Structural Engineering and Mechanics, Vol. 3, No. 2 (1995) 173-183
DOI: http://dx.doi.org/10.12989/sem.1995.3.2.173

Vector algorithm for layered 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 new vector algorithm is presented for computing the stiffness matrices of layered reinforced concrete shell elements. Each element stiffness matrix is represented in terms of three vector arrays of lengths 78, 96 and 36, respectively. One element stiffness matrix is calculated at a time without interruption in the vector calculations for the uncracked or cracked elements. It is shown that the present algorithm is 1.1 to 7.3 times more efficient then a previous algorithm developed by us on a Cray Y-MP supercomputer.

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

1. Introduction

We are presenting here a vector algorithm for calculating the stiffness matrix of a 4-node isoparametric element (Ahmad, *et al.* 1970) to be used to model reinforced concrete shells. In the previous paper (Min and Gupta 1994a), we considered the effect of inplane-membrane stresses only on the cracking of concrete and yielding of steel even though the element formulation includes the bending stiffness terms. The effect of both the bending and inplane stresses is considered here on the cracking of concrete and yielding of steel. There are other significant differences also between the previous and the present papers. To take advantage of the vector computing, in the previous paper all the arrays had a vector length of ne, the number of elements. For each iteration, the element stiffness matrices were calculated using the vector length ne and uncracked concrete properties. Scalar operations were performed to recalculate the stiffness matrices of the elements with cracked concrete. We felt that it was more economical to re-evaluate the stiffness matrices of the cracked elements than to sacrifice the uniform vector length ne throughout the stiffness matrix calculations. In the previous algorithm, since the effect of bending on the cracking of concrete and bending of steel was not considered, the entire element was treated as "one layer."

† Lecturer
‡ Professor of Civil Engineering and Director

To account for bending on the cracking of concrete and yielding of steel, a layering approach (Hand, *et al.* 1973, Lin and Scordelis 1975) is used here, in effect, increasing the number of elements by a factor equal to the number of layers. Doing so also increases the proportion of the number of cracked element layers averaged over the total number of iterations to be performed in any analysis. Therefore, the idea of recalculating stiffness matrices of the cracked elements becomes less attractive in the multi-layer element than in the previous single-layer element. An alternative vector algorithm for the layered reinforced concrete element is developed here, and its computational efficiency is compared against the previous algorithm for the single layer element.

2. Formulation of layered shell element stiffness matrix

As before (Min and Gupta 1994a), the element stiffness matrix $[\boldsymbol{k}]$ is represented in terms of $nn \times nn$ submatrices $[\boldsymbol{k}_{ij}]$ of the order 6×6, where $nn=4$, the number of nodes in an element. Further, a submatrix $[\boldsymbol{k}_{ij}]$ is separated into two matrices, $[\boldsymbol{k}_{p_{ij}}]$ for inplane stiffness, and $[\boldsymbol{k}_{s_{ij}}]$ for transverse stiffness.

In the layering approach, integration along the third isoparametric coordinate, ζ can be approximated by summation on the concrete layers, and on the steel layers. The isoparametric coordinate, ζ is defined perpendicular to the middle surface of the shell. Each layer is assumed to be in a state of plane stress and can have different material properties. The submatrix $[\boldsymbol{k}_{p_{ij}}]$ can be written as

$$[\boldsymbol{k}_{p_{ij}}]_{6\times6}=\sum_k\begin{bmatrix}[\boldsymbol{p}] & \zeta[\boldsymbol{p}]\\ \zeta[\boldsymbol{p}] & \zeta^2[\boldsymbol{p}]\end{bmatrix}_k,\quad [\boldsymbol{P}]_k=\begin{bmatrix}(D_1)_k[\boldsymbol{p}_1]+(D_2)_k[\boldsymbol{p}_2]+(D_3)_k[\boldsymbol{p}_3]\\ +(D_4)_k[\boldsymbol{p}_4]+(D_5)_k[\boldsymbol{p}_5]+(D_6)_k[\boldsymbol{p}_6]\end{bmatrix}, \tag{1}$$

where

$$[D_1\ D_2\ D_3\ D_4\ D_5\ D_6]_k=h_k[E_1\ E_2\ E_3\ E_4\ E_5\ E_6]_k \text{ for a concrete layer,}$$

$$\text{or}=(E_s)_k\left[\left(\frac{A_s}{b}\right)_{\bar{x}} 0\ 0\ \left(\frac{A_s}{b}\right)_{\bar{y}}\ 0\ 0\right] \text{ for a steel layer,} \tag{2}$$

in which h_k is the thickness of concrete layer k; E_1, E_2, E_3, E_4, E_5 and E_6 the terms of the constitutive matrix relating concrete stresses and strains; E_s Young's modulus of steel; and $(A_s, b)_{\bar{x}}$ and $(A_s, b)_{\bar{y}}$ are the total steel area for the layer and the width of an element in the $\bar{x}$ and $\bar{y}$ directions, respectively. For uncracked concrete layers, E_3 and E_5 are zero. Steel is assumed to be an elastic-perfectly plastic material. The six $\boldsymbol{p}$-submatrices are of the order 3×3 and can be expressed as before.

