallel. Therefore, one could expect code segments (i.e., tasks) to perform with near-linear speedup. However, to expect the entire code to be executed in parallel is a tall order and highly unlikely to be attained in practice. Second, as the number of processors increases, so does the percentage of

code needed to be executed in parallel in order to approach linear speedup.

The discussion related to Amdahl's law leads to the following conclusions regarding expected performance of the computer program using advanced computing technology (e.g., vector processing and parallel processing). First of all, significant speedup from vectorization should not be expected. The computer program used in this study was not overhauled to allow maximum vector computation. In essence, the Cray C90 Fortran compiler was used to determine loops that were not vectorized automatically by the compiler. Once these loops were flagged, minimal recoding (e.g., loop unrolling<sup>40</sup>) was performed to allow vector processing to occur. If it was deemed that significant recoding was required, none was done. With regard to parallel processing, the computer program was developed with emphasis on achieving maximum parallel execution within each subroutine (task). Therefore, it is expected that speedups due to parallel processing will be more significant. Also, since parallel processing is localized to subroutines (tasks), the performance enhancement should be significant.

## 6 STEEL FRAMES ANALYZED

Three structural steel frames shown in Figure 5 were used to evaluate code performance. The computer program allows fully restrained (FR) and partially restrained (PR) connections. Performance results for the FR frames alone are provided. Performance for both FR and PR connected frames is expected to be the same because evaluation of the current state of connection stiffness (refer to Figure 14) is done for both frame types.

The service gravity and lateral loads for all three frames are based on the following general building layout and configuration. Each frame is part of a lateral load-resisting system in which each lateral load-resisting moment frame is spaced at 9.1 m on center. The dimension of the building orthogonal to the wind direction is 54.9 m for all buildings. All frame loadings are based on a 7.6-m bay spacing (both directions) and a common story height of 3.8 m. Wind loading is based on a basic wind speed of 128.7 km/h, an importance factor of 1.0, and exposure category A.<sup>18</sup> Typical office loading was used to assign gravity loads to the frames. Roof live loading was assumed to be 1.05 kN/m<sup>2</sup> (snow). Floor live loading was assumed to be 3.11 kN/m<sup>2</sup>. The dead loading was assumed to be 3.59 and 3.11 kN/m<sup>2</sup> for the roof and floor, respectively. All members within the structure were assumed to have a yield stress of 248 MPa, and residual stresses vary in linear fashion over the flange width (from tension to compression) and are constant (tension) within the web.<sup>39</sup>

The large-building frame structures studied in this research effort required that live load reduction procedures

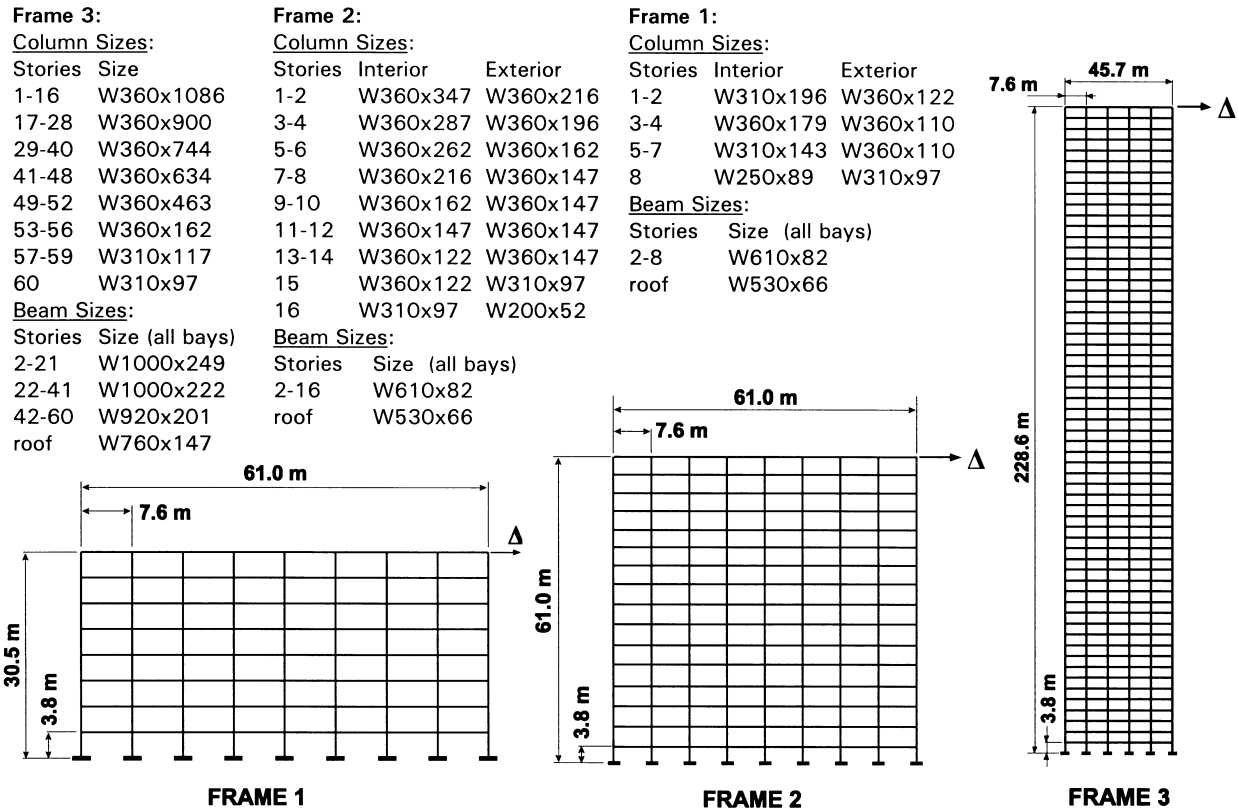


Fig. 14. Frames used in present study to measure program performance.

