From Eqs. (12) and (13), we have

$$M_{xx} = K_{xx}[p_{el} \quad p_{ah}]\begin{bmatrix}\Omega_{el}^2 & 0\\ 0 & \Omega_{ah}^2\end{bmatrix}[p_{el} \quad p_{ah}]^{-1}$$

$$= K_{xx}[p_{el} \quad p_{ah}]\begin{bmatrix}\bar{\Omega}_{el}^{-2} & 0\\ 0 & \Omega_{ah}^{-2}\end{bmatrix}[p_{el} \quad p_{ah}]^t K_{xx}$$

$$= K_{xx}p_{el}\Omega_{el}^{-2}p_{el}^t K_{xx} + K_{xx}p_{ah}\Omega_{ah}^{-2}p_{ah}^t K_{xx}$$

$$= K_{xx}p_{el}\Omega_{el}^{-2}p_{el}^t K_{xx} + K_{xx}p_{ah}p_{ah}^t M_{axx}p_{ah}p_{ah}^t K_{xx} \qquad (14)$$

From Eq. (13), we still have $p_{ah}p_{ah}^t = K_{xx}^{-1} - p_{el}p_{el}^t$, and by substituting this relation into Eq. (14) we obtain Eq. (9) again. Therefore, it is concluded that Eq. (9) tends to replace the unavailable p_{eh} by p_{ah} and Ω_{eh} by Ω_{ah}. Generally, approximately 50% of the analytical modes can be considered acceptable, but it is not unusual that only 5% of the modes are measured. Thus, Eq. (9) provides a reasonable source for compensation for the lack of higher measured modes.

Numerical Examples

A uniform straight beam discretized by 12 elements is used as an example. Only transverse in-plane displacements are considered. The consistent and roughly lumped mass matrices are taken to be M_{xx} and M_{axx}, and the errors of the first eight nonrigid frequencies and mode shapes of the unconstrained structure evaluated with the analytical mass matrix M_{axx} are listed in Tables 1 and 2, respectively; m test modes of the constrained structure by fixing one end of the beam (cantilever beam) are then used to produce M_{xx} with Eq. (9). The errors of the extracted free-free modes by the proposed and Chen et al.'s methods are also shown in Tables 1 and 2 for comparison. It is clearly seen that the proposed method yields an improved result: it can be used with a smaller number of measured modes and gives more accurate modal data of the unconstrained structure for both frequencies and mode shapes.

Conclusions

Chen et al.[2] extended Przemieniecki's analytical method[1] for determining free-free modes from ground constrained test data to deal with the modal truncation problem. In this Note, a structural dynamic model modification method[3–6] is introduced for compensation of the modal truncation effect; the method essentially uses analytical modes in place of the unavailable measured higher ones. The improvement is expected to yield a better approximation of the dynamic model from which one can extract more accurate free- free modes and natural frequencies using the incomplete measured ones of the constrained structure. A simple example is presented to illustrate the performance of the proposed method.

References

[1]Przemieniecki, J. S., *Theory of Matrix Structural Analysis*, McGraw–Hill, New York, 1968, pp. 357–359.

[2]Chen, S., Liu, Z., Han, W., and Ma, A., "Determining Free-Free Modes from Experimental Data of Constrained Structures," *AIAA Journal*, Vol. 32, No. 2, 1994, pp. 440–443.

[3]Baruch, M., and Bar Itzhack, I. Y., "Optimal Weighted Orthogonalization of Measured Modes," *AIAA Journal*, Vol. 16, No. 4, 1978, pp. 346–351.

[4]Wei, F. S., "Stiffness Matrix Correction from Incomplete Test Data," *AIAA Journal*, Vol. 18, No. 10, 1980, pp. 1274, 1275.

[5]Baruch, M., "Optimal Correction of Mass and Stiffness Matrices Using Measured Modes," *AIAA Journal*, Vol. 20, No. 11, 1982, pp. 1623–1626.

[6]Berman, A., and Nagy, E. J., "Improvement of a Large Analytical Model Using Test Data," *AIAA Journal*, Vol. 21, No. 8, 1983, pp. 1168–1173.