The submatrix for $[\boldsymbol{k}_{s_{ij}}]$ for transverse shear stiffness can be written as

$$[\boldsymbol{k}_{s_{ij}}]_{6\times6}=\sum_k\left\{(D_7)_k\begin{bmatrix}q_{a_{11}} & \frac{2}{h}q_{a_{12}}\\ \frac{2}{h}q_{a_{21}} & \left(\frac{2}{h}\right)^2 q_{a_{22}}\end{bmatrix}+(D_7)_k\begin{bmatrix}q_{b_{11}} & \frac{2}{h}q_{b_{12}}\\ \frac{2}{h}q_{b_{21}} & \left(\frac{2}{h}\right)^2 q_{b_{22}}\end{bmatrix}\right\}, \tag{3}$$

where $(D_7)_k=(hE_7)_k$, in which E_7 is the transverse shear modulus; h the total thickness of an element, $h=\sum_k h_k$. The submatrices $[\boldsymbol{q}_a]$ and $[\boldsymbol{q}_b]$ are defined as before.

3. Numerical integration

Appropriate selective numerical integration using the Gaussian quadrature formulation is performed in the calculation of various terms in $\boldsymbol{p}$ and $\boldsymbol{q}$-submatrices (Pawsey and Clough 1971, Gupta and Akbar 1984). Before cracking, the stiffness terms related to $\varepsilon_{\bar{x}}$ and $\varepsilon_{\bar{y}}$ are integrated using a 2×2 quadrature, those related to $\gamma_{\bar{x}\bar{y}}$ using a 1 point quadrature, and those related to $\gamma_{\bar{x}\bar{z}}$ and $\gamma_{\bar{y}\bar{z}}$ using 1×2 and 2×1 quadrature, respectively. After cracking, all the concrete stiffness terms related to $\varepsilon_{\bar{x}}$, $\varepsilon_{\bar{y}}$, $\gamma_{\bar{x}\bar{y}}$, $\gamma_{\bar{x}\bar{z}}$, $\gamma_{\bar{y}\bar{z}}$ are integrated using a 1 point quadrature, and the steel reinforcement stiffness terms (related to $\varepsilon_{\bar{x}}$ and $\varepsilon_{\bar{y}}$) using a 2×2 quadrature.

4. Transformation

The three global rotational degrees of freedom are transformed 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. After pre- and post-multiplying the stiffness submatrices given by Eqs. (1) and (3) with the transformation matrices, we get

$$[\bar{\boldsymbol{k}}_{ij}]_{5\times 5}=\sum_{r=1}^{6}\sum_{k}\left[(D_r)_k[\boldsymbol{P}^a_{r_{ij}}]+(\zeta D_r)_k[\boldsymbol{P}^b_{r_{ij}}]+(\zeta^2 D_r)_k[\boldsymbol{P}^c_{r_{ij}}]\right]$$
$$+\sum_{k}\left[(D_7)_k[\boldsymbol{P}^a_{7_{ij}}]+\frac{2}{h}(D_7)_k[\boldsymbol{P}^b_{7_{ij}}]+\frac{4}{h^2}(D_7)_k[\boldsymbol{P}^c_{7_{ij}}]\right], \tag{4}$$

in which

$$[\boldsymbol{P}^a_{r_{ij}}]=\begin{bmatrix}[\boldsymbol{P}_r] & [0]_{3\times 2}\\ [0]_{2\times 3} & [0]_{2\times 2}\end{bmatrix}_{ij},\quad [\boldsymbol{P}^b_{r_{ij}}]=\begin{bmatrix}[0]_{3\times 3} & [\boldsymbol{p}_r][\boldsymbol{T}_{22}]_j\\ [\boldsymbol{T}_{22}]^T_i[\boldsymbol{p}_r] & [0]_{2\times 2}\end{bmatrix}_{ij},$$
$$[\boldsymbol{P}^c_{r_{ij}}]=\begin{bmatrix}[0]_{3\times 3} & [0]_{3\times 2}\\ [0]_{2\times 3} & [\boldsymbol{T}_{22}]^T_i[\boldsymbol{p}_r][\boldsymbol{T}_{22}]_j\end{bmatrix}_{ij}, \tag{5}$$

