Structural Engineering and Mechanics, Vol. 2, No. 2 (1994) 125-139
DOI: http://dx.doi.org/10.12989/sem.1994.2.2.125

Vector algorithm for reinforced concrete shell element stiffness matrix

Chang Shik Min †

Dept. of Ocean Civil Eng., Cheju National Univ., Cheju 690-756, Korea

Ajaya Kumar Gupta ‡

Center for Nuclear Power Plant Structures, Equipment and Piping, North Carolina State University, Raleigh, NC 27695-7908, U.S.A.

Abstract. A vector algorithm for calculating the stiffness matrices of reinforced concrete shell elements is presented. The algorithm is based on establishing vector lengths equal to the number of elements. The computational efficiency of the proposed algorithm is assessed on a Cray Y-MP supercomputer. It is shown that the vector algorithm achieves scalar-to-vector speedup of 1.7 to 7.6 on three moderate sized inelastic problems.

Key words: vector algorithm; stiffness matrix; inelastic finite element analysis; cray Y-MP; supercomputer.

1. Introduction

Currently, the design of reinforced concrete shells is performed based on stresses obtained from a linear elastic analysis and the reinforcing steel bars are provided according to pointwise limit state behavior (Gupta 1984b, ACI 1989). General applicability of this design practice is not yet established for reinforced concrete shells. Numerical simulations using computers have been quite common in recent years as an alternative to expensive experiments (Hand et al. 1973, Lin and Scordelis 1975, Milford and Schnobrich 1984, Akbar and Gupta 1985, Min and Gupta 1992). But the number and the size of the problems that could be studied have been limited due to the computing resources of conventional scalar machines (Akbar and Gupta 1985).

Introduction of supercomputers has brightened the prospects of solving many problems that could not be solved a few years ago. To achieve the full potential of supercomputers, not only do we need to reorganize the program structure, but also adopt alternate numerical algorithms in conjunction with the machine architecture and the vector compiler for the target machine (Lambiotte 1975, Dongarra et al. 1984, Noor and Peters 1986, Hughes et al. 1987, Sil-

† Lecturer
‡ Professor of Civil Engineering and Director

vester 1988, Storaasli et al. 1989, Min and Gupta 1991, Golub and Ortega 1993). Currently available supercomputers can be broadly categorized into two major groups: (1) vector computers, which consist of a few very powerful pipelined vector processors, for example, Cray Y-MP C90, IBM ES/9000, Alliant FX/Series, and Convex C-3 models, (2) massively parallel processing (MPP) computers, which usually have hundreds or thousands of inexpensive processors, such as INTEL's Paragon XP/S, NCUBE's Ncube, MasPar's MP line, Bolt, Beranek and Newman's BUTTERFLY, Thinking Machine's Connection Machine, and Kendall Square Research's KSR1. In vector supercomputers the performance is achieved through utilizing pipelined processors and vector optimization (Noor and Peters 1986, Silvester 1988, Min and Gupta 1991). Coarse-grain parallelization can be implemented to improve the performance further in machines that have a small number of processors (Hughes et al. 1987, Storaasli et al. 1989, Lou and Friedman 1991, Foresti et al. 1993, Yagawa et al. 1993). For MPP computers the performance is achieved by fine-grain parallelization (Malone 1988, Farhat et al. 1990, Hutchinson et al. 1991).

In the present study, a finite element computer program for inelastic analysis of reinforced concrete shells, originally developed by Akbar and Gupta (1985) for an IBM 3081 scalar computer, was vectorized and implemented on a Cray Y-MP at the North Carolina Supercomputing Center (NCSC). Earlier (1991), we had developed vector algorithms for a computer program for two-dimensional finite element analysis. We studied vector algorithms for four basic modules, namely computation of element stiffness matrices, assembly of the global stiffness matrix, solution of the system of linear simultaneous equations, and calculation of stress-strains with an 8-node isoparametric element for linear elastic analysis. Out of four modules the equation solver takes most of the computational effort in the analysis and consequently needs to be optimized the most. But, by optimizing the rest of the program, we could achieve an additional speedup of 10-40% from scalar to vector operations. In the case of nonlinear analysis, the computation for the stiffness matrix and stress-strain modules is increased substantially due to the iterative nature of nonlinear analysis and additional calculations for residual forces.

A vector algorithm for calculating the element stiffness matrices for uncracked and cracked reinforced concrete shell elements is presented here. A four-node superparametric element (Ahmad et al. 1970) is used. In the formulations presented here, we have retained the assumption made by Akbar and Gupta (1985) that the bending stresses do not affect the cracking of concrete and yielding of steel. This assumption was justified on the grounds that inplane-membrane stresses govern the behavior of many shells, for which, this program was used.

2. Formulation of shell element stiffness matrix

A stiffness matrix formulation similar to that proposed by Gupta and Mohraz (1972) and Gupta (1983) was used. The element stiffness matrix $[k]$ consists of nn x nn submatrices $[k_{ij}]$ of the order of 6x6 relating the forces at node i to the displacements at node j, where nn is the number of nodes in an element, and is 4 here. A submatrix $[k_{ij}]$ can be separated into two matrices, $[k_{P_{ij}}]$ for inplane stiffness, and $[k_{S_{ij}}]$ for transverse shear stiffness.

When the constitutive properties are constant on the surface of an element, the submatrix $[k_{P_{ij}}]$ can be written as

$$[k_{P_{ij}}]_{6\times 6} = h\begin{bmatrix} [P] & 0 \\ 0 & \frac{1}{3}[P] \end{bmatrix},\ [P]_{3\times 3} = \begin{bmatrix} (E_1+E_S\rho_{\bar{x}})[p_1]+E_2[p_2]+E_3[p_3] \\ +(E_4+E_S\rho_{\bar{y}})[p_4]+E_5[p_5]+E_6[p_6] \end{bmatrix}, \tag{1}$$

in which h is the thickness of the element and is assumed to be constant; E_1, E_2, E_3, E_4, E_5 and E_6 are the terms of the constitutive matrix for inplane concrete stresses, defined in the following equation

$$\begin{Bmatrix} \sigma_{\bar{x}} \\ \sigma_{\bar{y}} \\ \tau_{\bar{x}\bar{y}} \end{Bmatrix} = \begin{bmatrix} E_1 & E_2 & E_3 \\ & E_4 & E_5 \\ sym & & E_6 \end{bmatrix} \begin{Bmatrix} \varepsilon_{\bar{x}} \\ \varepsilon_{\bar{y}} \\ \gamma_{\bar{x}\bar{y}} \end{Bmatrix}, \tag{2}$$

where $[\sigma_{\bar{x}}\ \sigma_{\bar{y}}\ \tau_{\bar{x}\bar{y}}]^T$ and $[\varepsilon_{\bar{x}}\ \varepsilon_{\bar{y}}\ \gamma_{\bar{x}\bar{y}}]^T$ represent inplane stresses and strains with respect to the orthogonal shell coordinates, $\bar{x}$ and $\bar{y}$. For uncracked concrete elements, E_3 and E_5 are zero. For the steel part, $\sigma_{\bar{x}}=E_S\rho_{\bar{x}}\varepsilon_{\bar{x}}$, $\sigma_{\bar{y}}=E_S\rho_{\bar{y}}\varepsilon_{\bar{y}}$, in which E_S is the Young's modulus of steel, and $\rho_{\bar{x}}$ and $\rho_{\bar{y}}$ are the reinforcement ratios in the $\bar{x}$ and $\bar{y}$ directions, respectively. Steel is assumed to be elastic and perfectly plastic. In the calculation of stiffness matrices, E_S is assumed to remain constant even after $\varepsilon_{\bar{x}}$ or $\varepsilon_{\bar{y}}$ has exceeded the yield strains to avoid numerical problems. Any overestimation of corresponding stresses is compensated by applying appropriate residual forces.