Effect of General Sparse Matrix Algorithm on Optimization of Space Structures

Kamal C. Sarma* and Hojjat Adeli†
Ohio State University, Columbus, Ohio 43210

Introduction

THE global stiffness matrix of a structural analysis problem is a sparse matrix with many zero elements. The number of zeros in this matrix increases with the size of the structure. The zero terms inside the stiffness matrix slow down the processing speed of the computations due to unnecessary operations performed on them. To overcome these problems, banded and skyline (envelope or variable band) methods of data structures were developed and are in use for storage of stiffness coefficients and their subsequent operations.[1] However, as the size of the structure increases, the bandwidth will also increase, and the number of zeros within the band becomes large for large structures with hundreds or thousands of members. In this case, the solution by the banded or the skyline method may not be the most efficient one. In this work, we investigate the effect of a general sparse matrix approach on the optimization of large structures using the optimality criteria approach and the Cholesky lower-upper (LU) decomposition method for the solution of the resulting linear simultaneous equations.

General Sparse Matrix Solution

In a general sparse matrix solution approach, the sparse matrix is stored in compact form with the help of some additional information about the locations of the nonzero elements in the sparse matrix.[2] By means of indirect addressing and gather operation (described later), only the nonzero elements are stored in compact form, and the arithmetic operations on zero terms are avoided. The results are then restored back to their respective positions in the original matrix by the scatter operation.

The gather operation means collecting nonzero terms of a sparse matrix and storing them in a one-dimensional array (vector) with the help of pointers to their locations in the sparse matrix. For example, in the term $x[k(i)]$, $k(i)$ is a pointer to the original index i of x in the sparse matrix. The term x is addressed as $x[k(i)]$ indirectly. This is called indirect addressing. Similarly, the scatter operation means spreading back the terms from their compact vector form to their original sparse matrix locations.

Before storing the nonzero elements in compact form, it is necessary to evaluate the fill-ins to be stored along with the nonzero terms. The fill-ins are zero terms in the original sparse matrix, but their positions will be occupied by nonzero terms after the Cholesky decomposition. To record the fill-ins and the nonzero terms, we use a logical binary variable. It is assigned a value of TRUE for nonzero and fill-in terms in the sparse stiffness matrix and FALSE for zero terms.

For every column i, the pointer to the first nonzero element is defined as first_index(i) (Fig. 1). The pointer to the diagonal element of column i is defined as diagonal_index(i) (Fig. 1). Using the two pointers and the row index (Fig. 1), we convert the noncompact sparse Cholesky factorization algorithm into an indirect addressing, compact factorization algorithm. Before doing that, the two pointers, the row index, and the value of each nonzero or fill-in term need to be recorded. The data structure used for storing the pointers and the nonzero values is illustrated in Fig. 1.

Received May 31, 1994; revision received Feb. 8, 1995; accepted for publication Feb. 9, 1995.

*Ph.D. Candidate.

†Professor, Department of Civil Engineering, 2070 Neil Avenue.

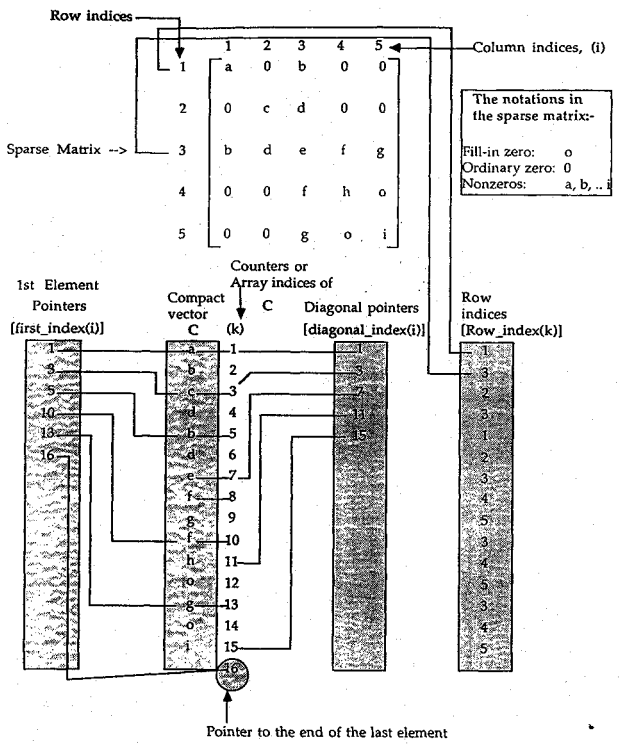

Fig. 1 Data structure for storing sparse matrix in compact form.

