AIAA JOURNAL
Vol. 40, No. 4, April 2002

Algorithms and Application of Sparse Matrix Assembly and Equation Solvers for Aeroacoustics

W. R. Watson*
NASA Langley Research Center, Hampton, Virginia 23681
D. T. Nguyen†
Old Dominion University, Norfolk, Virginia 23529
C. J. Reddy‡
EM, Inc., Hampton, Virginia 23666
V. N. Vatsa§
NASA Langley Research Center, Hampton, Virginia 23681
and
W. H. Tang¶
Hong Kong University of Science and Technology, Kowloon, Hong Kong, People's Republic of China

An algorithm for symmetric sparse equation solutions on an unstructured grid is described. Efficient, sequential sparse algorithms for degree-of-freedom reordering, supernodes, symbolic/numerical factorization, and forward/backward solution phases are reviewed. Three sparse algorithms for the generation and assembly of symmetric systems of matrix equations are presented. The accuracy and numerical performance of the sequential version of the sparse algorithms are evaluated over the frequency range of interest in a three-dimensional aeroacoustics application. Results show that the solver solutions are accurate using a discretization of 12 points per wavelength. Results also show that the first assembly algorithm is impractical for high-frequency noise calculations. The second and third assembly algorithms have nearly equal performance at low values of source frequencies, but at higher values of source frequencies the third algorithm saves CPU time and RAM. The CPU time and the RAM required by the second and third assembly algorithms are two orders of magnitude smaller than that required by the sparse equation solver. A sequential version of these sparse algorithms can, therefore, be conveniently incorporated into a substructuring (or domain decomposition) formulation to achieve parallel computation, where different substructures are handled by different parallel processors.

Nomenclature

$[A]$, $\{F\}$	=	global stiffness matrix and load vector without source effects
$[\bar{A}]$, $\{\bar{F}\}$	=	global stiffness matrix and loads vector with source effects
$[\mathcal{A}]$, $[\mathcal{B}]$, $[\mathcal{C}]$, $[\mathcal{F}]$	=	local element matrices for a rigid wall duct
$\{AD\}$, $\{AN\}$, $\{a\}$	=	one-dimensional arrays containing sparse matrix coefficients
$[A^{(e)}]$, $\{F^{(e)}\}$	=	element stiffness matrix and loads vector
A_{IJ}, $A_{IJ}^{(e)}$, f_I	=	complex matrix coefficients
$[B]$, $[\mathcal{P}]$	=	contributions to the element stiffness matrix due to the exit plane and interior elements
$[\mathcal{D}]$, $[\mathcal{L}]$	=	diagonal and unit lower triangular matrices
d_I	=	value of field variable at local node I
$[E]$	=	element degree-of-freedom matrix
$\{EN\}$	=	discrete error vector
E_r, N_m	=	error and three-dimensional basis functions
F	=	fill in resulting from matrix factorization
$\{F\}_I$	=	Ith component of $\{F\}$
$\{F(N)\}$	=	$\{F\}$ vector of length N
f, k	=	source frequency and free space wave number
H, L, W	=	height, length, and width of three-dimensional duct
$[HA]$, $\{ha\}$	=	matrix of pointers for sparse assembly
$[HA]_{IJ}$	=	coefficient of the Ith row and Jth column of $[HA]$
$[HA(N, M)]$	=	$[HA]$ an $N \times M$ matrix
h, l, w	=	height, length, and width of three-dimensional finite element
$\{IA\}$	=	array containing the number of nonzeroes per row
$\{IC\}$, $\{IE\}$, $\{IET\}$	=	arrays of starting locations of nonzero coefficients
$\{IP\}$, $\{IV\}$	=	permutation and inverse permutation vectors
$\{IR\}$, $\{JC\}$	=	array of row and column indexes for nonzero matrix coefficients
i	=	$\sqrt{-1}$
$\{JA\}$	=	array containing the column numbers of the nonzero off-diagonal matrix coefficients
$\{JE\}$	=	array of element connectivities
$\{JET\}$	=	array of element numbers connected to each degree of freedom

Received 16 February 2001; revision received 1 September 2001; accepted for publication 13 September 2001.