The six p–matrices are defined below,

$$[p_1] = \int_A N_{i,\bar{x}}N_{j,\bar{x}}[\vec{n}_{\bar{x}}]^T[\vec{n}_{\bar{x}}]dA,$$

$$[p_2] = \int_A (N_{i,\bar{y}}N_{j,\bar{x}}[\vec{n}_{\bar{y}}]^T[\vec{n}_{\bar{x}}]+N_{i,\bar{x}}N_{j,\bar{y}}[\vec{n}_{\bar{x}}]^T[\vec{n}_{\bar{y}}])dA,$$

$$[p_3] = \int_A \begin{bmatrix} (N_{i,\bar{y}}N_{j,\bar{x}}+N_{i,\bar{x}}N_{j,\bar{y}})\ [\vec{n}_{\bar{x}}]^T[\vec{n}_{\bar{x}}] \\ +N_{i,\bar{x}}N_{j,\bar{x}}([\vec{n}_{\bar{y}}]^T[\vec{n}_{\bar{x}}]+[\vec{n}_{\bar{x}}]^T[\vec{n}_{\bar{y}}]) \end{bmatrix} dA,$$

$$[p_4] = \int_A N_{i,\bar{y}}N_{j,\bar{y}}[\vec{n}_{\bar{y}}]^T[\vec{n}_{\bar{y}}]dA, \tag{3}$$

$$[p_5] = \int_A \begin{bmatrix} (N_{i,\bar{y}}N_{j,\bar{x}}+N_{i,\bar{x}}N_{j,\bar{y}})\ [\vec{n}_{\bar{y}}]^T[\vec{n}_{\bar{y}}] \\ +N_{i,\bar{y}}N_{j,\bar{y}}([\vec{n}_{\bar{y}}]^T[\vec{n}_{\bar{x}}]+[\vec{n}_{\bar{x}}]^T[\vec{n}_{\bar{y}}]) \end{bmatrix} dA,$$

$$[p_6] = \int_A \begin{bmatrix} N_{i,\bar{y}}N_{j,\bar{y}}[\vec{n}_{\bar{x}}]^T[\vec{n}_{\bar{x}}]+N_{i,\bar{x}}N_{j,\bar{y}}[\vec{n}_{\bar{y}}]^T[\vec{n}_{\bar{x}}] \\ +N_{i,\bar{y}}N_{j,\bar{x}}[\vec{n}_{\bar{x}}]^T[\vec{n}_{\bar{y}}]+N_{i,\bar{x}}N_{j,\bar{x}}[\vec{n}_{\bar{y}}]^T[\vec{n}_{\bar{y}}] \end{bmatrix} dA,$$

$$dA = |J|\,d\xi d\eta,$$

where N_i's are the shape functions defined in terms of the local isoparametric coordinates (ξ, η); $|J|$ the determinant of the Jacobian matrix for transformation between ($\bar{x}$, $\bar{y}$) and (ξ, η) coordinate systems; $N_{i,\bar{x}}$ and $N_{i,\bar{y}}$ partial derivatives of N_i with respect to $\bar{x}$ and $\bar{y}$, respectively (as in this case, a subscript followed by a comma anywhere in this paper denotes a partial derivative); and $[\vec{n}_{\bar{x}}]$ and $[\vec{n}_{\bar{y}}]$ the row vectors of the direction cosines of the orthogonal shell coordinates $\bar{x}$ and $\bar{y}$, respectively, with respect to the global Cartesian coordinates (x, y, z). Another row vector of direction cosines $[\vec{n}_{\bar{z}}]$ is defined for a coordinate normal to the shell, and is introduced later. Formulas for evaluating these direction cosines are straight forward (Ahmad et al. 1970, Akbar and Gupta 1985), and are not presented here.

The submatrix $[k_{S_{ij}}]$ can be written as

$$[k_{S_{ij}}]_{6\times6} = E_7\, h \begin{bmatrix} q_{a_{11}} & \frac{2}{h} q_{a_{12}} \\ \frac{2}{h} q_{a_{21}} & (\frac{2}{h})^2 q_{a_{22}} \end{bmatrix} + E_7\, h \begin{bmatrix} q_{b_{11}} & \frac{2}{h} q_{b_{12}} \\ \frac{2}{h} q_{b_{21}} & (\frac{2}{h})^2 q_{b_{22}} \end{bmatrix}, \tag{4}$$

in which E_7 is transverse shear modulus, and submatrices $[q_a]$ and $[q_b]$ are defined as

$$\begin{bmatrix} q_{a_{11}} & q_{a_{12}} \\ q_{a_{21}} & q_{a_{22}} \end{bmatrix} = \int_A \begin{bmatrix} N_{i,\bar{x}} N_{j,\bar{x}} [\vec{n}_{\bar{z}}]^T [\vec{n}_{\bar{z}}] & N_{i,\bar{x}} N_j\, [\vec{n}_{\bar{z}}]^T [\vec{n}_{\bar{x}}] \\ N_i N_{j,\bar{x}} [\vec{n}_{\bar{x}}]^T [\vec{n}_{\bar{z}}] & N_i N_j\, [\vec{n}_{\bar{x}}]^T [\vec{n}_{\bar{x}}] \end{bmatrix} dA,$$

$$\begin{bmatrix} q_{b_{11}} & q_{b_{12}} \\ q_{b_{21}} & q_{b_{22}} \end{bmatrix} = \int_A \begin{bmatrix} N_{i,\bar{y}} N_{j,\bar{y}} [\vec{n}_{\bar{z}}]^T [\vec{n}_{\bar{z}}] & N_{i,\bar{y}} N_j\, [\vec{n}_{\bar{z}}]^T [\vec{n}_{\bar{y}}] \\ N_i N_{j,\bar{y}} [\vec{n}_{\bar{y}}]^T [\vec{n}_{\bar{z}}] & N_i N_j\, [\vec{n}_{\bar{y}}]^T [\vec{n}_{\bar{y}}] \end{bmatrix} dA, \tag{5}$$

In the original Akbar–Gupta program (1985), the transverse shear modulus parallel to the cracks was assumed to be zero. The assumption, however, led to numerical instability (Min and Gupta 1992). Any nonzero modulus would solve the numerical instability problem. We did not expect the results to be sensitive to the actual value of the modulus used. Therefore, we assumed the shear modulus to be the same in both transverse planes, parallel and perpendicular to the crack direction. This assumption is implied in Equation (4).