be used in preliminarily sizing the members for analysis. However, the usual live load reduction methods incorporate a column takedown analysis. This is not conducive to implementation as part of a matrix structural analysis technique. Therefore, reduction of live loading is simulated via upward-compensating forces<sup>62,63</sup> applied at the beam-to-column joints. These upward loads are treated as live loading in the matrix structural analysis.

The three multistory, multibay frames are analyzed for ultimate load and postcritical response using the fiber-element (plastic zone) analytical procedure. Each beam member has 30 subelements, and each column member has 20 subelements. The subelements are of uniform length along the member. The number of nodes, elements, fibers, and degrees of freedom in each framework is given in Table 1. The number of nodes and elements contains all nodes and elements between the structure nodes (i.e., the beam to column joints) used to model along the member yielding. The number of degrees of freedom in Table 1 accounts for restrained displacements at support conditions in the framework. Run times for a Pentium Pro 200-MHz PC with a single processor are provided. The number of incremental load steps used by the nonlinear algorithm are also contained in the table. Lastly, the (average) time required for completion of each load increment is given.

As expected, the computational effort (on a per-increment basis) is the same for all connections assumed. Furthermore, the computational effort increases as the size of the frame increases, and this increase is nonlinear. Lastly,

Table 1

Model size parameters and PC timing information for the frames used in this study

Parameter*	Frame designations		
	1	2	3
Nodes	3305	6601	15906
Elements	3360	6720	16200
Fibers/element	66	66	66
Total fibers in model	221760	443520	1069200
Degrees of freedom	9888	19776	47700
FR Run time	1:35:33	2:24:33	3:08:07
No. of increments	2300	1742	937
Time/increment	2.49 s	4.98 s	12.05 s
EFP Run time	2:56:09	5:27:34	3:55:55
No. of increments	4198	3928	1150
Time/increment	2.52 s	5.00 s	12.31 s
FP Run time	3:07:17	1:18:30	3:03:20
No. of increments	4527	945	889
Time/increment	2.48 s	4.98 s	12.37 s

\*FR, fully restrained; EEP, extended end plate; FP, flange plate.

the solution of these problems on a scalar computer is computationally intensive and perhaps impractical for design office use. However, one could argue that a 200-MHz PC is less than state of the art.

### 7 RESULTS AND DISCUSSION

Speedups for the various subroutines (tasks) are provided in Tables 2 through 4. The definition of speedup in these cases has been presented in Equations (26) and (27). The speedup due to vectorization alone was not appreciable for some tasks, whereas for others the vectorization speedup was significant. Significant speedups were attained in steps 12, 13, and 14. Speedups of 5 and greater indicate that a large number of computations were performed in vector mode during execution of the task. An example of the effect of overhead to start a vector process on speedup attained is illustrated through the results of step 14, whose pseudocode is contained in Figure 13. Loop 2, which takes advantage of vector processing, has its limits defined by the number of elements contained in each subdomain. It is clearly seen in Table 2 that the speedup achieved through vector processing drops significantly as the number of subdomains increases. The reason for this is that as the number of subdomains increases, the vectorized loop becomes shorter

**Table 2**

Speedup for various steps in the nonlinear solution algorithm as a function of the number of substructures for frame 1

Subroutine	Compilation and run-time mode	Number of substructures (processors)				
		2	3	4	6	8
Step 1	$S \rightarrow V$	1.21	1.20	1.20	1.20	1.20
	$V \rightarrow P$	1.95	2.95	3.86	5.81	7.72
Step 2	$S \rightarrow V$	1.31	1.42	1.55	1.82	2.05
	$V \rightarrow P$	1.98	2.94	3.83	5.35	6.64
Step 3	$S \rightarrow V$	1.17	1.20	1.24	1.32	1.40
	$V \rightarrow P$	1.99	3.00	4.00	6.01	7.99
Step 5	$S \rightarrow V$	1.11	1.12	1.15	1.19	1.23
	$V \rightarrow P$	1.96	2.93	3.95	5.93	7.89
Step 10	$S \rightarrow V$	1.09	1.08	1.08	1.08	0.08
	$V \rightarrow P$	1.99	3.00	4.02	6.01	8.04
Step 11	$S \rightarrow V$	1.11	1.10	1.08	1.10	1.10
	$V \rightarrow P$	1.94	2.93	3.91	5.87	7.79
Step 12	$S \rightarrow V$	10.21	10.20	10.05	10.29	10.19
	$V \rightarrow P$	1.64	2.49	3.34	4.93	6.54
Step 13	$S \rightarrow V$	6.05	5.94	5.90	6.00	6.01
	$V \rightarrow P$	0.331	0.501	0.67	1.00	1.33
Step 14	$S \rightarrow V$	3.23	2.61	2.34	1.80	1.47
	$V \rightarrow P$	1.70	2.71	3.34	4.96	6.41