*Senior Research Scientist, Computational Modeling and Simulation Branch, Aerodynamics, Aerothermodynamics, and Acoustics Competency, Mail Stop 128; w.r.watson@larc.nasa.gov. Senior Member AIAA.

†Professor, Civil Engineering, and Head of Multidisciplinary Parallel-Vector Computation Center, 135 Kauf Building; dnguyen@lions.odu.edu.

‡President and Chief Technical Officer, 24 Research Drive; cjreddy@appliedem.com.

§Senior Research Scientist, Computational Modeling and Simulation Branch, Aerodynamics, Aerothermodynamics, and Acoustics Competency, Mail Stop 128; v.n.vatsa@larc.nasa.gov. Senior Member AIAA.

¶Professor and Chairperson, Civil and Structural Engineering Department.

M = number of nonzero coefficients in a sparse matrix, $N + N1$
ME = maximum number of elements connected to a degree of freedom
$\{MM\}$ = array of element numbers connected to a degree of freedom
$\{MP\}$ = array containing the number of elements connected to a degree of freedom
$\{MS\}$ = master degree-of-freedom array
MZ = maximum number of nonzero coefficients per row
N = number of unknowns in the finite element discretization
NE, NP = number of finite elements and degrees of freedom per element
NF = number of fill ins during factorization of a matrix
NX, NY, NZ = total number of transverse, spanwise, and axial nodes
$N1$ = number of nonzero, off-diagonal coefficients before factorization
$N2$ = number of nonzero, off-diagonal coefficients after factorization
p = acoustic pressure field
p_m, p_s = acoustic pressure at local node m and source pressure
Relerr, u_0 = relative error norm and uniform flow speed
S, V = surface and volume of a finite element
X = nonzero value that was modified during matrix factorization
x, y, z = Cartesian coordinates
x_I, y_J, z_K = transverse, spanwise, and axial locations of grid lines
β_{exit} = dimensionless exit admittance
$\partial p/\partial n$ = derivative of the acoustic pressure normal to a surface
ζ, ζ_{exit} = dimensionless wall and exit impedance
$\{\Phi\}$ = global vector of unknowns
$\{\Phi^e\}$, $\{\Phi^{(I,J,K)}\}$ = local vectors of unknowns
$\{\Phi_{FF}\}$, $\{\Phi_{BB}\}$ = intermediate vectors for forward and backward substitution
Φ_I, $F_I^{(e)}$ = vector components
$\{\nabla\}$, ∇^2 = gradient vector and Laplace operator
$\bullet$ = vector dot product

Subscripts

exit, s = exit and source plane index
F, R = factored and reordered matrix
FF, BB = forward and backward substitution
I, J = row and column index of a matrix

Superscripts

e = truss element number
I, J, K = grid line locator for three-dimensional duct
T = matrix or vector transpose
$*$ = complex conjugate

I. Introduction

THREE-DIMENSIONAL aeroacoustics codes that can accurately predict the noise radiated from commercial aircraft are needed.[1] Currently, noise prediction codes require the use of a linear equation solver before radiated noise can be predicted. An optimizer must then run the noise predictive code on a digital computer hundreds of times to achieve an aircraft design with a minimal noise radiation signature.

Currently, industry and government aircraft noise predictive codes are either two-dimensional or treat only axisymmetric noise signatures.[1] When the volumes are three-dimensional, the currently used equation solvers require an excessive amount of CPU time and RAM for their assembly and solution. This excessive CPU time and computer storage restricts aircraft noise prediction codes to low-frequency sound sources in two-dimensional or axisymmetric environments.