3. Numerical integration

Integrations in Equations (3) and (5) are performed numerically by using appropriate selective Gaussian quadrature schemes. To avoid shear-locking, before cracking of concrete, the stiffness terms related to ε_x and ε_y are integrated using a 2x2 quadrature, and those related to γ_{xy} by a 1x1 quadrature (Pawsey and Clough 1971). Once the concrete element cracks, the preceding selective integration scheme does not work because direct and shear stresses and strains become coupled. Gupta and Akbar (1984a) proposed an alternative selective integration technique for the cracked concrete elements. They assumed that all the strains are constant in a cracked element and calculated the stiffness matrix using a 1x1 quadrature. This uniformly reduced integration scheme after cracking of concrete would lead to a singular stiffness matrix. The problem of singularity is avoided by integrating the steel stiffness using a 2x2 quadrature. Before cracking, the stiffness terms related to γ_{xz} and γ_{yz} are integrated using 1x2 and 2x1 quadratures, respectively (Pawsey and Clough 1971). A 1x1 quadrature is used after concrete cracking (Akbar and Gupta 1985).

4. Transformation

The above formulation is developed in terms of the global coordinates. Six degrees of freedom are assumed at each node, three translations and three rotations. However, the shell element used here has only two rotational degrees of freedom, one each along the local shell coordinates and $\bar{x}$ and $\bar{y}$. That means that the element has zero rotational stiffness along the $\bar{z}$–coordinate. When the curvature at any node is either zero or very small, the global structural stiffness matrix becomes singular. To avoid this singularity problem, we transform the three global rotational degrees of freedom into two rotations at each node along the shell coordinates $\bar{x}$ and $\bar{y}$. Thus, each node has five global degrees of freedom after the transformation.

The transformation matrix at any node can be partitioned into four submatrices as

$$[T]_{6\times 5} = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} = \begin{bmatrix} [I]_{3\times 3} & [0]_{3\times 2} \\ [0]_{3\times 2} & [T_{22}]_{3\times 2} \end{bmatrix}, \quad [T_{22}] = \begin{bmatrix} \bar{x}_{,x} & \bar{y}_{,x} \\ \bar{x}_{,y} & \bar{y}_{,y} \\ \bar{x}_{,z} & \bar{y}_{,z} \end{bmatrix} \tag{6}$$

where [I] and [0] represent identity and null matrices, respectively. After pre– and post–multiplying the inplane stiffness submatrix given by Equation (1) with the transformation matrices, we get

$$[\overline{k_{P_{ij}}}]_{5\times 5} = \begin{bmatrix} h[\boldsymbol{P}] & 0 \\ 0 & [T_{22}]_i^T \frac{h}{3}[\boldsymbol{P}]\,[T_{22}]_j \end{bmatrix}. \tag{7}$$

For transverse shear, $[k_{S_{ij}}]$ of Equation (4) can be similarly transformed into

$$[\overline{k_{S_{ij}}}]_{5\times 5} = E_7 \begin{bmatrix} h[q_{11}] & 2[q_{12}][T_{22}]_j \\ [T_{22}]_i^T\, 2[q_{21}] & [T_{22}]_i^T \frac{4}{h}\,[q_{22}]\,[T_{22}]_j \end{bmatrix}, \tag{8}$$

where

$$\begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} = \begin{bmatrix} q_{a_{11}}+q_{b_{11}} & q_{a_{12}}+q_{b_{12}} \\ q_{a_{21}}+q_{b_{21}} & q_{a_{22}}+q_{b_{22}} \end{bmatrix}.$$

5. Vector Algorithm

Vectorization is achieved by establishing vector lengths equal to the number of elements whenever possible. All the $\boldsymbol{p}$ and $\boldsymbol{q}$–matrices are independent of the constitutive properties. The matrices are calculated once and stored. We use these matrices whenever we need to update a stiffness matrix for new constitutive properties in an iteration.

There is a problem associated with keeping the vector length equal to the number of elements. As explained earlier, the quadrature used for the numerical integration of the stiffness matrix changes after an element cracks. To avoid a complete scalar operation, we adopt a modified scheme. First, a set of element stiffness matrices is calculated assuming that all the elements are uncracked, making use of vector operations. Then, we update each cracked element stiffness matrix using an index array pointing to the cracked elements by performing scalar operations. As we would expect, the scheme works well when the structure develops a relatively small number of cracked elements.

The vector algorithm is summarized in Tables 1–5. Tables 1–4 show the vector algorithms for stiffness matrix terms to be integrated using 2x2, 1x1, 1x2 and 2x1 quadrature points, respectively, and need to be computed only once, and not recalculated for each iteration. Table 5 presents the vector algorithm for combining stiffness terms integrated using various quadrature rules and for evaluating transformed stiffness matrices. Therefore, only the computational steps of Table 5 are repeated for each iteration with updated constitutive properties.

Table 1 Algorithm for stiffness matrix terms to be integrated using 2x2 quadrature points

