Structural Engineering and Mechanics, Vol. 4, No. 2 (1996) 139-148
DOI: http://dx.doi.org/10.12989/sem.1996.4.2.139

Inelastic vector finite element analysis of RC shells

Chang-Shik Min†

Department of Ocean and Civil Engineering, Cheju National University, 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. Vector algorithms and the relative importance of the four basic modules (computation of element stiffness matrices, assembly of the global stiffness matrix, solution of the system of linear simultaneous equations, and calculation of stresses and strains) of a finite element computer program for inelastic analysis of reinforced concrete shells are presented. Performance of the vector program is compared with a scalar program. For a cooling tower problem, the speedup factor from the scalar to the vector program is 34 for the element stiffness matrices calculation, 25.3 for the assembly of global stiffness matrix, 27.5 for the equation solver, and 37.8 for stresses, strains and nodal forces computations on a Cray Y-MP. The overall speedup factor is 30.9.

When the equation solver alone is vectorized, which is computationally the most intensive part of a finite element program, a speedup factor of only 1.9 is achieved. When the rest of the program is also vectorized, a large additional speedup factor of 15.9 is attained. Therefore, it is very important that all the modules in a nonlinear program are vectorized to gain the full potential of the supercomputers. The vector finite element computer program for inelastic analysis of RC shells with layered elements developed in the present study enabled us to perform mesh convergence studies. The vector program can be used for studying the ultimate behavior of RC shells and used as a design tool.

Key words: inelastic; inelasticity; nonlinear; nonlinearity; vector computer; vector programming; vector algorithm; finite element method; finite element analysis; reinforced concrete shells

1. Introduction

Inelastic analysis of a reinforced concrete (RC) shell needs to be performed to study its ultimate behavior and to assure that the design methods give a shell that is safe. Finite element methods to perform such analysis have been developed and the inelastic behavior of shells studied by many investigators in the past (Hand *et al.* 1973, Lin and Scordelis 1975, Mang *et al.* 1983, Milford and Schnobrich 1984, Akbar and Gupta 1985, Min and Gupta 1992, Mahmoud and Gupta 1993). As would be expected there are some differences in the modeling and constitutive assumptions made in the computer programs developed by these investigators. For example, Akbar and Gupta (1985), Milford and Schnobrich (1984), Min and Gupta (1992) and Mahmoud and Gupta (1993) allow the crack directions to change according to the principal strain directions

† Assistant Professor
‡ Professor

as the loading is progressively increased, and others don't. A common limitation in all the past research is that very little, if any, finite element mesh convergence studies have been peformed. Inelastic analyses are computationally intensive, and performance of such studies using the conventional mainframe computers is practically impossible. For example, analysis of a hyperbolic cooling tower by Gupta and Maestrini (1986) using a 12×12 (144 elements) mesh of 4-node superparametric elements took around 10 hours of CPU time on an IBM 3081, and that of a nuclear containment vessel by Akbar and Gupta (1986) using a 432 element mesh required roughly 40 hours on the same machine.

The advent of supercomputers in recent years has provided an opportunity to study problems that could't be earlier. It is often possible to migrate a scalar program to a supercomputer with very few changes. Even scalar operations can be performed much faster in a supercomputer than in a conventional mainframe both in terms of the clock and the central processing unit (CPU) time. We solved an inelastic shell problem that ran on a Cray Y-MP using one processor around 30 times faster than on an IBM 3081 in terms of the CPU time. Even though the CPU time on the two machines cannot be directly compared, the factor of 30 does give an idea of their relative efficiency. A much higher level of efficiency can be achieved by writing a program that would use a specific supercomputer's vector and parallel processing capabilities to their maximum possible potential (Lambiotte 1975, Dongarra *et al.* 1984a, 1984b and 1992, Noor and Peters 1986, Silvester 1988, Levesque and Williamson 1989, Storaasli *et al.* 1989, Hutchinson *et al.* 1991, Min and Gupta 1991, 1994a and 1994b, Golub and Ortega 1993, Yagawa *et al.* 1993, Agrawal *et al.* 1994, Barragy *et al.* 1994).