and

$$[\boldsymbol{P}^a_{7_{ij}}]=\begin{bmatrix}[\boldsymbol{q}_{\boldsymbol{a}_{11}}+\boldsymbol{q}_{\boldsymbol{b}_{11}}] & [0]_{3\times 2}\\ [0]_{2\times 3} & [0]_{2\times 2}\end{bmatrix}_{ij},\quad [\boldsymbol{P}^b_{7_{ij}}]=\begin{bmatrix}[0]_{3\times 3} & [\boldsymbol{q}_{\boldsymbol{a}_{12}}+\boldsymbol{q}_{\boldsymbol{b}_{12}}][\boldsymbol{T}_{22}]_j\\ [\boldsymbol{T}_{22}]^T_i[\boldsymbol{q}_{\boldsymbol{a}_{21}}+\boldsymbol{q}_{\boldsymbol{b}_{21}}] & [0]_{2\times 2}\end{bmatrix}_{ij},$$
$$[\boldsymbol{P}^c_{7_{ij}}]=\begin{bmatrix}[0]_{3\times 3} & [0]_{3\times 2}\\ [0]_{2\times 3} & [\boldsymbol{T}_{22}]^T_i[\boldsymbol{q}_{\boldsymbol{a}_{22}}+\boldsymbol{q}_{\boldsymbol{b}_{22}}][\boldsymbol{T}_{22}]_j\end{bmatrix}_{ij}, \tag{6}$$

where $[\boldsymbol{T}_{22}]$ is the 3×2 transformation matrix relating the three global and the two local rotations (Min and Gupta 1994a). The submatrices $[\boldsymbol{P}^a_{r_{ij}}]$, $[\boldsymbol{P}^b_{r_{ij}}]$, $[\boldsymbol{P}^c_{r_{ij}}]$, $[\boldsymbol{P}^a_{7_{ij}}]$, $[\boldsymbol{P}^b_{7_{ij}}]$ and $[\boldsymbol{P}^c_{7_{ij}}]$ are independent of the layer properties. Therefore, the numerical integration to evaluate these submatrices needs to be performed only once for each element. Since the Gaussian quadratures are different before and after cracking, actually, two sets of the submatrices are precalculated and stored in the beginning of the analysis.

We can now write various $[\boldsymbol{P}_r]$, $r=1-7$ submatrices for all the $nn\times nn$ pair of nodes. For example, $[\boldsymbol{P}^a_r]$, can be expressed as

$$[\boldsymbol{P}^a_r]=\begin{bmatrix}[\boldsymbol{P}^a_{r_{11}}] & [\boldsymbol{P}^a_{r_{12}}] & [\boldsymbol{P}^a_{r_{13}}] & [\boldsymbol{P}^a_{r_{14}}]\\ & [\boldsymbol{P}^a_{r_{22}}] & [\boldsymbol{P}^a_{r_{23}}] & [\boldsymbol{P}^a_{r_{24}}]\\ & & [\boldsymbol{P}^a_{r_{33}}] & [\boldsymbol{P}^a_{r_{34}}]\\ \text{sym} & & & [\boldsymbol{P}^a_{r_{44}}]\end{bmatrix} \tag{7}$$

in which each $[P^a_{r_{ij}}]$ submatrix is 5×5. The four diagonal submatrices are symmetric and have 6 non-zero unique terms each. Each of the 6 off-diagonal submatrices has 9 non-zero terms. Thus, $[P^a_r]$ has a total of 78 upper triangular, non-zero terms. These terms are stored in a vector array which has a length of 78. Similarly, $[P^b_r]$ and $[P^c_r]$ have a total of 96 and 36 upper triangular non-zero unique terms, respectively, and these submatrices are also stored as vector arrays of appropriate lengths.

5. Vector algorithm

We calculate various $\boldsymbol{p}$ and $\boldsymbol{q}$ matrices using the ne-vector length as we did before (Min and Gupta 1994a). Table 1 presents an algorithm for stiffness matrix terms to be integrated using 2×2 quadrature points which results in the calculation of $\boldsymbol{p}_{1,2,4}$ matrices that are stored in the $\boldsymbol{BM}$ (ne, 27, 10) array. Algorithms for other quadratures (1×1, 1×2 and 2×1) are similar and are given in Min and Gupta (1994b). In a 1 point quadrature, $\boldsymbol{p}_{1-6}$ matrices are computed and stored in the $\boldsymbol{BN}$ (ne, 54, 10) array. For transverse shear stiffness, $\boldsymbol{q}_{a_{11,12,21,22}}$ matrices, integrated using a 1×2 quadrature, are stored in $\boldsymbol{BM}_1$ (ne, 6, 10), $\boldsymbol{BM}_2$ (ne, 9, 10), $\boldsymbol{BM}_3$ (ne, 9, 10) and $\boldsymbol{BM}_4$ (ne, 6, 10) arrays, respectively. The terms of $\boldsymbol{q}_{b_{11,12,21,22}}$ matrices integrated with 2×1 quadrature points are added to the appropriate $\boldsymbol{BM}_{1-4}$ arrays. Similarly, for a 1 point quadrature, transverse shear stiffness terms that are combinations of various $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$ matrices are calculated and stored in $\boldsymbol{BN}_1$ (ne, 6, 10), $\boldsymbol{BN}_2$ (ne, 9, 10), $\boldsymbol{BN}_3$ (ne, 9, 10) and $\boldsymbol{BN}_4$ (ne, 6, 10) arrays.