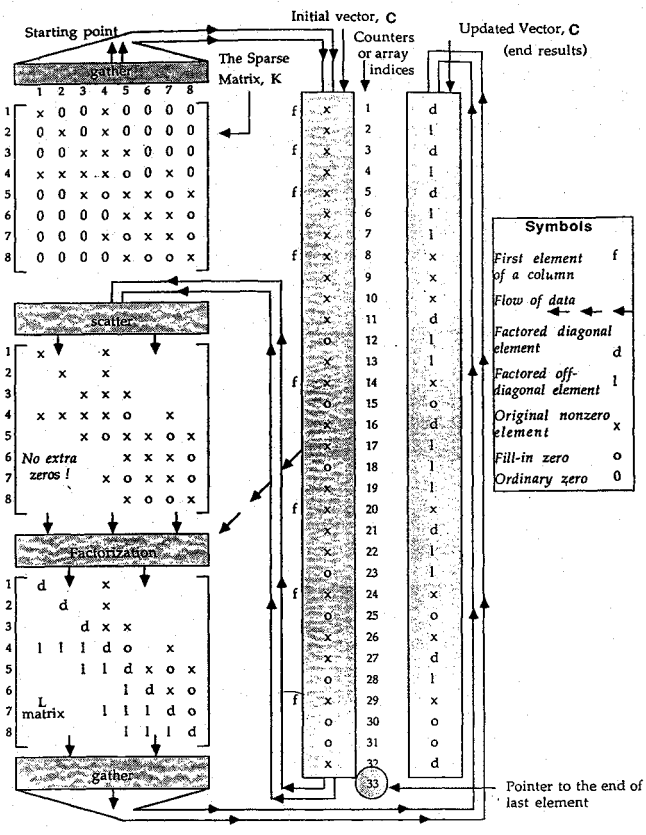

Fig. 2 Pictorial explanation of factorization by sparse matrix algorithm.

The sparsity of the global stiffness matrix depends on the configuration of the structure. It is also dependent on the numbering of the nodes and the degrees of freedom per node. The sparsity pattern does not change during the optimization procedure. Therefore, in the process of optimization, it is sufficient to record the pointers and the row indices at the first iteration only. In succeeding iterations, only the values of the stiffness matrix elements need to be recorded.

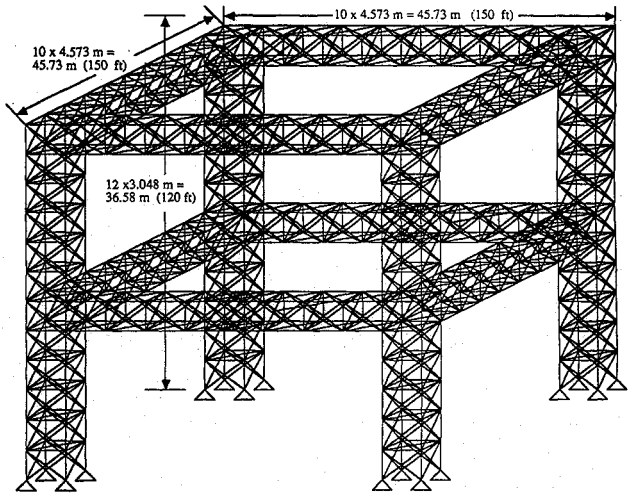

Fig. 3 1968-member space truss.