In the present study, we will limit ourselves to vector processing only. Other than the I/O (input and output) and some other bookkeeping functions, a finite element program consists of four basic modules, namely, computation of element stiffness matrices, assembly of the global stiffiness matrix, solution of the system of linear simultaneous equations, and calculation of stresses and strains. Out of the four basic modules, the equation solver takes the most computational effort in the analysis; consequently it needs to be vectorized first. In case of the linear analysis of a plane stress problem using the 8-node isoparametric element, we found that a speedup factor of 5 to 50 was achieved by appropriately vectorizing the equation solver alone (Min and Gupta 1991). Vectorizing the rest of the program led to an additional speedup of 10 to 40 percent.

Vector algorithms and the relative importance of the four modules of a finite element computer program for inelastic analysis of reinforced concrete shells are presented here. Results from two computer programs are presented. The first program was developed by Akbar and Gupta (1985). It consists of 4-node superparametric shell elements. The elements do not account for the effect of bending on the cracking of concrete and the yielding of steel. It is a scalar program, originally developed for an IBM 3081, and was migrated to the North Carolina Supercomputing Center's (NCSC) Cray Y-MP with minor changes. The second program is a vectorized program that consists of the same 4-node superparametric shell elements as in the original scalar program, except that the vectorized program's element is discretized into ten layers allow the element to account for the effect of bending on cracking and yielding.

2. Element stiffness matrix

A vector algorithm for evaluating the element stiffness matrix of a layered 4-node superparametric shell element is presented elsewhere (Min and Gupta 1994b, 1995). In this algorithm, the element stiffness matrix is grouped according to three constitutive coefficients. These groups have lengths of 78, 96 and 36, respectively, for stiffness coefficients in the upper triangular arrays. The same computation is performed with three vector arrays of 78, 96 and 36 lengths for uncracked and cracked elements and for all the layers, calculating one element stiffness matrix at a time. The vector length of the complete upper triangular element stiffness matrix is 210 ($=78+96+36$). In an earlier vector algorithm for evaluating the stiffness matrix of the unlayered element, a vector length equal to the number of elements (ne) was used (Min and Gupta 1994a). The stiffness matrices of the cracked elements were re-evaluated using scalar computations. The 210-vector length algorithm is more efficient than the ne-vector length algorithm because the stiffness matrices of the cracked elements are not recalculated in the former.

The peformance of the new vector algorithm for element stiffness matrix calculation is compared with the scalar algorithm in the Akbar and Gupta (1985) computer program. Because the Akbar-Gupta program modeled the shell as a single-layer element, the CPU time obtained to calculate the element stiffness matrix in the scalar program was multiplied by a factor of 10 for comparing it with that of the present 10-layer element stiffness matrix in the vector program. The scalar and vector programs are compiled with 'novector' and 'full'-options for the Cray CFT77 compiler, respectively. The novector-option activated only vectorization and scalar optimization (Cray SR-0018 C). The full-option, on the other hand, attempts all optimization and vectorization. The speedup with the full-option is usually greater than that with the novector-option. The scalar program is implemented successfully with the novector-option, but it failed with the full-option. For a cooling tower problem, C24 (see Table 3) the scalar program needs an average of 1672 ms (milliseconds) of CPU time for each iteration to compute element stiffness matrices, and the vector program requires only 49 ms when both the problems were run up to the respective ultimate loads. The speedup factor for scalar-to-vector operations is 34 for this case. The 34-speedup factor is conservative because we have not accounted for the computational effort for adding the stiffness of the ten layers in the single layered element, and because the single layered model (that does not account for the effect of bending on concrete cracking) is likely to have smaller crack density than would the layered model used in the vector program.

3. Assembly of the global stiffness matrix

The Min and Gupta (1991) algorithm for the assembly of the global stiffness matrix on an IBM 3090 supercomputer is implemented with a few modifications. On a Cray Y-MP, vectorized-indirectly-addressed-gather/scatter operations are performed using a compressed index array and a gather/scatter hardware, thus greatly improving the computational efficiency. The Cray CFT77 compiler assumes that operations containing array elements with subscripts may involve recurrences. Therefore, we need to check dependencies of element stiffness coefficients in each element. If no dependence exists, we can proceed to perform vector operations using a compiler directive: IVDEP (Cray SR-0018 C). The directive should appear immediately before the indirect gather/scatter loop. Table 1 shows a FORTRAN code for the assembly operation. The arrays K and k represent the global and the element stiffness matrices, respectively, and index (i, j) is an index array defined for connectivity between global degrees on freedom and local element degrees of freedom. When a particular degree of freedom is constrained, the corresponding index value