Step	Description	Array sizes
1	Evaluate at each integration point, $N_{i,\xi}$, $N_{i,\eta}$	2x(nn,4)
2*	Compute the elements of the Jacobian matrices $\boldsymbol{J}$, $x_{,\xi}$, $y_{,\xi}$, $x_{,\eta}$, $y_{,\eta}$	(ne,4)
	Compute the direction cosines, $\boldsymbol{n} = [\boldsymbol{n}_{\bar{x}}\ \boldsymbol{n}_{\bar{y}}\ \boldsymbol{n}_{\bar{z}}]$ $\boldsymbol{n}_{\bar{x}} = [\bar{x}_{,x}\ \bar{x}_{,y}\ \bar{x}_{,z}]$, $\boldsymbol{n}_{\bar{y}} = [\bar{y}_{,x}\ \bar{y}_{,y}\ \bar{y}_{,z}]$, $\boldsymbol{n}_{\bar{z}} = [\bar{z}_{,x}\ \bar{z}_{,y}\ \bar{z}_{,z}]$	(ne,9,ni)
3*	Evaluate the determinant of the Jacobian and its inverse, $\lvert \boldsymbol{J} \rvert$, $U = 1/\lvert \boldsymbol{J} \rvert$	2x(ne)
	Compute, $N_{i,\bar{x}} = U(N_{i,\xi}y_{,\eta} - N_{i,\eta}y_{,\xi})$, $N_{i,\bar{y}} = U(-N_{i,\xi}x_{,\eta} + N_{i,\eta}x_{,\xi})$	2x(ne,nn,ni)
4*	Compute, $\bar{w} = \lvert \boldsymbol{J} \rvert \cdot h$ For a 2 x 2 quadrature, a weight coefficient of unity is implied.	(ne,ni)
5*	Calculate; for i=1,nn; j=i,nn $\begin{bmatrix} DN_1 & DN_2 \\ DN_3 & DN_4 \end{bmatrix} = \bar{w} \begin{bmatrix} N_{i,\bar{x}}N_{j,\bar{x}} & N_{i,\bar{x}}N_{j,\bar{y}} \\ N_{i,\bar{y}}N_{j,\bar{x}} & N_{i,\bar{y}}N_{j,\bar{y}} \end{bmatrix}$	(ne,4nc)
6*	Compute the 21 upper triangular terms of the 6x6 matrix $[\boldsymbol{n}_{\bar{x}}\boldsymbol{n}_{\bar{y}}]^T\ [\boldsymbol{n}_{\bar{x}}\boldsymbol{n}_{\bar{y}}]$ and store in the array TT.	(ne,21,ni)
7*	Compute $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and $\boldsymbol{p}_4$ to be used for uncracked elements. (For any pair of nodes i,j for an element, the size of each $\boldsymbol{p}$-array is 3x3. Due to symmetry, we store only the 6 upper triangular terms for $\boldsymbol{p}_1$ and $\boldsymbol{p}_4$.)	
	$\boldsymbol{p}_1 = \boldsymbol{p}_1 + DN_1\ TT(1,2,3,7,8,12)$	(ne,6nc)
	$\boldsymbol{p}_2 = \boldsymbol{p}_2 + DN_3\ TT(4,5,6,9,10,11,13,14,15)$ $+ DN_2\ TT(4,9,13,5,10,14,6,11,15)$	(ne,9nc)
	$\boldsymbol{p}_4 = \boldsymbol{p}_4 + DN_4\ TT(16,17,18,19,20,21)$	(ne,6nc)
	All $\boldsymbol{p}$-arrays are initialized before the operation.	

ne = number of elements,

ni = number of integration points = 4,

nn = number of nodes in an element = 4,

nc = number of the upper triangular coefficients in an nn x nn matrix = nn(nn+1)/2 = 10,

* Steps 2–7 are performed in a do loop on the integration points.

Table 2 Algorithm for stiffness matrix terms to be integrated using 1x1 quadrature point

Step	Description	Array sizes
8	Evaluate the integration point, $N_{i,\xi}$, $N_{i,\eta}$	2x(nn)
9	Compute the elements of the Jacobian matrices $\boldsymbol{J}$, $\boldsymbol{x}_{,\xi}$, $\boldsymbol{y}_{,\xi}$, $\boldsymbol{x}_{,\eta}$, $\boldsymbol{y}_{,\eta}$	(ne,4)
	Compute the direction cosines, $\boldsymbol{n} = [\boldsymbol{n}_{\bar{x}}\, \boldsymbol{n}_{\bar{y}}\, \boldsymbol{n}_{\bar{z}}]$	(ne,9)
10	Evaluate the determinant of the Jacobian and its inverse, $\lvert \boldsymbol{J} \rvert$, $U = 1/\lvert \boldsymbol{J} \rvert$	2x(ne)
	Compute, $N_{i,\bar{x}} = U(N_{i,\xi}\boldsymbol{y}_{,\eta} - N_{i,\eta}\boldsymbol{y}_{,\xi})$, $N_{i,\bar{y}} = U(-N_{i,\xi}\boldsymbol{x}_{,\eta} + N_{i,\eta}\boldsymbol{x}_{,\xi})$	2x(ne,nn)
	Compute, $NN = \frac{1}{4}(N_{i,\bar{x}}, N_{i,\bar{y}})$ to be used for stiffness terms corresponding to shear strain components in cracked elements.	(ne,2nn)
11	Compute, $\bar{\boldsymbol{w}} = \lvert \boldsymbol{J} \rvert \cdot (W\boldsymbol{h})$, in which W is the quadrature weight coefficient and is equal to 4.	(ne)
12	Compute; for i=1,nn; j=i,nn $\begin{bmatrix} \boldsymbol{DN}_5 & \boldsymbol{DN}_6 \\ \boldsymbol{DN}_7 & \boldsymbol{DN}_8 \end{bmatrix} = \begin{bmatrix} N_{i,\bar{x}}N_{j,\bar{x}} & N_{i,\bar{x}}N_{j,\bar{y}} \\ N_{i,\bar{y}}N_{j,\bar{x}} & N_{i,\bar{y}}N_{j,\bar{y}} \end{bmatrix}$	(ne,4nc)
	Calculate; for i=1,nc; $\boldsymbol{DN}_T = \boldsymbol{DN}_6 + \boldsymbol{DN}_7$	(ne,nc)
13	Calculate, $\tilde{\boldsymbol{n}}_{xy} = \bar{\boldsymbol{w}}[\boldsymbol{n}_{\bar{x}}\boldsymbol{n}_{\bar{y}}]$	(ne,6)
	Compute the 21 upper triangular terms of the integral of the 6x6 matrix $\tilde{\boldsymbol{n}}_{xy}^T\, [\boldsymbol{n}_{\bar{x}}\boldsymbol{n}_{\bar{y}}]$ and store in the array $\boldsymbol{PR}$ $\boldsymbol{PR} = \tilde{\boldsymbol{n}}_{xy}^T\, [\boldsymbol{n}_{\bar{x}}\boldsymbol{n}_{\bar{y}}]$	(ne,21)
14	Calculate $\boldsymbol{p}_6$ to be used for E_6 term, $\boldsymbol{p}_6 = \boldsymbol{DN}_5\, \boldsymbol{PR}(16,17,18,17,19,20,18,20,21)$ $+\boldsymbol{DN}_6\, \boldsymbol{PR}(4,9,13,5,10,14,6,11,15)$ $+\boldsymbol{DN}_7\, \boldsymbol{PR}(4,5,6,9,10,11,13,14,15)$ $+\boldsymbol{DN}_8\, \boldsymbol{PR}(1,2,3,2,7,8,3,8,12)$	(ne,9nc)
15	Compute the 27 terms of the integral of the 3x(3x3) matrix $\bar{\boldsymbol{w}}[\boldsymbol{n}_{\bar{x}}^T\, \boldsymbol{n}_{\bar{z}}, \boldsymbol{n}_{\bar{y}}^T\, \boldsymbol{n}_{\bar{z}}, \boldsymbol{n}_{\bar{z}}^T\, \boldsymbol{n}_{\bar{z}}]$ and store in the array $\boldsymbol{QS}$ in row-wise vector $\boldsymbol{QS} = \bar{\boldsymbol{w}}[\boldsymbol{n}_{\bar{x}}^T\, \boldsymbol{n}_{\bar{z}}, \boldsymbol{n}_{\bar{y}}^T\, \boldsymbol{n}_{\bar{z}}, \boldsymbol{n}_{\bar{z}}^T\, \boldsymbol{n}_{\bar{z}}]$	(ne,27)

ne = number of elements,
nn = number of nodes in an element = 4,
nc = number of the upper triangular coefficients in an nn x nn matrix = nn(nn+1)/2 = 10.