**Table 3**

Speedup for various steps in the nonlinear solution algorithm as a function of the number of substructures for frame 2

Subroutine	Compilation and run-time mode	Number of substructures (processors)				
		2	4	6	8	10
Step 1	$S \rightarrow V$	1.20	1.20	1.19	1.19	1.19
	$V \rightarrow P$	1.95	3.84	5.77	7.67	9.64
Step 2	$S \rightarrow V$	1.21	1.46	1.59	1.84	2.09
	$V \rightarrow P$	1.98	3.70	5.45	6.53	7.20
Step 3	$S \rightarrow V$	1.12	1.17	1.20	1.24	1.28
	$V \rightarrow P$	2.00	3.98	6.00	7.98	9.96
Step 5	$S \rightarrow V$	1.09	1.11	1.12	1.15	1.17
	$V \rightarrow P$	1.97	3.94	5.92	7.87	9.81
Step 10	$S \rightarrow V$	1.08	1.08	1.08	1.08	1.08
	$V \rightarrow P$	2.00	4.00	6.02	8.02	10.03
Step 11	$S \rightarrow V$	1.10	1.10	1.09	1.08	1.09
	$V \rightarrow P$	1.95	3.92	5.90	7.85	9.83
Step 12	$S \rightarrow V$	10.26	10.12	10.07	10.09	10.11
	$V \rightarrow P$	1.68	3.38	5.04	6.64	8.38
Step 13	$S \rightarrow V$	6.05	6.01	5.95	5.85	5.85
	$V \rightarrow P$	0.33	0.68	1.02	1.36	1.70
Step 14	$S \rightarrow V$	3.85	3.23	2.74	2.34	2.06
	$V \rightarrow P$	1.87	3.65	5.20	6.53	8.19

**Table 4**

Speedup for various steps in the nonlinear solution algorithm as a function of the number of substructures for frame 3

Subroutine	Compilation and run-time mode	Number of substructures (processors)				
		2	4	6	8	10
Step 1	$S \rightarrow V$	1.19	1.89	1.19	1.19	1.19
	$V \rightarrow P$	1.97	3.99	5.90	8.02	9.78
Step 2	$S \rightarrow V$	1.19	1.36	1.62	1.90	2.18
	$V \rightarrow P$	1.96	3.85	5.43	6.56	7.54
Step 3	$S \rightarrow V$	1.10	1.13	1.16	1.19	1.21
	$V \rightarrow P$	1.97	3.99	5.94	7.92	9.90
Step 5	$S \rightarrow V$	1.13	1.15	1.16	1.18	1.19
	$V \rightarrow P$	1.98	3.96	5.94	7.94	9.88
Step 10	$S \rightarrow V$	1.05	1.05	1.05	1.06	1.06
	$V \rightarrow P$	2.00	4.02	5.96	7.98	9.96
Step 11	$S \rightarrow V$	1.10	1.10	1.10	1.10	1.10
	$V \rightarrow P$	1.94	3.89	5.85	7.81	9.64
Step 12	$S \rightarrow V$	9.78	9.82	9.84	9.87	9.80
	$V \rightarrow P$	1.65	3.29	4.96	6.60	8.21
Step 13	$S \rightarrow V$	6.24	6.25	6.30	6.24	6.17
	$V \rightarrow P$	0.33	0.66	0.98	1.36	1.72
Step 14	$S \rightarrow V$	4.98	4.56	3.70	3.31	3.09
	$V \rightarrow P$	1.84	3.67	5.343	7.20	8.81

and shorter in length. Therefore, the process time to start the vector loop becomes significant, and the time the vector process takes once it is started decreases significantly. Therefore, one should keep in mind that the vector being processed must have sufficient length to ensure that the overhead to start the computation is outweighed by the time it takes to finally execute the process. One can see that the loss in speedup is less drastic for frames 2 and 3 because the vector loops are larger as a result of more elements within each subdomain.

In all cases (with one exception), near-linear speedup was attained via parallel processing implementation ( $V \rightarrow P$  row in Tables 2 through 4). If one were to include the speedup attained through parallel-vector computations over scalar computations, superlinear results are achieved. Step 12 for frame 2 (Table 3) indicates that the speedup in going from scalar to vector-parallel computation was  $10.1 \times 8.4$ , or 84.8. The exception with regard to performance increase is step 13, with the pseudocode given in Figure 12. While the vector computation achieved significant speedup over the scalar computations, the parallel computation did not perform well. Two inner (vectorized) loops run over the 66-element fibers (array indices). While the vector processing within this step performed quite well, it appears that the combination of the *Shortloop* compiler directive used in this study and parallel processing seems to have caused unpredictable results. Unfortunately, the poor parallel processing performance in this step remains to be explained definitively.