Table 1 FORTRAN code for the assembly of the global stiffness matrix.

```
          DO 100 j=1, ne  ! a loop for the number of elements.
C         Check the element has any dependencies,
          IF (no dependency) THEN  ! vector operation;
CDIR$     IVDEP  ! Compiler directive
          DO 10 i=1, 210
            K(index(i, j))=K(index(i, j))+k(i, j)
10        CONTINUE
          ELSE  ! yes-scalar operation;
          DO 20 i =1, 210
            K(index(i, j))=K(index(i, j))+k(i, j)
20        CONTINUE
          ENDIF
100       CONTINUE
```

is set to zero. A dummy space in the global stiffness array is provided for index=0 to discard the extra stiffness coefficients.

The creation of the index array is straightforward. In a nonlinear analysis this array will have to be created only once whereas the global stiffness assembly is performed repeatedly. Therefore, generation of the index array can be considered a small overhead. As before, the C24 cooling tower problem is used to compare the performance of the scalar and the vector programs. The Akbar-Gupta scalar program consumes an average of 139 ms of CPU time for the assembly of the global stiffness matrix in an iteration and the vector program only 5.5 ms. The vector-scalar speedup factor is 25.3 for this problem.

4. Solution of linear systems of equations

Solution of the system of linear simultaneous equations is computationally the most intensive part of a finite element program. Accordingly, it has been a topic of vigorous research activity (Dongarra *et al.* 1984a and 1992, Lambiotte 1975, Ortega 1988, Storaasli 1989). We (Min and Gupta 1991) studied and implemented several equation solving algorithms and also reviewed performance of the DPBF and DPBS routines from the ESSL library (IBM 1987) on an IBM 3090 supercomputer. We found that the DPBF and DPBS routines were most efficient. We performed a similar study on the Cray Y-MP machine. We developed our own routines based on the *ijk* and *kji* algorithms (Dongarra *et al* 1984a, Ortega 1988, Min and Gupta 1992) and used two pairs of routines from the Cray UNICOS Math and Scientific Library (Cray SR-2081): SPBFA and SPBSL from LINPACK and SPBTRF and SPBTRS from LAPACK.

Actual CPU time for solving equations for three problems using four vector equation solvers is given in Table 2. Two runs were made for each problem using the LAPACK routines, using four processors and one processor. Various parameters for the problem C24 are given in Table 3. Finite element grids of the problems SM16 and SM32 are identical to those of S16 and S32, Table 3. However, most of the elements in the former are membrane type with only three degrees of freedom per node. All the problems in Table 3 have the element that includes bending with five degrees of freedom per node. SPBFA and SPBTRF perform the Cholesky factorization of a symmetric positive definite band matrix. SPBSL and SPBTRS do the forward and backward substitution. SPBFA and SPBSL routines were developed by Dongarra *et al.* (1979), and the routines were ported to the Cray and optimized. The LAPACK (Linear Algebra Package) which has the SPBTRF and SPBTRS routines was developed by various institutions such as University of Tennessee, Oak Ridge National Lab., Argonne National Lab., Courant Institute, NAG Ltd., and Rice University. The LAPACK codes have been carefully restructured for supercomputers to reuse as much data as possible in order to reduce the cost of data movement.

Based on Table 2, we can rank the four vector equation solvers from the best to the worst in the following order: LAPACK (one processor), LAPACK (four processors), *kji*, *ijk*, and LINPACK. We used the LAPACK routines with one processor. LAPACK routines with four processors take less wallclock time than that with one processor. Therefore, it may be desirable to use all the four processors when the wallclock time is of greater concern that the CPU time. In case of the three problems used for Table 2, SM16, SM32 and C24, the ratio of the wallclock time between the runs using four processors and one processor is 2.0, 1.6 and 1.2, respectively (not shown in Table 2). Table 2 also gives the Cray Y-MP CPU time consumed by the scalar equation solver in the Akbar-Gupta program. The speedup factor from scalar to the best vector solver (LAPACK, one processor) ranges from 16.9 to 27.5.