Table 3 Algorithm for stiffness matrix terms to be integrated using 1x2 quadrature points

Step	Description	Array sizes
16	Evaluate N_i, $N_{i,\xi}$, $N_{i,\eta}$ at each integration point.	3x(nn,2)
17*	Compute the elements of the Jacobian matrices $\boldsymbol{J}$, $\boldsymbol{x}_{,\xi}$, $\boldsymbol{y}_{,\xi}$, $\boldsymbol{x}_{,\eta}$, $\boldsymbol{y}_{,\eta}$	(ne,4)
	Compute the direction cosines, $\boldsymbol{n} = [\boldsymbol{n}_{\bar{z}} \boldsymbol{n}_{\bar{x}}]$ $\boldsymbol{n}_{\bar{z}} = [\,\bar{z}_{,x}\, \bar{z}_{,y}\, \bar{z}_{,z}\,]$, $\boldsymbol{n}_{\bar{x}} = [\bar{x}_{,x}\, \bar{x}_{,y}\, \bar{x}_{,z}]$	(ne,6,ni)
18*	Evaluate the determinant of the Jacobian and its inverse, $\lvert \boldsymbol{J} \rvert$, $U = 1/\lvert \boldsymbol{J} \rvert$	2x(ne)
	Compute, $N_{i,\bar{x}} = U(N_{i,\xi}\boldsymbol{y}_{,\eta} - N_{i,\eta}\boldsymbol{y}_{,\xi})$,	(ne,nn,ni)
19*	Compute, $\bar{w} = \lvert \boldsymbol{J} \rvert \cdot (W\boldsymbol{h})$, in which W is the quadrature weight coefficient and is equal to 2.	(ne,ni)
20*	Calculate; for i=1,nn; j=i,nn $$\begin{bmatrix} \boldsymbol{DM}_1 & \boldsymbol{DM}_2 \\ \boldsymbol{DM}_3 & \boldsymbol{DM}_4 \end{bmatrix} = \bar{w} \begin{bmatrix} N_{i,\bar{x}}N_{j,\bar{x}} & N_{i,\bar{x}}N_j \\ N_iN_{j,\bar{x}} & N_iN_j \end{bmatrix}$$	(ne,4nc)
21*	Compute the 21 upper triangular terms of the 6x6 matrix $[\boldsymbol{n}_{\bar{z}}\boldsymbol{n}_{\bar{x}}]^T\ [\boldsymbol{n}_{\bar{z}}\boldsymbol{n}_{\bar{x}}]$ and store in the array $\boldsymbol{TR}$.	(ne,21,ni)
22*	Compute $q_{a_{11}}$, $q_{a_{12}}$, $q_{a_{21}}$ and $q_{a_{22}}$ to be used for uncracked elements. (Due to symmetry, we store only the 6 upper triangular terms for $q_{a_{11}}$ and $q_{a_{22}}$.)	
	$q_{a_{11}} = q_{a_{11}} + \boldsymbol{DM}_1\, \boldsymbol{TR}(1,2,3,7,8,12)$	(ne,6nc)
	$q_{a_{12}} = q_{a_{12}} + \frac{2}{\boldsymbol{h}} \boldsymbol{DM}_2\, \boldsymbol{TR}(4,5,6,9,10,11,13,14,15)$	(ne,9nc)
	$q_{a_{21}} = q_{a_{21}} + \frac{2}{\boldsymbol{h}} \boldsymbol{DM}_3\, \boldsymbol{TR}(4,9,13,5,10,14,6,11,15)$	(ne,9nc)
	$q_{a_{22}} = q_{a_{22}} + \frac{4}{\boldsymbol{h}^2} \boldsymbol{DM}_4\, \boldsymbol{TR}(16,17,18,19,20,21)$	(ne,6nc)
	All q_a-arrays are initialized before the operation.	

ne = number of elements,
ni = number of integration points = 2,
nn = number of nodes in an element = 4,
nc = number of the upper triangular coefficients in an nn x nn matrix = nn(nn+1)/2 = 10,
* Steps 17–22 are performed in a do loop on the integration points.

Table 4 Algorithm for stiffness matrix terms to be integrated using 2x1 quadrature points

Step	Description	Array sizes
23	Evaluate N_i, $N_{i,\xi}$, $N_{i,\eta}$ at each integration point	3x(nn,2)
24*	Compute the elements of the Jacobian matrices J, $x_{,\xi}$, $y_{,\xi}$, $x_{,\eta}$, $y_{,\eta}$	(ne,4)
	Compute the direction cosines, $n = [n_{\bar z} n_{\bar y}]$ $n_{\bar z} = [\bar z_{,x} \bar z_{,y} \bar z_{,z}]$, $n_{\bar y} = [\bar y_{,x} \bar y_{,y} \bar y_{,z}]$	(ne,6,ni)
25*	Evaluate the determinant of the Jacobian and its inverse, $\lvert J \rvert$, $U = 1/\lvert J \rvert$	2x(ne)
	Compute, $N_{i,\bar y} = U(-N_{i,\xi} x_{,\eta} + N_{i,\eta} x_{,\xi})$	(ne,nn,ni)
26*	Compute, $\bar w = \lvert J \rvert \cdot (Wh)$, in which W is the quadrature weight coefficient and is equal to 2.	(ne,ni)
27*	Calculate; for i=1,nn; j=i,nn $\begin{bmatrix} DM_5 & DM_6 \\ DM_7 & DM_8 \end{bmatrix} = \bar w \begin{bmatrix} N_{i,\bar y} N_{j,\bar y} & N_{i,\bar y} N_j \\ N_i N_{j,\bar y} & N_i N_j \end{bmatrix}$	(ne,4nc)
28*	Compute the 21 upper triangular terms of the 6x6 matrix $[n_{\bar z} n_{\bar y}]^T$ $[n_{\bar z} n_{\bar y}]$ and store in the array TR.	(ne,21,ni)
29*	Compute $q_{b_{11}}$, $q_{b_{12}}$, $q_{b_{21}}$ and $q_{b_{22}}$ to be used for uncracked elements. (Due to symmetry, we store only the 6 upper triangular terms for $q_{b_{11}}$ and $q_{b_{22}}$.)	
	$q_{b_{11}} = q_{b_{11}} + DM_5\, TR(1,2,3,7,8,12)$	(ne,6nc)
	$q_{b_{12}} = q_{b_{12}} + \frac{2}{h} DM_6\, TR(4,5,6,9,10,11,13,14,15)$	(ne,9nc)
	$q_{b_{21}} = q_{b_{21}} + \frac{2}{h} DM_7\, TR(4,9,13,5,10,14,6,11,15)$	(ne,9nc)
	$q_{b_{22}} = q_{b_{22}} + \frac{4}{h^2} DM_8\, TR(16,17,18,19,20,21)$	(ne,6nc)
	All q_b-arrays are initialized before the operation.	
30	Sum q_a and q_b, to form q vectors for uncracked element,	
	$q_{11} = q_{a_{11}} + q_{b_{11}}$, $q_{12} = q_{a_{12}} + q_{b_{12}}$,	(ne,6nc), (ne,9nc)
	$q_{21} = q_{a_{21}} + q_{b_{21}}$, $q_{22} = q_{a_{22}} + q_{b_{22}}$,	(ne,9nc), (ne,6nc)