The second measure of program performance used in this study is efficiency. The efficiency of a parallel task can be defined as

$$\eta_P = \left( \frac{SU_P}{N} \right) 100 \quad (31)$$

where  $SU_P$  is the speedup attained via parallel processing, and  $N$  is the number of processors used (equal to the number of structure subdomains). The efficiency of each subroutine can be inferred from the speedups contained in Tables 2 through 4. Figure 15 illustrates the efficiency of step 1 for the three frames studied with a variety of subdomains. As can be seen in the figure, greater than 95 percent efficiency is attained throughout. Virtually identical efficiency plots to that shown in Figure 15 were obtained for steps 3, 5, 10, and 11.<sup>30</sup>

One notable exception to the favorable efficiency is step 2, which forms the condensed tangent stiffness matrix and condensed incremental nodal load vector mathematically stated in Equations (15) and (16). Figure 16 indicates that as the number of subdomains increases (regardless of frame size), the number of boundary degrees of freedom increases. There is a point of diminishing returns whereby the number of boundary degrees of freedom becomes significant and subroutine performance is significantly impeded.<sup>30,31</sup> From Figures 17 and 19 it can be seen

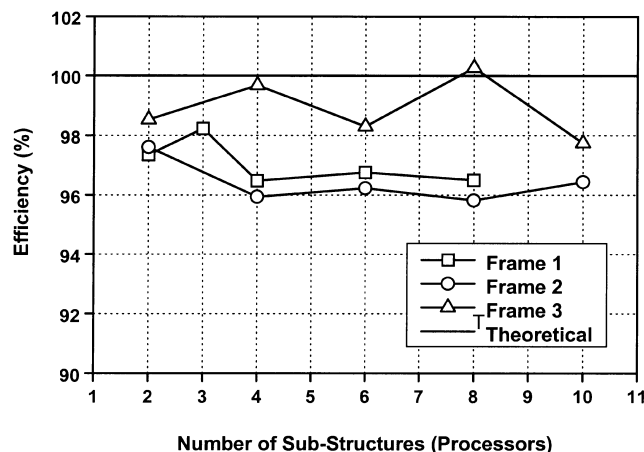


Fig. 15. Performance for parallel versus vectorized code for step 1 in the nonlinear solution algorithm.

that subroutine steps 12 and 13 had much less efficiency as a result of the size of the parallel tasks within the routines. That is, the lengths of the tasks executed in parallel were not significant, and therefore, the speedup due to parallelization was not as great. Step 13 suffered from very poor efficiency, as indicated in Figure 18. This should be expected after the discussion of the poor speedups attained in this routine.

A final measure of program performance often used is the number (millions) of floating-point operations executed per second (MFLOPS). Unfortunately, these data were not recorded in this study. As a result of significant output generation, the program was not performing floating-point operations for significant time durations. MFLOPS data at

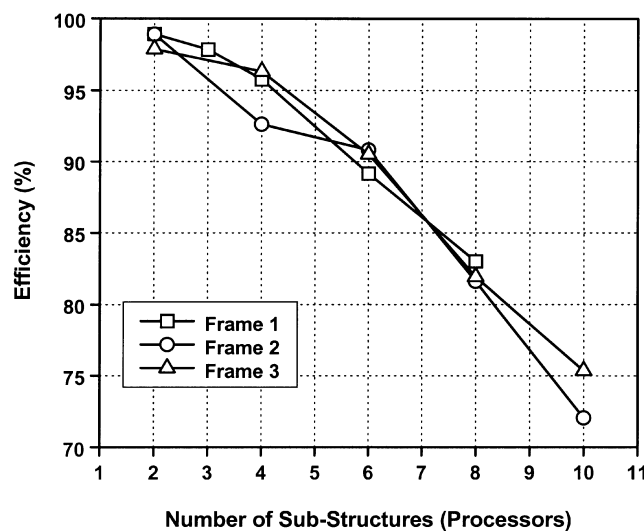


Fig. 16. Performance for parallel versus vectorized code for step 2 in the nonlinear solution algorithm.

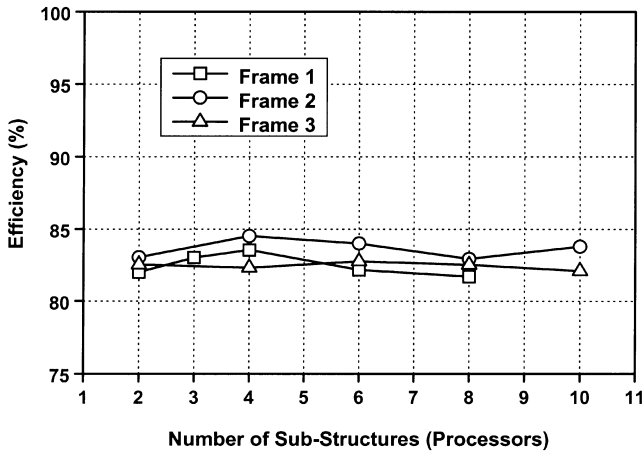


Fig. 17. Performance for parallel versus vectorized code for step 12 in the nonlinear solution algorithm.