5. Stresses, strains and nodal forces

The best way to vectorize this module would be to use a vector length equal to the number of elements, ne. However, parts of the calculations are different for the uncracked and the cracked elements. For those parts, the only way to vectorize would be to perform one set of vector calculations assuming that all the elements are uncracked and re-perform another set of scalar calculations for the cracked elements. Rather than repeating the calculations, we used scalar calculations in the parts of the routine where the straight vectorization is not possible. In summary, we used a vector length of ne for the parts of the routine requiring the same type of calculations for the uncracked and the cracked elements and performed scalar calculations otherwise.

For the cooling tower problem, C24, the Akbar-Gupta scalar program needs an average of

Table 2. Comparison of the CPU time (seconds) for the equation solver on a Cray Y-MP

Model	Scalar	Vector equation solvers				
	(1) Akbar-Gupta	(2) *ijk*	(3) *kji*	(4) LINPACK	LAPACK (SPBTRF/SPBTRS)	
				(SPBFA/ SPBSL)	(5) 4 processors	(6) 1 processor
SM16	54.2	17.1	12.0	23.2	6.4	3.2
		(3.2)*	(4.5)	(2.3)	(8.5)	(16.9)
SM32	1141.0	237.2	170.0	307.8	79.3	56.9
		(4.8)	(6.7)	(3.7)	(14.4)	(20.1)
C24	1940.5	336.9	237.2	446.0	125.9	70.6
		(5.8)	(8.2)	(4.4)	(15.4)	(27.5)

*Speedup (scalar/vector)

Table 3. Parameters of the models

Model	Saddle shell		Cooling tower		
	S16	S32	C12	C24	C36
Number of elements	89	305	144	432	1,296
Number of nodes	133	389	169	475	1,369
Number of degrees of freedom	448	1,536	743	2,201	6,551
Semi-bandwidth	87	167	70	130	190

2571 ms of CPU time for each iteration to compute stresses, strains, and nodal forces, and the vector program spends only 68 ms representing a speedup by a factor of 37.8. As we did in evaluating the scalar-vector speedup for the calculation of the stiffness matrix, the timing reported for the scalar program here is equal to the actual CPU time multiplied by a factor of 10 to compare it with the time from the 10-layer vector program.

6. Performance evaluation

To measure the efficiency of the vector program, two problems — 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 into three models. Various parameters of the five models are summarized in Table 3.

In a nonlinear program, various data arrays are saved at the end of a converged step. The saving serves two functions. When the solution does not converge during any step, an attempt for reanalysis is often made using a smaller load or displacement increment than was used in the unconverged step. This is accomplished by restarting the solution at the end of the previous converged step rather than re-analyzing the complete problem from the beginning, which would be wasteful and costly. The data saved at the end of the converged step, thus, serves the purpose of restarting the solutions at the end of any converged step. Equally important is the use of this data to selectively retrieve information for printing and graphic display purposes. In a large problem, it is tedious and unnecessary to obtain a hard copy output of the complete solution.

The volume of data that is saved at the end of any step is quite large and may add significant output overhead. Considerable reduction in this overhead was accomplished by using unformatted output for the related operations. It resulted in I/O CPU time that was less than one percent of the total for the solution of any problem. For evaluating the four modules in the program, therefore, we have neglected the CPU time used for I/O operations.

It may be argued that vectorizing the equation solver only in a finite element computer program is sufficient for all practical purposes (Min and Gupta 1991). We are, therefore, evaluating three programs; Program 1-all scalar, Program 2-vectorized equation solver, rest scalar (partially vector), and Program 3-all vector. The average CPU times per iteration for solving the C24 problem using the three types of program are given in Table 4. As discussed earlier, the reported timing for the element stiffness and stresses, strains and residual force calculations in the scalar Akbar-Gupta program has been adjusted. By vectorizing the equation solver only, we achieve a speedup factor of 1.9 (programs 1/2). The additional speedup factor due to vectorization of the rest of

Table 4. Speedup for the cooling tower problem (C24) with partially and completely vector programs

CPU time per iteration (ms)			Speedup factor	
Program 1 -scalar	Program 2 -Partially vector	Program 3 -vector	Program 1/2	Program 2/3
8833	4545	286	1.9	15.9

Table 5. Performance of various modules in CPU time (ms) for each iteration on a Cray Y-MP.