ne = number of elements,
ni = number of integration points = 2,
nn = number of nodes in an element = 4,
nc = number of the upper triangular coefficients in an nn x nn matrix = nn(nn+1)/2 = 10,
* Steps 24–29 are performed in a do loop on the integration points.

Table 5 Algorithm for combining stiffness terms integrated using various quadrature rules and for evaluating transformed stiffness matrices

Step Description Array sizes

Elements of the submatrix $[\boldsymbol{k}_{ij}]$ are stored in the following order

$$[\boldsymbol{k}_{ij}] = \begin{bmatrix} [\boldsymbol{k}_{ij}]_{11} & [\boldsymbol{k}_{ij}]_{12} \\ [\boldsymbol{k}_{ij}]_{21} & [\boldsymbol{k}_{ij}]_{22} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 \\ 19 & 20 & 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 & 29 & 30 \\ 31 & 32 & 33 & 34 & 35 & 36 \end{bmatrix}$$

31 Compute the concrete contributions (2x2 quadrature) to the terms of the $[\boldsymbol{k}_{P_{ij}}]_{11}$ matrix assuming elements to be uncracked; for i = 1,nn; j = i,nn

$$\boldsymbol{k}_{P_{ij}}(1,2,3,7,8,9,13,14,15) = \begin{cases} \boldsymbol{E}_1\,\boldsymbol{p}_1(1,2,3,2,4,5,3,5,6) \\ +\boldsymbol{E}_2\,\boldsymbol{p}_2(1,2,3,4,5,6,7,8,9) \\ +\boldsymbol{E}_4\,\boldsymbol{p}_4(1,2,3,2,4,5,3,5,6) \end{cases}$$ (ne,9nc)

32# Re-calculate the concrete contributions (1x1 quadrature) for cracked elements; for i = 1,nn; j = i,nn (9nc)

$$\begin{aligned} &\boldsymbol{k}_{P_{ij}}(1,2,3,7,8,9,13,14,15) \\ &= (E_1\,DN_5 + E_3\,DN_T)\ \boldsymbol{PR}(1,2,3,2,7,8,3,8,12) \\ &+ (E_2\,DN_6 + E_3\,DN_5 + E_5\,DN_8)\ \boldsymbol{PR}(4,5,6,9,10,11,13,14,15) \\ &+ (E_2\,DN_7 + E_3\,DN_5 + E_5\,DN_8)\ \boldsymbol{PR}(4,9,13,5,10,14,6,11,15) \\ &+ (E_4\,DN_8 + E_5\,DN_T)\ \boldsymbol{PR}(16,17,18,17,19,20,18,20,21) \end{aligned}$$

33 Include the steel contributions (2x2 quadrature) and E_6 concrete terms (1x1 quadrature); for i = 1,nn; j = i,nn

$$\boldsymbol{k}_{P_{ij}}(1,2,3,7,8,9,13,14,15) = \begin{cases} \boldsymbol{k}_{P_{ij}}(1,2,3,7,8,9,13,14,15) \\ +\boldsymbol{E}_{\bar{x}}\,\boldsymbol{p}_1(1,2,3,2,4,5,3,5,6) \\ +\boldsymbol{E}_{\bar{y}}\,\boldsymbol{p}_4(1,2,3,2,4,5,3,5,6) \\ +\boldsymbol{E}_6\,\boldsymbol{p}_6(1,2,3,4,5,6,7,8,9) \end{cases}$$ (ne,9nc)

34 Compute $[\boldsymbol{k}_{S_{ij}}]$ assuming elements to be uncracked; for i = 1,nn; j = i,nn (ne,4x9,nc)

$$\boldsymbol{k}_{S_{ij}}(1,2,3,7,8,9,13,14,15) = \boldsymbol{E}_7\,\boldsymbol{q}_{11}(1,2,3,2,4,5,3,5,6)$$

$$\boldsymbol{k}_{S_{ij}}(4,5,6,10,11,12,16,17,18) = \boldsymbol{E}_7\,\boldsymbol{q}_{12}(1,2,3,4,5,6,7,8,9)$$

$$\boldsymbol{k}_{S_{ij}}(19,20,21,25,26,27,31,32,33) = \boldsymbol{E}_7\,\boldsymbol{q}_{21}(1,2,3,4,5,6,7,8,9)$$

$$\boldsymbol{k}_{S_{ij}}(22,23,24,28,29,30,34,35,36) = \boldsymbol{E}_7\,\boldsymbol{q}_{22}(1,2,3,2,4,5,3,5,6)$$

35# Re-calculate $[\boldsymbol{k}_{S_{ij}}]$ for cracked elements; i = 1,nn; j = i,nn (4x9,nc)

$$\boldsymbol{k}_{S_{ij}}(1,2,3,7,8,9,13,14,15) = E_7(DN_5+DN_8)\ \ \boldsymbol{QS}(19,20,21,22,23,24,25,26,27)$$

$$\boldsymbol{k}_{S_{ij}}(4,5,6,10,11,12,16,17,18) = \frac{2}{h}E_7\left\{\begin{array}{l} NN(i)\ \boldsymbol{QS}(1,4,7,2,5,8,3,6,9) \\ +NN(i+4)\ \boldsymbol{QS}(10,13,16,11,14,17,12,15,18)\end{array}\right.$$

$$\boldsymbol{k}_{S_{ij}}(19,20,21,25,26,27,31,32,33) = \frac{2}{h}E_7\left\{\begin{array}{l} NN(j)\ \boldsymbol{QS}(1,2,3,4,5,6,7,8,9) \\ +NN(j+4)\ \boldsymbol{QS}(10,11,12,13,14,15,16,17,18)\end{array}\right.$$

$$\boldsymbol{k}_{S_{ij}}(22,23,24,28,29,30,34,35,36) = \frac{E_7}{4h^2}\left\{\begin{array}{l} \boldsymbol{PR}(1,2,3,2,7,8,3,8,12) \\ +\boldsymbol{PR}(16,17,18,17,19,20,18,20,21)\end{array}\right.$$