Sparse equation solving technologies[2–14] have been developed and are well documented for several engineering applications, and the computational advantage of sparse solver technology over the more conventional technologies (such as band or skyline solvers) has been demonstrated. In addition, for practical engineering applications, system matrix equations must be developed for an unstructured grid to which boundary conditions are often difficult to apply. The finite element method is the simplest for generating the system matrix on an unstructured grid.

Only recently have sparse solver technologies been applied to aeroacoustics.[1,15] In Ref. 1, several direct and iterative equation solvers were evaluated to determine their applicability to two-dimensional duct aeroacoustics computations with the direct sparse solver emerging as the most promising. In Ref. 15, sparse solver equation solving methodology was extended to three-dimensional acoustically lined ducts. However, the work presented in Ref. 15 adopted the assembly strategy that is currently available in the literature for assembling system sparse matrix equations. This simple but inefficient assembly strategy precludes the use of sparse solvers for three-dimensional aeroacoustic computations.[15]

Most, if not all, major codes for analysis and optimal design allow users to select either iterative or direct equation solvers. For nacelle aeroacoustics computations, iterative solvers are not as robust as direct solvers because the nacelle equation system is poorly conditioned.[1] Iterative solution methods, when applied to systems of poorly conditioned equations, have the disadvantage that they do not converge, or they converge very slowly. A further disadvantage of applying iterative solution methods to solve the nacelle equation system is that the nacelle equation system often contains multiple right-hand sides. Iterative methods are not as efficient as direct methods on equation systems with multiple right-hand sides because the equation system must be reformed and resolved for each right-hand side.

The long-term objective of this research is to acquire the capability to design quiet aircraft in a fully three-dimensional aeroacoustic environment using direct sparse solver technologies and the finite element methodogy. The current paper has two objectives. The first objective is to bridge the gap between the aeroacousticians (who may not have a comprehensive knowledge of sparse assembly and equation solver technologies) and members of the sparse research community (who may not have comprehensive knowledge of finite element analysis and aeroacoustics). The second objective is to present efficient algorithms for assembling sparse matrix equations.

Section II describes three sparse assembly algorithms for generating systems of sparse linear equations. Section III describes the template that is used to develop a complete, unstructured grid, finite element code, that is, equation reordering, symbolic/numerical factorization, supernodes/loop unrolling, and forward/backward solution phases. Section IV presents a detailed formulation of the element stiffness matrices that will be assembled using the sparse assembly algorithms to form the system matrix for a three-dimensional duct aeroacoustics application. Finally, Sec. V discusses the accuracy and numerical performance of the developed algorithms over the frequency range of interest for a three-dimensional aeroacoustics application. Note that although the sparse algorithms presented assume that the system matix equation is symmetric, these algorithms are easily extendible to nonsymmetric systems of equations. The algorithms can also be conveniently incorporated into a substructuring (or domain decomposition) formulation to take advantage of parallel computation to further reduce CPU time and RAM.

II. Sparse Assembly Algorithms for Symmetric Systems

Figure 1 is a two-dimensional truss (or rod) structure assembled from individual truss elements labeled (1), (2), . . . , (13) that are interconnected at eight nodes labeled 1, 2, . . . , 8. An element (e) of the structure is assumed to possess only two points of connection, and the external loads are assumed to be applied at the nodes of the truss elements. Only a single degree of freedom (DOF) at each node is assumed. To further simplify discussions, it is assumed that, by a separate calculation, the element stiffness matrix and external load vector for the truss element (e) are known and expressed as

$$\left[A^{(e)}(2,2)\right] = \begin{bmatrix} A_{11}^{(e)} & A_{12}^{(e)} \\ A_{21}^{(e)} & A_{22}^{(e)} \end{bmatrix}, \qquad \left\{F^{(e)}(2)\right\} = \begin{Bmatrix} F_1^{(e)} \\ F_2^{(e)} \end{Bmatrix} \quad (1)$$