Model / Module	S16		S32		C12		C24		C36	
	CPU time (ms)	Percent of total	CPU time (ms)	Percent of total	CPU time (ms)	Percent of total	CPU time (ms)	Percent of total	CPU time (ms)	Percent of total
Element	13	24	44	16	17	25	49	17	139	11
Assembly	2	4	6	2	2	3	6	2	18	1
Solver	17	31	175	64	22	32	163	57	934	73
Stress	22	41	48	18	27	40	68	24	194	15
Total	54	100	273	100	68	100	286	100	1285	100
MFLOP/s*	135		202		141		189		232	
Performance Index(%)**	41		61		42		57		70	

*Million of floating-point operations per second of CPU time, is obtained by hpm (Hardware Performance Monitor).
**Percent of maximum possible peak rate, 333 HFLOP/S.

the program (2/3) is 15.9, which is much greater than that obtained by vectorizing the equation solver alone. This result is contrary to our earlier observation in conjunction with the vectorization of a two-dimensional finite element program (Min and Gupta 1991). The change may be attributed to a relative increase in the computational effort for evaluating stiffness matrices, stresses, strains and residual forces in the layered shell element as compared to that for a plane element. Even in the finite element program for plane elements, we had found that vectorization of the non-equation solver modules was desirable. For the present program with layered shell elements, it is clearly mandatory to vectorize all the four modules.

Table 5 gives the CPU time for various modules for the five inelastic problems. The element stiffness matrix computation takes up around 11 to 25 percent of the total CPU time from the largest to the smallest problem. Only around 1 to 4 percent of the total is now consumed by the assembly of the global stiffness matrix calculation. The equation solver uses in the range of 31 to 73 percent of the total CPU time. About 15 to 41 percent of the total is spent by the stresses, strains and nodal force computation. For smaller problems, such as S16 and C12, the stress module emerges as computationally most intensive, and the equation solver is the next most intensive module. For larger problems, S32, C24 and C36, the equation solver become the first and the stress module second in the usage of CPU time.

The speed of operations of a computer is measured in MFLOP/s (million of floating-point operations per second of CPU time) units. For a Cray Y-MP, the theoretical maximum possible speed per procesor is 333 MFLOP/s with a 6 nanosecond clock period, This rate is approached during a chained vector operation with no re-loads, using the add and multiply units continuously. We define a performance index here, which is a percent of the theoretical maximum speed, 333 MFLOP/s, that a computer program run actually accomplishes. The higher the performance index the higher is the vector efficiency of the program. The present vectorization scheme achieves in the range of 135 to 232 MFLOP/s for the five inelastic problems analyzed, which is equivalent to performance indices of 41 to 70 percent. For reference, speeds of 20 to 50 MFLOP/s (performance index, 6 to 15 percent) on the Cray Y-MP are quite common for commercially available finite element software packages.

7. Conclusion

Vector algorithms and the relative importance of the four basic modules of a finite element computer program for inelastic analysis of RC shells are presented. The performance of the present vector program is measured against the scalar Akbar-Gupta program (1985) on a Cray Y-MP supercomputer. For the C24, a cooling tower problem, the scalar-vector speedup factor is 34 in the calculation of element stiffness matrices, 25.3 for the assembly of global stiffness matrix, 27.5 for the solution of equations, and 37.8 for stresses, strains and nodal forces calculation. The overall speedup factor obtained is 30.9.

Even though the equation solver is computationally the most intensive part of a scalar finite element computer program, it was found with the layered shell elements that other modules in the program play a much more important role as far as impact on the scalar-vector speedup is concerned. For the C24 problem, a speedup factor of only 1.9 could be achieved by vectorizing the equation solver alone. The program speeded up further by a factor of 15.9 when the rest of the program was also vectorized. The present vectorization scheme performs in the range of 135 to 232 MFLOP/s for the five inelastic problems analyzed. This is equivalent to performance indices of 41 to 70 percent when compared with the theoretical maximum rate of 333 MFLOP/s. The commercial finite element programs on the Cray Y-MP achieve performance indices in the range of 6 to 15 percentage only.