At this point, we rearrange the data in the form of $[P^a_r]$, $[P^b_r]$ and $[P^c_r]$ matrices as explained in the preceding section. Table 2 gives the steps for the $[P^a_r]$ matrix which is stored in the arrays $\boldsymbol{BO}_1$ (78, 5, ne) and $\boldsymbol{BQ}_1$ (78, 7, ne) for uncracked and cracked elements, respectively. Steps for $[P^b_r]$ and $[P^c_r]$ are similar (Min and Gupta 1994b). The $[P^b_r]$ matrix is stored in $\boldsymbol{BO}_2$ (96, 5, ne) and $\boldsymbol{BQ}_2$ (96, 7, ne) arrays and the $[P^c_r]$ matrix in $\boldsymbol{BO}_3$ (36, 5, ne) and $\boldsymbol{BQ}_3$ (36, 7, ne) arrays, respectively. In themselves, the steps carried out in creating $[P^a_r]$, $[P^b_r]$ and $[P^c_r]$ matrices appear to constitute unnecessary overhead that would reduce the computational efficiency. However, these steps (that need to be carried out once only, before iterations start) allow us to vectorize the stiffness matrix calculations presented in Table 3 leading to significant computational efficiency. Each element stiffness matrix has a vector length of 210. Calculations are performed in a sequence of three vectors that have lengths of 78, 96 and 36. The process does not change from uncracked to cracked elements or among layers. Stiffness matrices of the cracked elements need not be re-evaluated using scalar calculations as we did in the previous algorithm (Min and Gupta 1994a).

In the present algorithm, the storage of each element stiffness matrix in a vector of length 210 directly leads to assembly of the global stiffness matrix using indirect gather/scatter operations (Cray SR-0018 C) without having to rearrange the element stiffness matrices. In the previous algorithm, we had to rearrange the element stiffness matrices before the assembly. This rearrangement introduces a significant overhead because it is repeated for each iteration. We recall that a similar rearrangement is carried out in developing the $\boldsymbol{p}$ and $\boldsymbol{q}$ arrays in the present algorithm. However, this is done only once, before the iterations start.

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

Step	Description	Array sizes
1	Evaluate at each integration point, $N_{i,\xi}$, $N_{i,\eta}$	$2\times(nn, 4)$
2*	Compute the elements of the Jacobian matrices J, $x_{,\xi}$, $y_{,\xi}$, $x_{,\eta}$, $y_{,\eta}$	$(ne, 4)$
	Compute the direction cosines, $n=[n_{\bar{x}}\ n_{\bar{y}}\ n_{\bar{z}}]$ $n_{\bar{x}}=[\bar{x}_{,x}\ \bar{x}_{,y}\ \bar{x}_{,z}]$, $n_{\bar{y}}=[\bar{y}_{,x}\ \bar{y}_{,y}\ \bar{y}_{,z}]$, $n_{\bar{z}}=[\bar{z}_{,x}\ \bar{z}_{,y}\ \bar{z}_{,z}]$,	$(ne, 9, ni)$
3*	Evaluate the determinant of the Jacobian and its inverse, $\lvert J\rvert$, $U=1/\lvert J\rvert$	$2\times(ne)$
	Compute, $N_{i,\bar{x}}=U(N_{i,\xi}y_{,\eta}-N_{i,\eta}y_{,\xi})$, $N_{i,\bar{y}}=U(-N_{i,\xi}x_{,\eta}+N_{i,\eta}x_{,\xi})$	$2\times(ne, nn, ni)$
4*	Compute, $\bar{W}=\lvert J\rvert$ (A weight coefficient of unity is implied.)	(ne, ni)
5*	Calculate; for $i=1, nn$; $j=i, nn$ $\begin{bmatrix} DN_1 & DN_2 \\ DN_3 & DN_4 \end{bmatrix}=\begin{bmatrix} N_{i,\bar{x}}N_{j,\bar{x}} & N_{i,\bar{x}}N_{j,\bar{y}} \\ N_{i,\bar{y}}N_{j,\bar{x}} & N_{i,\bar{y}}N_{j,\bar{y}} \end{bmatrix}$, $DN_5=DN_2+DN_3$	$(ne, 5nc)$
6*	Compute the 21 upper triangular terms of the 6×6 matrix $\bar{W}[n_{\bar{x}}\ n_{\bar{y}}]^T[n_{\bar{x}}\ n_{\bar{y}}]$ and store in the array TT.	$(ne, 21, ni)$
7*	Three 3×3 submatrices, p_1, p_2, p_4 are to be calculated for 10 upper triangular pairs of nodes shown below: $\begin{bmatrix} (1,1) & (1,2) & (1,3) & (1,4) \\ & (2,2) & (2,3) & (2,4) \\ & & (3,3) & (3,4) \\ \text{Symmetric} & & & (4,4) \end{bmatrix}$	
	Each of the submatrices (1,1), (2,2), (3,3) and (4,4) is symmetric and has six unique terms. These terms are numbered for sets of p_1, p_2 and p_4 as follows: $\begin{bmatrix} (1,2,3) & (4,5,6) & (7,8,9) \\ & (10,11,12) & (13,14,15) \\ \text{Symmetric} & & (16,17,18) \end{bmatrix}$	
	Calculate p_1, p_2, p_4 submatrices corresponding to the above terms and store in the BM array	$(ne, 27, nc)$**
	$BM(1,4,7,10,13,16)=BM(1,4,7,10,13,16)+DN_1\ TT(1,2,3,7,8,12)$ $BM(2,11,17)=BM(2,11,17)+DN_5\ TT(4,10,15)$ $BM(3,6,9,12,15,18)=BM(3,6,9,12,15,18)+DN_4\ TT(16,17,18,19,20,21)$ $BM(5,8,14)=BM(5,8,14)+DN_3\ TT(9,13,14)+DN_2\ TT(5,6,11)$	
	Each of the off-diagonal submatrices, (1,2), (1,3), (1,4), (2,3), (2,4) and (3,4) has 9 unique terms. Again, these terms are numbered for sets of p_1, p_2 and p_4 as follows: $\begin{bmatrix} (1,2,3) & (4,5,6) & (7,8,9) \\ (10,11,12) & (13,14,15) & (16,17,18) \\ (19,20,21) & (22,23,24) & (25,26,27) \end{bmatrix}$	
	Calculate p_1, p_2, p_4 submatrices corresponding to the above terms and store in the BM array	
	$BM(1,4,7,10,13,16,19,22,25)=\begin{cases} BM(1,4,7,10,13,16,19,22,25) \\ +DN_1\ TT(1,2,3,2,7,8,3,8,12) \end{cases}$	
	$BM(2,14,26)=BM(2,14,26)+DN_5\ TT(4,10,15)$	
	$BM(3,6,9,12,15,18,21,24,27)=\begin{cases} BM(3,6,9,12,15,18,21,24,27) \\ +DN_4\ TT(16,17,18,17,19,20,18,20,21) \end{cases}$	