Under the assumption of Eq. (1), the 13 truss elements (Fig. 1) may be assembled using the rules of finite element assembly[16] to obtain the system matrix equation

$$[A]\{\Phi\} = \{F\} \quad (2)$$

$$[A(N,N)] = \sum_{e=1}^{e=13} \left[A^{(e)}(2,2)\right]$$

$$= \begin{bmatrix} A_{11} & A_{12}^{(4)} & 0 & A_{21}^{(3)} & A_{21}^{(5)} & 0 & A_{21}^{(1)} & 0 \\ A_{21}^{(4)} & A_{22} & A_{12}^{(7)} & 0 & A_{21}^{(8)} & 0 & 0 & 0 \\ 0 & A_{21}^{(7)} & A_{33} & 0 & A_{21}^{(9)} & A_{21}^{(11)} & 0 & A_{21}^{(12)} \\ A_{12}^{(3)} & 0 & 0 & A_{44} & A_{12}^{(6)} & 0 & A_{21}^{(2)} & 0 \\ A_{21}^{(5)} & A_{12}^{(8)} & A_{12}^{(9)} & A_{21}^{(6)} & A_{55} & A_{12}^{(10)} & 0 & 0 \\ 0 & 0 & A_{12}^{(11)} & 0 & A_{21}^{(10)} & A_{66} & 0 & A_{12}^{(13)} \\ A_{12}^{(1)} & 0 & 0 & A_{12}^{(2)} & 0 & 0 & A_{77} & 0 \\ 0 & 0 & A_{12}^{(22)} & 0 & 0 & A_{21}^{(13)} & 0 & A_{88} \end{bmatrix} \quad (3)$$

$$A_{11} = A_{22}^{(1)} + A_{22}^{(3)} + A_{11}^{(4)} + A_{22}^{(5)}, \qquad A_{22} = A_{22}^{(4)} + A_{11}^{(7)} + A_{22}^{(8)}$$

$$A_{33} = A_{22}^{(7)} + A_{22}^{(9)} + A_{22}^{(11)} + A_{22}^{(12)}, \qquad A_{44} = A_{22}^{(2)} + A_{11}^{(3)} + A_{11}^{(6)}$$

$$A_{55} = A_{22}^{(5)} + A_{22}^{(6)} + A_{11}^{(8)} + A_{11}^{(9)} + A_{11}^{(10)}$$

$$A_{66} = A_{22}^{(6)} + A_{11}^{(11)} + A_{41}^{(13)}, \qquad A_{77} = A_{11}^{(1)} + A_{11}^{(2)}$$

$$A_{88} = A_{22}^{(12)} + A_{22}^{(13)} \quad (4)$$

$$\{\Phi(N)\} = \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ \Phi_7 \\ \Phi_8 \end{Bmatrix}$$

$$\{F(N)\} = \begin{Bmatrix} F_2^{(1)} + F_2^{(3)} + F_1^{(4)} + F_2^{(5)} \\ F_2^{(4)} + F_1^{(7)} + F_2^{(8)} \\ F_2^{(7)} + F_2^{(9)} + F_2^{(11)} + F_1^{(12)} \\ F_2^{(2)} + F_1^{(3)} + F_6^{(1)} \\ F_2^{(5)} + F_2^{(6)} + F_1^{(8)} + F_1^{(9)} + F_1^{(10)} \\ F_2^{(6)} + F_1^{(11)} + F_1^{(13)} \\ F_1^{(1)} + F_2^{(3)} + F_1^{(2)} \\ F_2^{(12)} + F_2^{(13)} \end{Bmatrix} \quad (5)$$

Although only a single DOF (Φ_I) is assumed at node I, the discussion to follow is easily extended to q DOF per node by extending the coefficients in $[A]$, that is, $A_{IJ}^{(e)}$, to $q \times q$ submatrices. The rules of matrix algebra would then be applied to each $q \times q$ submatrix as if it were a scalar.