The availability of the Cray Y-MP machine and our ability to efficiently vectorize the finite element computer program for inelastic analysis of reinforced concrete shells with layered elements made it possible for us to perform several mesh convergence studies. To our knowledge, such studies have not been performed and reported in the scientific literature before. Results of the convergence studies were extremely valuable and have been published elsewhere (Min and Gupta 1993, 1994c). The vector computer program can be used, if necessary, for studying the ultimate behavior of reinforced concrete shells after the completion of design and before construction. It can also be used to optimize the shape of a shell.

Acknowledgements

The supercomputing resources were provided by the North Carolina Supercomputing Center,

Research Triangle Park, North Carolina, U.S.A.

References

Agrawal, O. P., Danhof, K. J. and Kumar, R. (1994). "A superelement model based parallel algorithm for vehicle dynamics." *Computers & Structures*, **51**(4), 411-423.

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.*

Akbar, Habibollah, and Gupta, Ajaya Kumar. (1986). "Ultimate behavior of an R.C. nuclear containment subjected to internal pressure and earthquake." *J. Struct. Engrg., ASCE*, **112**(6), June, 1280-1295.

Barragy, E., Carey, G. F., and Geijn, R. Van De. (1994). "Performance and scalability of finite element analysis for distributed parallel computation." *J. of Parallel and Distributed Computing*, **21**(2), 202-212.

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

Cray SR-2040 6.0, *UNICOS performance utilities reference manual*, Cray Research, Inc.

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

Dongarra, J.J., Moler, C.B., Bunch, J.R., and Stewart, G.W. (1979) *LINPACK User's Guide*. Society for Industrial and Applied Mathematics (SIAM).

Dongarra, J. J., Gustavson, F. G., and Karp, A. (1984a). "Implementing linear algebra algorithms for dense matrices on a vector pipeline machine." *SIAM Review*, **26**(1), January, 91-112.

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

Dongarra, J., Kennedy, K., Messina, P., Sorensen, D. C., and Voigt, R., (1992). *Parallel processing for scientific computing, Proceedings of the fifth SIAM conference*, March 25-27, 1991, in Houston, Tex.

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 Maestrini, S. (1986). "Investigation of hyperbolic cooling tower ultimate behavior." *Engrg. Struct.*, **8**, April, 87-92.

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.

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 Engrg.*, **32**, 347-362.

IBM. Corporation. (1987). *IBM Engineering and Scientific Subroutine Library: Guide and Reference*, Release 2 edition, September. Program number: 5668-863.

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

Levesque, John M., and Williamson Joel W. (1989). *A guidebook of Fortran on supercomputers*. Academic Press, Inc., San Diego, California 92101.

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

Mahmoud, Bahaa Eldin H., and Gupta, Ajaya Kumar. (1993). "Inelastic large displacement behavior and buckling of hyperbolic cooling tower shells." Center for Nuclear Power Plant Structures, Equipment and Piping. North Carolina State University, Raleigh, North Carolina 27695-7908, May, *Report.*

Mang, H. A., Floegl, H., Trappel, F., and Walter, H. (1983). "Wind-loaded reinforced-concrete cooling towers: buckling or ultimate load?" *Engrg. Struct.*, **5**, July, 163-180.

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*, 514.

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. *Reinfored Concrete Shell Research Report.*

Min, Chang Shik, and Gupta, Ajaya Kumar. (1993). "Inelastic behavior of hyperbolic cooling tower." *J. of Struct. Engrg., ASCE,* **119**(7), July, 2235-2255.

Min, Chang Shik, and Gupta, Ajaya Kimar. (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.*

Min, Chang Shik, and Gupta, Ajaya Kumar. (1994c). "Inelastic behavior of reinforced concrete hyperbolic paraboloid saddle shell." *Engrg. Struct.,* **16**(4) 227-2375.

Min, Chang Shik, and Gupta, Ajaya Kumar. (1995), "Vector algorithm for layered reinforced concrete shell element stiffness matrix," *Structural Engineering and Mechanics, An International Journal,* **3**(2), 173-183.

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

Ortega, James M., (1988). *Introduction to Parallel and Vector Solution of Linear Systems,* Plenum Press, New York, N.Y. 10013.

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, Mobile, Alabama, April, *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 & Structures.* **47**(3), 407-418.