36 Sum $[\boldsymbol{k}_{P_{ij}}]$ and $[\boldsymbol{k}_{S_{ij}}]$, to form (6x6) x nc element stiffness matrices;
for i = 1,nn; j = i,nn

$$\boldsymbol{k}_{S_{ij}}(1,2,3,7,8,9,13,14,15) = \boldsymbol{k}_{S_{ij}}(1,2,3,7,8,9,13,14,15) + \boldsymbol{k}_{P_{ij}}(1,2,3,7,8,9,13,14,15)$$

$$\boldsymbol{k}_{S_{ij}}(22,23,24,28,29,30,34,35,36) = \boldsymbol{k}_{S_{ij}}(22,23,24,28,29,30,34,35,36) + \frac{1}{3}\boldsymbol{k}_{P_{ij}}(22,23,24,28,29,30,34,35,36)$$

37 Perform transformation to obtain (5x5) x nc element stiffness matrices.
The transformed $[\overline{\boldsymbol{k}_{ij}}]$ is stored as

$$[\overline{\boldsymbol{k}_{ij}}] = \begin{bmatrix} [\overline{\boldsymbol{k}_{ij}}]_{11} & [\overline{\boldsymbol{k}_{ij}}]_{12} \\ [\overline{\boldsymbol{k}_{ij}}]_{21} & [\overline{\boldsymbol{k}_{ij}}]_{22} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix} \qquad \text{(ne,ns)}$$

The transformation matrix* at node i is stored as

$$[\boldsymbol{T}_i] = \begin{bmatrix} \bar{x}_{,x} & \bar{y}_{,x} \\ \bar{x}_{,y} & \bar{y}_{,y} \\ \bar{x}_{,z} & \bar{y}_{,z} \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \qquad \text{(ne,6,nn)}$$

For i = 1,nn; j = i,nn

$$\boldsymbol{k}_{ij}(1,2,3,6,7,8,11,12,13) = \boldsymbol{k}_{S_{ij}}(1,2,3,7,8,9,13,14,15)$$

$$\boldsymbol{k}_{ij}(4,5,9,10,14,15) = \begin{cases} \boldsymbol{k}_{S_{ij}}(4,4,10,10,16,16) & T_j(1,2,1,2,1,2) \\ +\boldsymbol{k}_{S_{ij}}(5,5,11,11,17,17) & T_j(3,4,3,4,3,4) \\ +\boldsymbol{k}_{S_{ij}}(6,6,12,12,18,18) & T_j(5,6,5,6,5,6) \end{cases}$$

$$\boldsymbol{k}_{ij}(16,17,18,21,22,23) = \begin{cases} T_i(1,1,1,2,2,2) & \boldsymbol{k}_{S_{ij}}(19,20,21,19,20,21) \\ +T_i(3,3,3,4,4,4) & \boldsymbol{k}_{S_{ij}}(25,26,27,25,26,27) \\ +T_i(5,5,5,6,6,6) & \boldsymbol{k}_{S_{ij}}(31,32,33,31,32,33) \end{cases}$$

Compute a temporary matrix ***TM***,

$$\boldsymbol{TM} = \begin{cases} T_i(1,1,1,2,2,2) & \boldsymbol{k}_{S_{ij}}(22,23,24,22,23,24) \\ +T_i(3,3,3,4,4,4) & \boldsymbol{k}_{S_{ij}}(28,29,30,28,29,30) \\ +T_i(5,5,5,6,6,6) & \boldsymbol{k}_{S_{ij}}(34,35,36,34,35,36) \end{cases} \qquad (\text{ne},6)$$

$$\boldsymbol{k}_{ij}(19,20,24,25) = \begin{cases} \boldsymbol{TM}(1,1,4,4) & T_j(1,2,1,2) \\ +\boldsymbol{TM}(2,2,5,5) & T_j(3,4,3,4) \\ +\boldsymbol{TM}(3,3,6,6) & T_j(5,6,5,6) \end{cases}$$

ne = number of elements,
nn = number of nodes in an element = 4,
nc = number of the upper triangular coefficients in an nn x nn matrix = nn(nn+1)/2 = 10,
ns = total number of the upper triangular blocks of coefficients in an nn x nn transformed matrix = nc x (5x5) = 250.
# Steps 32 and 35 are performed on the cracked element by an index array.
* The nodal transformation matrices are computed only once. (Rest of the steps in the table are repeated for each iteration.)

6. Performance evaluation of the algorithm

To measure the efficiency of the algorithm presented in this paper, two structures, a hyperbolic paraboloid (HP) saddle shell and a hyperbolic cooling tower, are used. The HP saddle shell is discretized into two models and the hyperbolic cooling tower is discretized into one model with a varying number of elements to measure the effect of the size of the problem on the efficiency of the algorithm. A summary of the three models is given in Table 6.

Cray Y–MP computers have the capability to automatically optimize and vectorize a computer program written for scalar machines. This can be accomplished either using the 'full' or the 'novector'-option (Cray SR–0018 C). The speedup with the full-option is usually greater than that with the novector–option. However, the full-option often gives erroneous results, as was the case when we ran the Akbar–Gupta program. We were able to implement the program successfully with the novector–option. Without any vector programming on our part, the use of the novector–option alone speeded up the CPU time for the HP saddle shell model S16 by

Table 6 Parameters of the models

	HP saddle shell		Cooling Tower
Model	S16	S32	C24
Number of elements	89	305	432
Number of nodes	133	389	475

a factor of 3.8 for the solution of the complete problem (not the evaluation of the stiffness matrix alone).

Because of the greater reliability of the novector–option over the full–option for automatic vectorization and optimization of a scalar program, we are using the novector–option as the basis for the scalar program to measure speedup due to vectorization of the algorithm. The version of the program specifically vectorized for the Cray Y–MP machine runs successfully with the full–option, which is used here.

Table 7 summarizes the CPU times taken in the calculations of stiffness matrices using the original Akbar–Gupta program and our program that employs the proposed algorithm, for the three models listed in Table 6. The CPU times given in Table 7 do not include the portion of the time taken to perform the calculations outlined in Tables 1–4. Because those calculations are performed only once, and those in Table 5 are repeated for each iteration, the former are insignificant.

Table 7 Comparison of the performance of the scalar algorithm and the present vector algorithm for element stiffness calculation

Model	Iterations*	CPU time (seconds)**		Speedup***
		Scalar	Vector	
S16	265	4.3	2.5	1.7
S32	493	19.2	6.1	3.1
C24	436	72.9	9.6	7.6

* Total number of iterations for inelastic analysis.