A. Sparse Data Formats for the System Matrix

For the sake of brevity, in the discussions to follow it will be assumed that the element stiffness matrix is symmetric so that

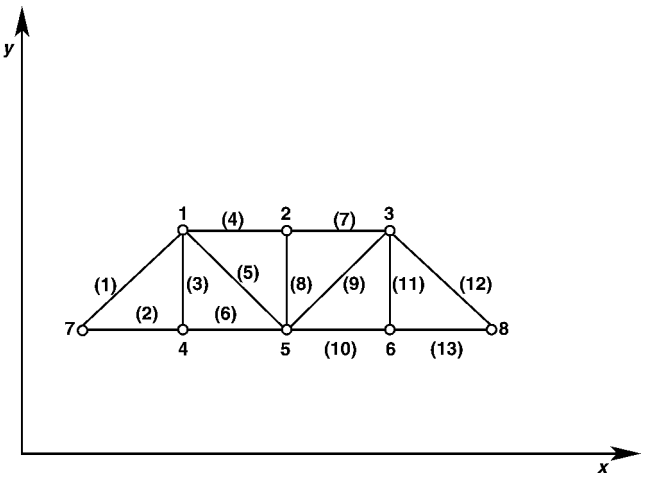

Fig. 1 Two-dimensional truss sample problem.

$$\left[A^{(e)}\right]_{JI} = \left[A^{(e)}\right]_{IJ} \quad (6)$$

Under the assumptions of Eq. (6), the system matrix $[A]$ is also symmetric and can be written in the form

$$[A(N,N)]$$

$$= \begin{bmatrix} A_{11} & A_{12}^{(4)} & 0 & A_{12}^{(3)} & A_{12}^{(5)} & 0 & A_{12}^{(1)} & 0 \\ A_{12}^{(4)} & A_{22} & A_{12}^{(7)} & 0 & A_{12}^{(8)} & 0 & 0 & 0 \\ 0 & A_{12}^{(7)} & A_{33} & 0 & A_{12}^{(9)} & A_{12}^{(11)} & 0 & A_{12}^{(12)} \\ A_{12}^{(3)} & 0 & 0 & A_{44} & A_{12}^{(6)} & 0 & A_{12}^{(2)} & 0 \\ A_{12}^{(5)} & A_{12}^{(8)} & A_{12}^{(9)} & A_{12}^{(6)} & A_{55} & A_{12}^{(10)} & 0 & 0 \\ 0 & 0 & A_{12}^{(11)} & 0 & A_{12}^{(10)} & A_{66} & 0 & A_{12}^{(13)} \\ A_{12}^{(1)} & 0 & 0 & A_{12}^{(2)} & 0 & 0 & A_{77} & 0 \\ 0 & 0 & A_{12}^{(12)} & 0 & 0 & A_{12}^{(13)} & 0 & A_{88} \end{bmatrix} \quad (7)$$

The sparse descriptions of any symmetric system matrix $[A]$ [see Eq. (7)] is fully described by the four one-dimensional vectors

$$\{IA(N)\} = \{4, 2, 3, 2, 1, 1, 0, 0\}^T$$

$$\{JA(N1)\} = \{2, 4, 5, 7, 3, 5, 5, 6, 8, 5, 7, 6, 8\}^T \quad (8)$$

$$\{AD(N)\} = \{A_{11}, A_{22}, A_{33}, A_{44}, A_{55}, A_{66}, A_{77}, A_{88}\}^T \quad (9)$$

$$\{AN(N1)\} = \left\{A_{12}^{(4)}, A_{12}^{(3)}, A_{12}^{(5)}, A_{12}^{(1)}, A_{12}^{(7)}, A_{12}^{(8)}, A_{12}^{(9)}, A_{12}^{(11)}, A_{12}^{(12)}, A_{12}^{(6)}, A_{12}^{(2)}, A_{12}^{(10)}, A_{12}^{(13)}\right\}^T \quad (10)$$

B. Application of Boundary Conditions

In most engineering applications, the field variable at several boundary nodes may require constraints to satisfy a Dirichlet boundary condition of the form

$$\{\Phi\}_I = d_I \quad (11)$$

where d_I is the specified value of the field variable at node I. Dirichlet boundary conditions may be applied at the element or system level. The impact of applying Dirichlet boundary conditions on the system matrix equation is identical whether applied at the element or system level. We will show the relatively easy process of applying Dirichlet boundary conditions at the element level and their impact on the system matrix equation [Eq. (2)].