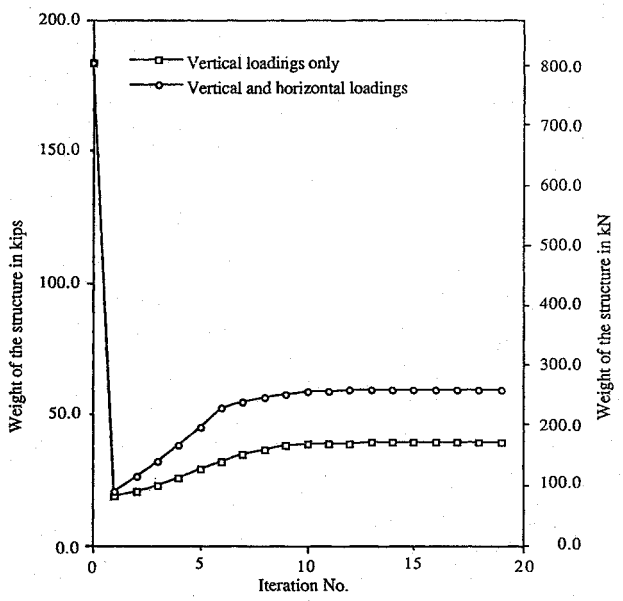

Fig. 4 Iteration history for 1968-member space truss.

In the factorization phase, the LU decomposition of stiffness matrix $\boldsymbol{K}$ is done using only nonzeros and fill-ins. This is done by indirect referencing of the sparse matrix elements with the help of the pointers and row indices. The first step of the factorization procedure is to scatter back the sparse stiffness matrix elements from the contiguous one-dimensional array to their original locations in the sparse matrix through the scatter operation. Next, both the normalization and update operations of the Cholesky factorization procedure are done on the nonzero elements or the fill-ins, in the lower triangular part of the $\boldsymbol{K}$ matrix, using a compact vector $\boldsymbol{C}$. After factorization, the nonzero elements of lower triangular matrix $\boldsymbol{L}$ are stored columnwise in contiguous vector form and overlaid on the corresponding elements of the vector $\boldsymbol{C}$. Thus, no additional storage is needed for $\boldsymbol{L}$. A pictorial explanation of this factorization procedure is presented in Fig. 2. In the solution phase, the forward and the backward solutions are carried out to find the unknown displacements. In this phase, indirect references to the elements are also made.

Example

The general sparse matrix optimization algorithm has been applied to minimum weight design of several large space structures. For the sake of brevity and due to length limitation we present only one example in this paper. This example is a large space truss structure with 1968 members and 432 joints (Fig. 3). The number of linear equations and the size of semibandwidth for this example

are 1296 and 282, respectively. The structure is made of steel with modulus of elasticity of 2×10^5 MPa (29,000.0 ksi) and unit weight of 769 kN/m^3 (490 lb/ft^3). Displacement is limited to ± 5.0 mm (± 0.2 in.) at all nodes in x, y, and z directions. Allowable stress in both tension and compression is 138 MPa (20 ksi). The lower bound for the cross-sectional area is 322.5 mm^2 (0.5 in.^{2}). Two sets of loadings are applied on this structure: 1) vertical loads of 111.2 kN (25 [kilopounds]) at all lower joints of the eight truss girders and 2) vertical loads of 111.2 kN (25 kips) as in the first loading plus horizontal loads of 44.48 kN (10.0 kips) at all of the joints on one face of the structure. Minimum weights of 176.1 kN (39.6 kips) and 261.5 kN (58.8 kips) are obtained for the loading cases 1 and 2, respectively. Iteration histories for the two loading cases are shown in Fig. 4.

Using the general sparse matrix algorithm, optimization of this structure after 20 iterations took 11.8 s of CPU time on a single processor of a Cray Y-MP 8/864 supercomputer. This compares with 25.5 s when the banded matrix approach was used for the solution of the simultaneous equations.

Thus, use of the general sparse matrix algorithm for optimization of large structures results in substantial savings in the CPU time. As mentioned earlier, the general sparse matrix optimization algorithm was applied to several other large structures. In all cases, the general sparse matrix optimization algorithm required less CPU time than using the banded matrix approach. Further, the efficiency of the sparse matrix optimization algorithm increases with the increase in the size of the structure and number of structural analyses required in the optimization procedure.

References

[1]Bathe, K. J., *Finite Element Procedures in Engineering Analysis*, Prentice–Hall, Englewood Cliffs, NJ, 1982.

[2]Duff, I. S., Erisman, A. M., and Reid, J. K., *Direct Methods for Sparse Matrices*, Oxford Univ. Press, London, 1986.