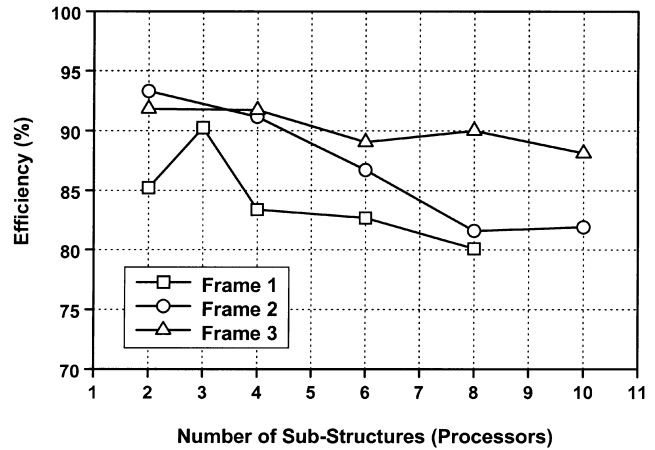


Fig. 19. Performance for parallel versus vectorized code for step 14 in the nonlinear solution algorithm.

the subroutine level could not be obtained during any of the runs. However, one can qualitatively make some judgments related to expected performance from previous research. MFLOP performance was measured for a similar program without the same magnitude of output generation (i.e., it was a timing study, not an inelastic frame analysis study).<sup>31</sup> In the former study, 40 to 100 MFLOPS were attained for frames similar to the frames in this study. It is surmised that performance of at least this magnitude could be expected from the program used in this study.

Qualitative statements can be made with respect to program performance on a PC using the timing data contained in Table 1 and the speedup data ( $V \rightarrow P$ ) contained in Tables 2 through 4. If one were to isolate frame 3 timing data in Table 1, it can be seen that each iteration (on average) took 12 s to execute. From Table 4 it can be seen that the major time-consuming steps (steps 1 through 12

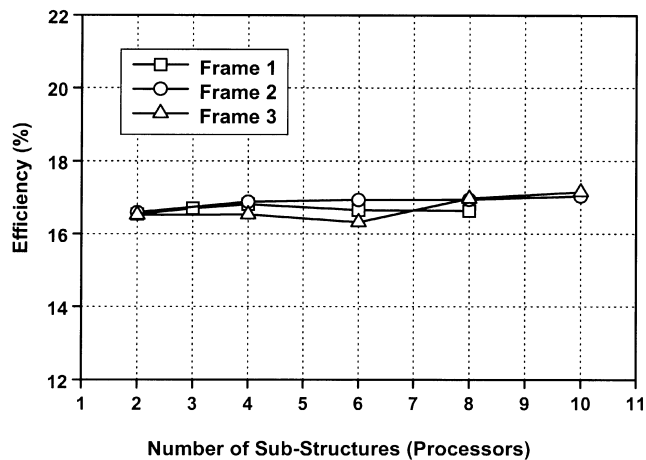


Fig. 18. Performance for parallel versus vectorized code for step 13 in the nonlinear solution algorithm.

and step 14) in the nonlinear algorithm had speedups due to parallelization ranging from 7.5 to 9.9. Step 13 is an obvious anomaly in the results. Therefore, one could say qualitatively that parallel implementation of this program on a multiprocessor PC could achieve a speedup of around 8 (reduced from 9.9 to account for step 13 performance). Thus 12 s per iteration could be reduced to 1.5 s. The analysis time for frame 3 then could be reduced to approximately 23 minutes for frame 3 with FR connections.

The entire goal of the vector and parallel computations described in this article was to compute the ultimate load and nonlinear load-deformation response for practically sized building frameworks. Figure 20 illustrates the load-deformation response for the three building structures shown in Figure 14 for three connection variations (FR, fully restrained; PR, partially restrained, with extended endplate connections; and PR with flange plate connections). The extended endplate and flange-plate connections were designed for the beam end moments obtained assuming a fully restrained frame design analysis. PR connection models were then developed using semianalytical expressions visually fitted with multilinear models. The connection design and trilinear modeling parameters can be found elsewhere.<sup>30</sup> It should be noted that both the ultimate load and postcritical response are captured in the results. Furthermore, uniformly distributed member loads are assumed to be present on the girders, and horizontal wind loading is assumed to be concentrated at the floor levels. A user-defined control node (the leeward roof node on the analytical model) is used to assign a maximum horizontal displacement. This limit is then used to stop the nonlinear algorithm after the limit load has been reached. From the information contained in Table 1, one can surmise that these analyses were computationally intensive. The constant-work method of analysis required that very