$$BM(5,8,11,17,20,23)=\begin{cases} BM(5,8,11,17,20,23) \\ +DN_3\ TT(9,13,5,14,6,11) \\ +DN_2\ TT(5,6,9,11,13,14) \end{cases} \qquad (ne,\ 27,\ nc)^{**}$$

BM-array is initialized before above operations.

---

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 $nn \times nn$ matrix $= nn(nn+1)/2 = 10$

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

**The dimension of 27 is based on $3\times3=9$ unique terms in the 3 submatrices (p_1, p_2, p_4). For symmetric matrices we have 6 unique terms per submatrix and require space for only 18 terms. However, the dimension of 27 is used both for diagonal and off-diagonal submatrices for convenience and efficiency.

Table 2 Algorithm for generation of $[P_r^a]$, Eq. (7)

Step	Description	Array sizes
	The $[P_r^a]$ matrix is stored in BO_1 and BQ_1-arrays for uncracked and cracked elements, respectively, in the following order to form a vector (vector length $= np = 78$);	

$$\left(\begin{array}{ccccc|ccccc|ccccc|ccccc}
1 & 2 & 3 & 0 & 0 & 7 & 8 & 9 & 0 & 0 & 16 & 17 & 18 & 0 & 0 & 25 & 26 & 27 & 0 & 0 \\
 & 4 & 5 & 0 & 0 & 10 & 11 & 12 & 0 & 0 & 19 & 20 & 21 & 0 & 0 & 28 & 29 & 30 & 0 & 0 \\
 & & 6 & 0 & 0 & 13 & 14 & 15 & 0 & 0 & 22 & 23 & 24 & 0 & 0 & 31 & 32 & 33 & 0 & 0 \\
 & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
 & & & & & 34 & 35 & 36 & 0 & 0 & 40 & 41 & 42 & 0 & 0 & 49 & 50 & 51 & 0 & 0 \\
 & & & & & & 37 & 38 & 0 & 0 & 43 & 44 & 45 & 0 & 0 & 52 & 53 & 54 & 0 & 0 \\
 & & & & & & & 39 & 0 & 0 & 46 & 47 & 48 & 0 & 0 & 55 & 56 & 57 & 0 & 0 \\
 & & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
 & & & & & & & & & & 58 & 59 & 60 & 0 & 0 & 64 & 65 & 66 & 0 & 0 \\
 & & & & & & & & & & & 61 & 62 & 0 & 0 & 67 & 68 & 69 & 0 & 0 \\
 & & & & & & & & & & & & 63 & 0 & 0 & 70 & 71 & 72 & 0 & 0 \\
 & & & & & & & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
 & & & & & & & & & & & & & & & 73 & 74 & 75 & 0 & 0 \\
 & & & & & & & & & & & & & & & & 76 & 77 & 0 & 0 \\
 & & & & & & & & & & & & & & & & & 78 & 0 & 0 \\
 & & & & & & & & & & & & & & & & & & 0 & 0 \\
\text{summetric} & & & & & & & & & & & & & & & & & & & 0
\end{array}\right)$$