** Time was measured using CPU timer SECOND function of the Cray Math library (Cray SR–2081 6.0).

*** Speedup = scalar/vector

For the three models, the speedup due to the vectorization of the stiffness matrix algorithm ranges from 1.7 to 7.6. It is observed that the speedup is greater for models with a greater number of elements which determines the vector length. As one would expect, the number of cracked elements also has a strong influence on the speedup. The speedup for the saddle shell model S32 with 305 elements is only 3.1 as compared to the cooling tower model with 432 elements for which the speedup is 7.6. Thus, the cooling tower model with only 40% more elements than the saddle shell model, has a speedup that is 2.5 times that for the saddle shell model. The obvious reason is that the proportion of the number of cracked elements in the saddle shell model is much greater than that for the cooling tower model (Min and Gupta 1992).

7. Conclusion

A vector algorithm for calculating the stiffness matrices of uncracked and cracked reinforced concrete shell elements is presented. The algorithm is based on establishing vector lengths equal to the number of elements assuming that all the elements are uncracked. The stiffness matrices of the cracked elements are calculated separately. The computational efficiency of the proposed algorithm is assessed on a Cray Y–MP supercomputer and shows scalar –to–vector speedup of 1.7 to 7.6 on three moderate sized inelastic problems. As the number of elements increases and the number of cracked elements decreases, the speedup increases.

Acknowledgements

The supercomputing resources were provided by the North Carolina Supercomputing Center, Research Triangle Park, North Carolina.

References

ACI 318–89. (1989), *Building code requirements for reinforced concrete (ACI 318–89),* American Concrete Institute, Box 19150, Redford Station Detroit, Michigan 48219.

Ahmad, Sohrabuddin, Irons, Bruce M., and Zienkiewicz, O. C. (1970), "Analysis of thick and thin shell structures by curved finite elements." *Int. J. for Numer. Meth. in Eng.,* **2**, 419–451.

Akbar, Habibollah, and Gupta, Ajaya K. (1985), "Membrane reinforcement in concrete shells: design versus nonlinear behavior." North Carolina State University, Raleigh, North Carolina 27695–7908, January. *Reinforced Concrete Shell Research Report.*

Cray SR–0018 C, *CFT77 reference manual, SR–0018 C,* Cray Research, Inc.

Cray SR–2081 6.0, *UNICOS math and scientific library reference manual,* Cray Research, Inc.

Dongarra, Jack J., and Eisenstat, Stanley C. (1984), "Squeezing the most out of an algorithm in Cray FORTRAN." *ACM Transactions on Mathematical Software,* **10**(30), 219–230.

Farhat, C., Sobh, N., and Park, K. C. (1990), "Transient finite element computations on 65536 processors: the connection machine." *Int. J. for Numer. Methods in Eng.,* **30**, 27–55.

Foresti, S., Hassanzadeh, S., Murakami, H., and Sonnad, V, (1993), "Parallel rapid operator for iterative finite element solvers on a sheared memory machine," *Parallel Computing,* **19**, 1–7.

Golub, Gene and Ortega, James M., (1993), *Scientific computing–An introduction with parallel computing.* Academic Press, Inc., San Diago, CA 92101–4311.

Gupta, A. K. and Mohraz, B., (1973), "A method of computing numerically integrated stiffness matrices," *Int. J. Numer. Methods in Eng.*, **5**, 83–89.

Gupta A. K. (1983), "Efficient numerical integration of element stiffness matrices," Short Communication, *Int. J. Numer. Methods in Eng.*, **19**, 1410–1413.

Gupta, A. K., and Akbar, H. (1984a), "Cracking in reinforced concrete analysis." *J. Struct. Engrg.*, **110**(8), 1735–1746.

Gupta, A. K. (1984b), "Membrane reinforcement in concrete shells: a review." *Nuclear Engineering and Design*, **82**, 63–75.

Hand, Frank R., Pecknold, David A., and Schnobrich, William C. (1973), "Nonlinear layered analysis of RC plates and shells." *J. Struct. Div., ASCE,* **99**(7), 1491–1505.

Hughes, Thomas J. R., Ferencz, R. M., and Hallquist, J. O. (1987), "Large scale vectorized implicit calculations in solid mechanics on a Cray X–MP/48 utilizing EBE preconditioned conjugate gradients." *Comp. Methods Appl. Mech. Eng.*, **61**(2), 215–248.

Hutchinson, S., Hensel, E., Castillo, S., and Dalton, K. (1991), "The Finite element solution of elliptical systems on a data parallel computer." *Int. J. for Numer. Methods in Eng.*, **32**, 347–362.

Lambiotte, Jules J. (1975), The Solution of linear systems of equations on a vector computer. Ph.D. thesis, University of Virginia.

Lin, Cheng–Shung, and Scordelis, Alexander C. (1975), "Nonlinear analysis of RC shells of general form." *J. Struct. Div., ASCE,* **101**(3), 523–538.

Lou, Jenn–Ching, and Friedman, Morton B. (1991), "A Formulation of the finite element method by implicit decomposition." *Finite Elements in Analysis and Design,* **10**, 137–150.

Malone, J. G. (1988), "Automated mesh decomposition and concurrent finite element analysis for hypercube multiprocessor computers." *Comput. Methods Appl. Mech. Eng.*, **70**, 27–58.

Milford, R. V., and Schnobrich, W. C. (1984), "Nonlinear behavior of reinforced concrete cooling towers." Technical report, University of Illinois, Urbana–Champaign, Illinois 61801, May, *Structural Research Series, No. 514.*

Min, Chang Shik, and Gupta, Ajaya K. (1991), "Vector finite–element analysis using IBM 3090–600E VF." *Commun. in Applied Numer. Methods,* **7** (2), 155–164.

Min, Chang Shik, and Gupta, Ajaya K. (1992), "A Study of Inelastic behavior of reinforced concrete shells using supercomputers." Technical report, Department of Civil Engrg., North Carolina State Univ., Raleigh, *North Carolina 27695–7908,* March 1992.

Noor, Ahmed K., and Peters, Jeanne M. (1986), "Element stiffness computation on CDC Cyber 205 computer." *Commun. in Applied Numer. Methods,* **2**, 317–328.

Pawsey, Stuart F., and Clough, Ray W. (1971), "Improved numerical integration of thick shell finite elements." *Int. J. for Numer. Methods in Eng.*, **3**, 575–586.

Silvester, D. J. (1988), "Optimizing finite element matrix calculation using the general technique of element vectorization." *Parallel Computing,* **6**, 157–164.

Storaasli, O. O., Nguyen, D. T., and Agrawal, T. K. (1989), "Parallel–vector solution of large–scale structural analysis problems on supercomputers." *In 30th Structures, Structural Dynamics and Material Conference,* 859–867, Mobil, Alabama, April 1989, AIAA/ASME/ASCE/AHS/ASC.

Yagawa, G., Yoshioka, A., Yoshimura, S., and Soneda, N. (1993), "A parallel finite element method with a supercomputer network." *Computers and Structures,* **47** (3), 407–418.