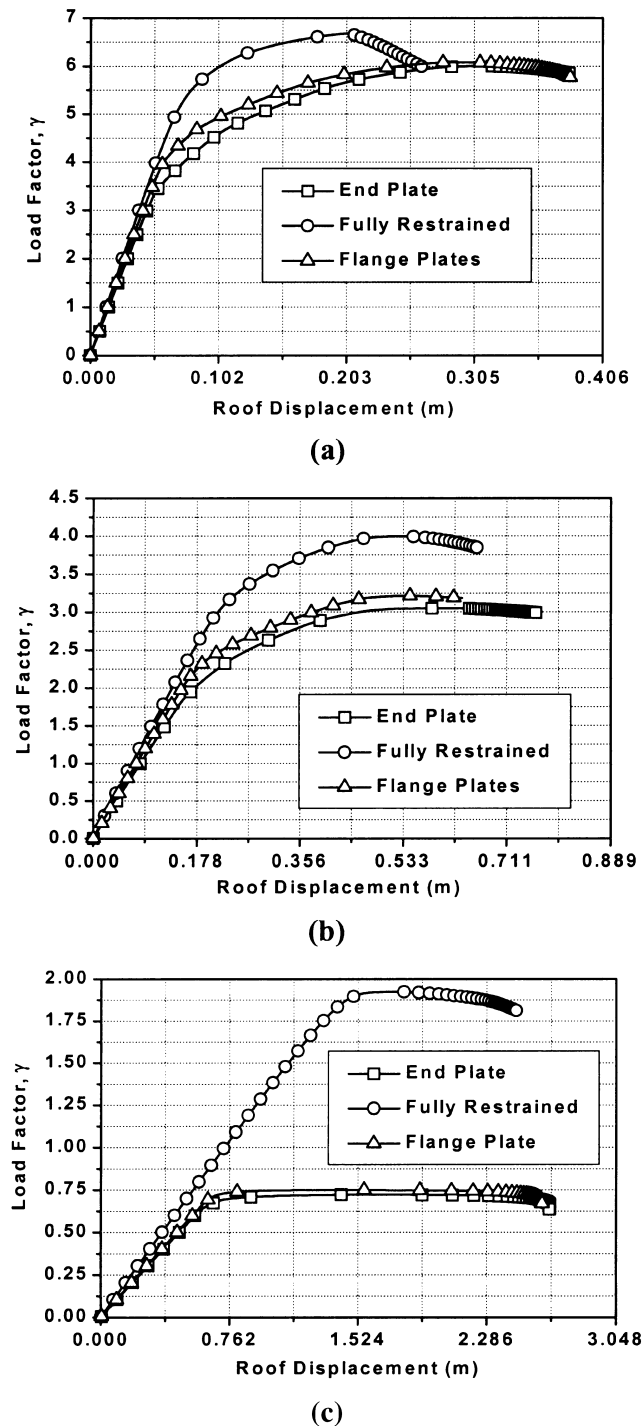


Fig. 20. Frame load-deformation response: (a) frame 1; (b) frame 2; (c) frame 3.

small increments in load be taken in regions of severe change in structural response (degree of expected nonlinearity). (The symbols shown in Figure 20 are for annotation only. They do not reflect the number of solution points

needed during the solution.) The literature does not contain analytical results for frames of this size using distributed plasticity analysis. Figure 20 indicates that distributed plasticity can be used to analyze practical frameworks for nonlinear load-deformation response not only to compute ultimate loads but also to determine the postcritical branches.

## 8 CONCLUSIONS

The development of a finite element capable of modeling spread of plasticity within the cross section of structural steel members has been described. This element is incorporated into a computer program for advanced (plastic zone) analysis of large scale FR and PR frameworks using parallel and vector computations. The method of substructuring incorporated into a nonlinear solution algorithm was outlined, and its implementation in nonlinear frame analysis was described. Measures relevant to program performance on a Cray Y-MP C90 supercomputer were discussed. Three large-scale multistory, multibay steel frames were analyzed using the proposed algorithm for collapse load and postcritical response. The results presented illustrate the significant computational speedup possible using parallel and vector computations. Although the results pertaining to vector computations are not readily extendable to desktop workstations and PCs, it is believed that the speedups attained via parallel computations will be seen on parallel PCs of the future. These computational improvements indicate that the plastic zone method of advanced analysis can become an everyday engineering tool in the design office of the future. Lastly, this study hopefully will give other researchers confidence that spatial (three-dimensional) plastic zone analysis<sup>54</sup> of large structures can be handled with confidence.

## ACKNOWLEDGMENTS

Portions of this research were carried out using supercomputing resources from the Pittsburgh Supercomputing Center and the National Science Foundation under Grant Numbers MSM920008P and MSM940008P. I also would like to acknowledge and thank Professor Ramulu S. Vinnakota for many hours of fruitful discussion related to plastic zone analysis and structural stability theory.

## REFERENCES

1. Ackroyd, M. & Gerstle, K. H., Behavior of type 2 steel frames, *Journal of the Structural Division*, **108** (ST7) (1982), 1541-55.
2. Adeli, H., *Supercomputing in Engineering Analysis*, Marcel Dekker, New York, 1992.