Step	Description	Array sizes
	The following steps are performed in three do loops in the following order (outermost-innermost): $i=1$, nn; $j=\mathrm{i}$, nn; $ie=1$, ne	
1	When $i=j$, calculate the 6 upper triangular terms for the four diagonal submatrices,	(np, 5, ne)
	$BO_1(1)=BM(1,4,7,10,13,16)$, $BO_1(2)=BM(2,5,8,11,14,17)$	
	$BO_1(3)=BM(3,6,9,12,15,18)$, $BO_1(4)=BN(6,12,18,24,30,36)$	
	$BO_1(5)=BM_1(1,2,3,4,5,6)$	
	When $i\neq j$, calculate the 9 terms in each of the six off- diagonal submatrices,	
	$BO_1(1)=BM(1,4,7,10,13,16,19,22,25)$	(np, 5, ne)

$\boldsymbol{BO}_1(2) = \boldsymbol{BM}(2,5,8,11,14,17,20,23,26)$
$\boldsymbol{BO}_1(3) = \boldsymbol{BM}(3,6,9,12,15,18,21,24,27)$
$\boldsymbol{BO}_1(4) = \boldsymbol{BN}(6,12,18,24\,30,36,42,48,54)$
$\boldsymbol{BO}_1(5) = \boldsymbol{BM}_1(1,2,3,2,4,5,3,5,6)$

2 When $i=j$, calculate the 6 upper triangular terms for the four diagonal submatrices,

$\boldsymbol{BQ}_1(1) = \boldsymbol{BN}(1,7,13,19,25,31)$,	$\boldsymbol{BQ}_1(2) = \boldsymbol{BN}(2,8,14,20,26,32)$
$\boldsymbol{BQ}_1(3) = \boldsymbol{BN}(3,9,15,21,27,33)$	$\boldsymbol{BQ}_1(4) = \boldsymbol{BN}(4,10,16,22,28,34)$
$\boldsymbol{BQ}_1(5) = \boldsymbol{BN}(5,11,17,23,29,35)$	$\boldsymbol{BQ}_1(6) = \boldsymbol{BN}(6,12,18,24,30,36)$
$\boldsymbol{BQ}_1(7) = \boldsymbol{BN}_1(1,2,3,4,5,6)$	

when $i \neq j$, calculate the 9 terms in each of the six off-diagonal submatrices,

$\boldsymbol{BQ}_1(1) = \boldsymbol{BN}(1,7,13,19,25,31,37,43,49)$ $(np, 7, ne)$
$\boldsymbol{BQ}_1(2) = \boldsymbol{BN}(2,8,14,20,26,32,38,44,50)$
$\boldsymbol{BQ}_1(3) = \boldsymbol{BN}(3,9,15,21,27,33,39,45,51)$
$\boldsymbol{BQ}_1(4) = \boldsymbol{BN}(4,10,16,22,28,34,40,46,52)$
$\boldsymbol{BQ}_1(5) = \boldsymbol{BN}(5,11,17,23,29,35,41,47,53)$
$\boldsymbol{BQ}_1(6) = \boldsymbol{BN}(6,12,18,24,30,36,42,48,54)$
$\boldsymbol{BQ}_1(7) = \boldsymbol{BN}_1(1,2,3,2,4,5,3,5,6)$

ne = number of elements, nn = number of nodes in an element = 4, nc = number of the upper triangular coefficients in an $nn \times nn$ matrix = $nn(nn+1)/2 = 10$; np = number of coefficients related with submatrix $[\boldsymbol{P}_i^e]$, 4 diagonal blocks (each 6 coefficients) and 6 off-diagonal blocks (each 9 coefficients) = $4\times6+6\times9=78$

Table 3 Algorithm for calculating element stiffness matrix

Step	Description	Array sizes

Each step is in a do loop, $i=1$, ne. Perform calculations for each layer accounting for its uncracked or cracked state identified by an index array. Element stiffness matrix for each element is stored as a vector of length 210 by sequentially stacking three vectors of 78, 96 and 36 (1-78, 79-174 and 175-210).

Steel contribution

Step 1 is carried out only once and is not repeated during iterations.

1 Compute stiffness contribution of each steel layer in three do loops:
(i) $np = 1,78$

$$[\boldsymbol{K}_1] = [\boldsymbol{K}_1] + \left\{E_S\left(\frac{A_S}{b}\right)_x\right\}_k \boldsymbol{BO}_1(1) + \left\{E_S\left(\frac{A_S}{b}\right)_y\right\}_k \boldsymbol{BO}_1(3) \qquad (210, ne)$$

(ii) $nq = 79,174$

$$[\boldsymbol{K}_1] = [\boldsymbol{K}_1] + \left\{\zeta E_S\left(\frac{A_S}{b}\right)_x\right\}_k \boldsymbol{BO}_2(1) + \left\{\zeta E_S\left(\frac{A_S}{b}\right)_y\right\}_k \boldsymbol{BO}_2(3) \qquad (210, ne)$$