3. Adeli, H. & Cheng, N.-T., Concurrent genetic algorithms for optimization of large structures, *Journal of Aerospace Engineering*, **7** (3) (1994), 276–96.
4. Adeli, H. & Kamal, O., Concurrent analysis of large structures: I. Algorithms, *Computers & Structures*, **42** (1992), 413–24.
5. Adeli, H. & Kamal, O., Concurrent analysis of large structures: I. Applications, *Computers & Structures*, **42** (1992), 425–32.
6. Adeli, H., Kamat, M. P., Kulkarni, G. & Vanluchene, R. D. High-performance computing in structural mechanics and engineering, *Journal of Aerospace Engineering*, **6**(3) (1993), 249–67.
7. Adeli, H. & Kumar, S., Concurrent structural optimization on massively parallel supercomputer, *Journal of Structural Engineering*, **121** (11) (1995), 1588–97.
8. Adeli, H. & Kumar, S., Distributed finite-element analysis on network of workstations: Algorithms, *Journal of Structural Engineering*, **121** (10) (1995), 1448–55.
9. Adeli, H. & Kumar, S., Distributed finite-element analysis on network of workstations: Implementation and application, *Journal of Structural Engineering*, **121** (10) (1995), 1456–62.
10. Adeli, H. & Kumar, S., *Distributed Computer-Aided Engineering*, CRC Press, Boca Raton, FL, 1999.
11. Adeli, H. & Park, H. S., *Neurocomputing for Design Automation*, CRC Press, Boca Raton, FL, 1998.
12. Adeli, H. & Saleh, A. Integrated structural/control optimization of large adaptive/smart structures, *International Journal of Solids and Structures*, **35** (28-29) (1998), 3815–30.
13. Adeli, H. & Saleh, A., *Control, Optimization and Smart Structures: High-Performance Bridges and Buildings of the Future*, John Wiley & Sons, New York, 1999.
14. Adeli, H. & Soegiarso, R., *High-Performance Computing in Structural Engineering*, CRC Press, Boca Raton, FL, 1999.
15. Adeli, H. & Vishnubhotla, P., Parallel machines, in *Parallel Processing in Computational Mechanics*, ed. by H. Adeli, Marcel Dekker, New York, 1992, pp. 1–20.
16. AISC, *Load and Resistance Design Specifications for Structural Steel Buildings*, American Institute of Steel Construction, Chicago, 1994.
17. Al-Bermani, F. G. A. & Kitipornchai, S., Elasto-plastic large deformation analysis of thin-walled structures, *Engineering Structures*, **12** (January 1989), 28–36.
18. ASCE, *ASCE 7-88: Minimum Design Loads for Buildings and Other Structures*, American Society of Civil Engineers, New York, 1990.
19. Atalla, M. R., Deierlein, G. G. & McGuire, W., Spread of plasticity: A quasi-plastic-hinge approach, *Journal of Structural Engineering*, **120** (8) (1995), 2451–73.
20. Bridge, R. Q., Clarke, M. J., Osterrieder, P., Pi, Y.-L. & Trahair, N. S., Design by advanced analysis, *Journal of Constructional Steel Research*, **46** (1–3) (1998), 144.
21. Chan, S. L. Inelastic post-buckling behavior of tubular beam-columns and frames, *Engineering Structures*, **11** (1) (1989), 23–30.
22. Clarke, M. J., Plastic-zone analysis of frames, in *Advanced Analysis of Steel Frames*, ed. by W. F. Chen & S. Toma, CRC Press, Boca Raton, FL, 1994, Chap. 6.
23. Clarke, M. J., Bridge, R. Q., Hancock, G. J. & Trahair, N. S., Advanced analysis of steel building frames, *Journal of Constructional Steel Research*, **23** (1992), 1–29.
24. CRI, *CF77 Compiling System*, Vol. 4: *Parallel Processing Guide*, SG-3074 5.0, Cray Research, 1991.
25. CRI, *CF77 Compiling System*, Vol. I: *Fortran Reference Manual*, SR-3071 5.0, Cray Research, 1991.
26. Fahmy, M. W. & Namini, A. H., A survey of parallel nonlinear dynamic analysis methodologies, *Computers & Structures*, **53** (4) (1994), 1033–43.
27. Farhat, C., A simple and efficient automatic FEM domain decomposer, *Computers & Structures*, **28** (1988), 579–602.
28. Farhat, C., Computational strategies for finite element simulations on supercomputers with 4 to 65,536 processors, in *Computer Utilization in Structural Engineering*, ed. by J. K. Nelson, American Society of Civil Engineers, New York, 1989, pp. 177–86.
29. Farhat, C. & Wilson, E. L. A new finite element concurrent program architecture, *International Journal for Numerical Methods in Engineering*, **24** (1987), 1771–92.
30. Foley, C. M., Inelastic behavior of partially restrained steel frames using parallel processing and supercomputers, Ph.D. dissertation, Marquette University, 1996.
31. Foley, C. M. & Vinnakota, S., Nonlinear analysis of structural frameworks using supercomputing techniques, in *First Congress on Computing in Civil Engineering*, ASCE, Washington, 1994, pp. 2058–65.
32. Foley, C. M. & Vinnakota, S., Parallel processing in the elastic nonlinear elastic analysis of high-rise frameworks, *Computers & Structures*, **52** (6) (1994), 157–65.
33. Foley, C. M. & Vinnakota, S., Inelastic behavior of multistory partially restrained steel frames, part I, *Journal of Structural Engineering*, **128** (8) (1999), 854–61.
34. Foley, C. M. & Vinnakota, S., Inelastic behavior of multistory partially restrained steel frames, part II, *Journal of Structural Engineering*, **128** (8) (1999), 862–9.
35. Hajjar, J. F. & Abel, J. F., Parallel processing for transient nonlinear structural dynamics of three-dimensional framed structures using domain decomposition, *Computers & Structures*, **30** (1988), 1237–54.
36. Jaques, M. W., Ross, C. T. F. & Strickland, P., Exploiting inherent parallelism in nonlinear finite element analysis, *Computers & Structures*, **58** (4) (1996), 801–7.
37. King, W., White, D. W. & Chen, W. F., *A Modified Plastic Hinge Method for Second-Order Inelastic Analysis of Steel Rigid Frames*, Purdue University, West Lafayette, IN, 1990.
38. Kitipornchai, S., Al-Bermani, F. G. A. & Chan, S. L. Elasto-plastic finite element models for angle steel frames, *Journal of Structural Engineering*, **116** (10) (1988), 2567–81.
39. Lehigh University, *Plastic Design of Multistory Frames*, Report, No 273.20, Fritz Engineering Laboratory, Lehigh University, Bethlehem, PA, 1965.
40. Levesque, J. M. & Williamson, J. W., *A Guidebook to Fortran on Supercomputers*, Academic Press, New York, 1989.
41. Liew, J. Y.-R., White, D. W. & Chen, W.-F. Second-order refined plastic-hinge analysis for frame design, part I, *Journal of Structural Engineering*, **119** (11) (1993), 3196–3216.

42. Liew, J. Y.-R., White, D. W. & Chen, W.-F., Second-order refined plastic-hinge analysis for frame design, part II, *Journal of Structural Engineering*, **119** (11) (1993), 3217–37.
43. Meek, J. L. & Lin, W. J., Geometric and material nonlinear analysis of thin-walled beam-columns, *Journal of Structural Engineering*, **116** (6) (1991), 1473–90.
44. Orbison, J. G., Nonlinear static analysis of three-dimensional steel frames, Ph.D. dissertation, Cornell University, 1982.
45. Park, H. S. & Adeli, H., Data parallel neural dynamics model for integrated design of large steel structures, *Microcomputers in Civil Engineering*, **12** (5) (1997), 311–26.
46. Park, H. S. & Adeli, H., Distributed neural dynamics algorithms for optimization of large steel structures, *Journal of Structural Engineering*, **123** (7) (1997), 880–8.
47. Powell, G. & Simons, J., Improved iteration strategy for nonlinear structures, *International Journal for Numerical Methods in Engineering*, **17** (1981), 1455–67.
48. Saleh, A. & Adeli, H., Optimal control of adaptive/smart multistory building structures, *Computer-Aided Civil & Infrastructure Engineering*, **13** (6) (1998), 389–403.
49. Soegiarso, R. & Adeli, H., Optimization of large steel truss structures using standard cross-sections, *Engineering Journal*, **33** (3) (1996), 83–94.
50. Soegiarso, R. & Adeli, H., Optimization of large space frame steel structures, *Engineering Journal*, **34** (2) (1997), 54–60.
51. Soegiarso, R. & Adeli, H., Optimum load and resistance factor design of steel space frame structures, *Journal of Structural Engineering*, **123** (2) (1997), 184–92.
52. SSRC, Technical Memorandum No. 5, in Galambos, T. V., Ed., *Guide to Stability Design Criteria for Metal Structures*, 4th ed. John Wiley & Sons, 1992, pp. 732–4.
53. Sziveri, J. & Topping, B. H., Transient dynamic nonlinear analysis using MIMD computer architectures, *Journal of Computing in Civil Engineering*, **14** (2) (2000), 79–91.
54. Teh, L. H. Plastic-zone analysis of 3D steel frames using beam elements, *Journal of Structural Engineering*, **125** (11) (1999), 1328–37.
55. Toma, S. J. & Chen, W. F., European calibration frames for second-order inelastic analysis, *Engineering Structures*, **14** (1) (1992), 7–14.
56. Vinnakota, S., Inelastic stability analysis of rigid jointed frames (in French), Ph.D. dissertation, Swiss Federal Institute of Technology, 1967.
57. Vogel, U. Calibrating frames, *Stahlbau* (October 1985), 295–301.
58. White, D. W., Advanced analysis/design of a typical moment frame, in *10th Structures Congress Compact Papers*, ASCE, Washington, 1992, pp. 330–3.
59. White, D. W., Plastic hinge based methods for advanced analysis of structural steel frames, *Journal of Constructional Steel Research*, **24** (1993), 121–52.
60. White, D. W. & Abel, J. F., Bibliography on finite elements and supercomputing, *Communications in Applied Numerical Methods*, **4** (2) (1988), 279–94.
61. Yang, Y.-B., Linear and nonlinear analysis of space frames with nonuniform torsion using interactive computer graphics, Ph.D. dissertation, Cornell University, 1984.
62. Ziemian, R. D., Advanced methods of inelastic analysis in the limit states design of steel structures, Ph.D. dissertation, Cornell University, 1992.
63. Ziemian, R. D. & McGuire, W. A method of incorporating live load reduction provisions in frame analysis, *Engineering Journal*, **29** (1) (1992), 1–3.
64. Ziemian, R. D., McGuire, W. & Deierlein, G. G. Inelastic limit states design: I. Planar frame studies, *Journal of Structural Engineering*, **118** (9) (1992), 2532–49.
65. Ziemian, R. D., McGuire, W. & Deierlein, G. G. Inelastic limit states design: II. Planar frame studies, *Journal of Structural Engineering*, **118** (9) (1992), 2550–68.