(iii) $nr = 175,210$

$$[\boldsymbol{K}_1] = [\boldsymbol{K}_1] + \left\{\zeta^2 E_S\left(\frac{A_S}{b}\right)_x\right\}_k \boldsymbol{BO}_3(1) + \left\{\zeta^2 E_S\left(\frac{A_S}{b}\right)_y\right\}_k \boldsymbol{BO}_3(3) \qquad (210, ne)$$

Matrix $[\boldsymbol{K}_1]$ is initialized before above operations.
The following are repeated for each iteration.
2 Initialize stiffness matrix $[\boldsymbol{K}]$ by; $[\boldsymbol{K}_1]$; $[\boldsymbol{K}] = [\boldsymbol{K}_1]$ $(210, ne)$

Concrete contribution

3 Add contribution of the submatrix $[P_r^u]$ for each layer k in a do loop, $np=1,78$
For uncracked elements:

$$[K]=[K]+(D_1)_k BO_1(1)+(D_2)_k BO_1(2)+(D_4)_k BO_1(3)$$
$$+(D_6)_k BO_1(4)+(D_7)_k BO_1(5) \qquad (210,\ ne)$$

For cracked elements:

$$[K]=[K]+(D_1)_k BQ_1(1)+(D_2)_k BQ_1(2)+(D_3)_k BQ_1(3)$$
$$+(D_4)_k BQ_1(4)+(D_5)_k BQ_1(5)+(D_6)_k BQ_1(6)$$
$$+(D_7)_k BQ_1(7) \qquad (210,\ ne)$$

4 Add contribution of the submatrix $[P_r^b]$ for each layer k in a do loop, $nq=79,174$
For uncracked elements:

$$[K]=[K]+(\zeta D_1)_k BO_2(1)+(\zeta D_2)_k BO_2(2)+(\zeta D_4)_k BO_2(3)$$
$$+(\zeta D_6)_k BO_2(4)+(D_7)_k BO_2(5) \qquad (210,\ ne)$$

For cracked elements:

$$[K]=[K]+(\zeta D_1)_k BQ_2(1)+(\zeta D_2)_k BQ_2(2)+(\zeta D_3)_k BQ_2(3)$$
$$+(\zeta D_4)_k BQ_2(4)+(\zeta D_5)_k BQ_2(5)+(\zeta D_6)_k BQ_2(6)$$
$$+(D_7)_k BQ_2(7) \qquad (210,\ ne)$$

5 Add contribution of the submatrix $[P_r^c]$ for each layer k in a do loop, $nr=175,210$
For uncracked elements:

$$[K]=[K]+(\zeta^2 D_1)_k BO_3(1)+(\zeta^2 D_2)_k BO_3(2)+(\zeta^2 D_4)_k BO_3(3)$$
$$+(\zeta^2 D_6)_k BO_3(4)+(D_7)_k BO_3(5) \qquad (210,\ ne)$$

For cracked elements:

$$[K]=[K]+(\zeta^2 D_1)_k BQ_3(1)+(\zeta^2 D_2)_k BQ_3(2)+(\zeta^2 D_3)_k BQ_3(3)$$
$$+(\zeta^2 D_4)_k BQ_3(4)+(\zeta^2 D_5)_k BQ_3(5)+(\zeta^2 D_6)_k BQ_3(6)$$
$$+(D_7)_k BQ_3(7) \qquad (210,\ ne)$$

---

ne=number of elements, np=number of coefficients related with submatrix $[P_r^u]$, 4 diagonal blocks (each 6 coefficients) and 6 off-diagonal blocks (each 9 coefficients)$=4\times6+6\times9=78$; nq=number of coefficients related with submatrix, $[P_r^b]$, 4 diagonal blocks (each 6 coefficients) and 6 off-diagonal blocks (each 2×6 coefficients)$=4\times6+6\times12=96$; nr=number of coefficients related with submatrix, $[P_r^c]$, 4 diagonal blocks (each 3 coefficients) and 6 off-diagonal blocks (each 4 coefficients)$=4\times3+6\times4=36$

6. Evaluation of efficiency of the algorithm

Relative efficiencies of the two algorithms, presented here and in the previous paper, are evaluated with the help of two models of a hyperbolic paraboloid saddle shell and three models of a hyperbolic cooling tower. Various parameters of the five models are summarized in Table 4. The shell element formulated in conjunction with the algorithm presented in this paper has ten concrete layers both for the saddle shell and the cooling tower models, one steel layer for

Table 4 Parameters of the models

	Saddle shell		Cooling tower		
Model	S16	S32	C12	C24	C36
Number of elements	89	305	144	432	1,296
Number of nodes	133	389	169	475	1,369

Table 5 Comparison of the efficiencies of the present and the previous Min-Gupta algorithms for element stiffness matrix calculation

Model	Previous Min-Gupta vector algorithm			Present vector algorithm			Efficiency ratio †
	Number of Iterations	CPU time*		Number of Iterations	CPU time*		
		Total(sec)	Per iteration-element-layer(μ sec)		Total(sec)	Per iteration-element-layer(μ sec)	
S16	265	2.5	106.0	1293	16.6	14.4	7.3
S32	493	6.1	40.6	1826	79.4	14.3	2.8
C12	947	1.7	12.5	744	12.8	11.9	1.1
C24	436	9.6	51.0	669	32.6	11.3	4.5
C36	993	70.2	54.5	489	67.8	10.7	5.1

*Time was measured using CPU timer SECOND function of the Cray Math Library (Cray SR-2081 6.0)
† Efficiency ratio = ratio of the CPU time taken in the previous Min-Gupta algorithm divided by that in the present algorithm for computing the element stiffness matrix per iteration-element-layer.

the saddle shell models, and four steel layers for the cooling tower models. Both the finite element computer programs are executed on a Cray Y-MP supercomputer. The Cray CFT77 compiler with the highest option, "full" is used. This option activates automatic vectorization with full optimization (Cray SR-0018 C).

The total CPU time used in the calculation of the element stiffness matrices in each of the five models using the two algorithms is given in Table 5. The number of iterations given in the table in each case is the total of the number of iterations of all the steps up to the load-displacement level roughly corresponding to what we considered to be the ultimate state. To measure the computational efficiency of the two algorithms, we have determined the CPU time used for calculating the stiffness matrix of one layer of each element in an average iteration. This CPU time, per iteration-element-layer is equal to the total time used in the calculation of stiffness matrices in an analysis divided by the numbers of iterations, elements and layers. The element used in the previous algorithm has only one layer, and that in the present algorithm has ten concrete layers and a variable number of steel layers for the two type of shells analyzed. The stiffness contribution of the steel layers is calculated only once for the elastic case and is not repeated for each iteration to avoid numerical instability after yielding of the steel. The effect of steel yielding is accounted for by applying the appropriately calculated residual forces in each iteration. The stiffness contribution of the ten concrete layers is recalculated in each iteration. Therefore, the effective value of the number of layers used to determine the CPU time per layer for the present algorithm is taken to be equal to ten, the number of concrete layers.

The CPU time per iteration-element-layer is quite consistent for the present algorithm: 10.7-11.9 μsec for the three cooling tower models, and 14.3 and 14.4 μsec for the two saddle shell models. The later may be higher because, proportionately, more elements crack in the saddle shell than in the cooling tower. As shown in Table 3, seven material constants are used in the calculation of a cracked element stiffness matrix as opposed to only five for the uncracked element. The CPU time per iteration-element-layer increases monotonically for the three hyperbolic cooling tower models, 12.5-54.5 μ sec, when the previous algorithm is used. The monotonic increase can be explained on the basis that a more refined finite element mesh leads to a proportionately higher number of cracked elements, which in conjunction with the previous algorithm (in which the cracked element stiffness matrices are recalculated using a scalar algorithm) would cause an increase in the CPU time. The CPU time-values for the calculation of the stiffness matrices for the two saddle shell models using the previous algorithm, however, defies the above trend, and we are not able to explain it. It is possible that the CPU time-values are incorrectly recorded for one or both the models. In any case, the relative efficiency of the present algorithm over the previous algorithm is clearly indicated by the efficiency ratios given in the last column of Table 5 that vary from 1.1 to 5.1 for the three cooling tower models, and are 7.3 and 2.8 for the two saddle shell models. At least in the cooling tower case, the higher the number of elements, the higher is the efficiency ratio.

7. Conclusions

A new vector algorithm is presented for calculating the stiffness matrices of the layered reinforced concrete shell elements that are uncracked and cracked. One element stiffness matrix is calculated at a time with three vector arrays of 78, 96 and 36 lengths. No scalar recalculation of the stiffness matrices of the cracked elements is required as was the case in a previous algorithm we developed (1994a). The efficiency of the present vector algorithm is investigated on a Cray Y-MP supercomputer. The new algorithm is 1.1 to 7.3 times more efficient than our previous algorithm.

Acknowledgements

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

References

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 Kumar (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.

Gupta, Ajaya Kumar and Akbar, Habibollah (1984), "Cracking in reinforced concrete analysis," *J. Struct. Engrg., ASCE*, **110**(8), 1735-1746.

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.

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

Min, Chang Shik and Gupta, Ajaya Kumar (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 Kumar (1992), "A Study of inelastic behavior of reinforced concrete shells using supercomputers," Department of Civil Engrg., North Carolina State University, Raleigh, North Carolina 27695-7908, March. Reinforced Concrete Shell Research Report.

Min, Chang Shik and Gupta, Ajaya Kumar (1994a), "Vector algorithm for reinforced concrete shell element stiffness Matrix," Structural Engineering and Mechanics, An International Journal, **2**(2), 125-139.

Min, Chang Shik and Gupta, Ajaya Kumar (1994b), "Vector algorithm for layered reinforced concrete shell element stiffness matrix," Center for Nuclear Power Plant Structures, Equipment and Piping, North Carolina State University, Raleigh, North Carolina 27695-7908, April, *Report.*

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.