The process for inserting the Dirichlet boundary condition, $\{\Phi\}_I = d_I$, is as follows:

1) The column of $[A^{(e)}]$ corresponding to the Ith DOF is multiplied by d_I, and the result is subtracted from $\{F^{(e)}\}$.

2) The column corresponding to the Ith DOF in $[A^{(e)}]$ is made zero.

3) The row corresponding to the Ith DOF in $[A^{(e)}]$ is made zero.

4) The modified element matrix and the modified element load vector are assembled.

5) $[A]_{II}$ is made equal to unity, and $\{F\}_I$ is made equal to d_I.

Thus, applying Dirichlet boundary conditions to the system matrix equation modifies Eq. (2) to

$$[\bar{A}]\{\Phi\} = \{\bar{F}\} \tag{12}$$

The numerical values of the coefficients in the modified system matrix $[\bar{A}]$ remain unchanged from those in $[A]$, except for a few that are made zero during the application of the Dirichlet boundary conditions. Therefore, we will illustrate application of the assembly algorithms to the nonzero pattern of $[A]$.

C. Sparse Assembly Algorithms

Three symmetric sparse assembly algorithms will be explained in this section. The purpose of each assembly algorithm is to generate the system loads vector $\{\bar{F}\}$ and the four vectors defined by Eqs. (8–10), which correspond to $[\bar{A}]$. The assembly algorithms are discussed starting with the simplest and proceeding to the most complex.

1. *Algorithm 1*

The main ideas of this algorithm can be summarized by the following computational tasks:

1) Find how many and which elements are connected to each DOF.

a) Input data for N, NE, NP, and the elements connectivity (see Fig. 1).

b) Compute the number of elements associated with each DOF, and store this information into the one-dimensional integer vector $\{MP(N)\}$.

c) Find the element numbers associated with each DOF, and store this information into the two-dimensional integer matrix $[MM(N, ME)]$.

2) Retrieve the stiffness matrix attached to each DOF, perform sparse matrix assembly one row at a time, and extract the four one-dimensional vectors required for the sparse equation solver.

2. *Algorithm 2*

The nonzero patterns of the symmetrical matrix $[A]$ [see Eq. (7)] for the two-dimensional truss example problem (Fig. 1) can be completely described by the two one-dimensional integer vectors

$$\{IR(M)\} = \{7, 1, 1, 4, 4, 1, 1, 2, 1, 5, 4, 2, 3, 2, 3, 5, 6, 3, 3, 8, 6\}^T \tag{13}$$

$$\{JC(M)\} = \{7, 7, 1, 7, 4, 4, 2, 2, 5, 5, 5, 3, 3, 5, 5, 6, 6, 6, 8, 8, 8\}^T \tag{14}$$

and the following integer matrix:

$$[HA(N, MZ)] = \begin{bmatrix} 2 & 3 & 6 & 7 & 9 \\ 8 & 12 & 14 & 0 & 0 \\ 13 & 15 & 18 & 19 & 0 \\ 4 & 5 & 11 & 0 & 0 \\ 10 & 16 & 0 & 0 & 0 \\ 17 & 21 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 20 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{15}$$

The two one-dimensional integer vectors in Eqs. (13) and (14) are constructed by cycling through each element (e) in increasing order and then determining the row and column index of each nonzero coefficient from the connectivity array for each element (Fig. 1). Note that the matrix $[HA]$ contains locations (or pointers) that are used to refer to vectors $\{IR\}$ and $\{JC\}$. For example, the values of $[HA]_{41} = 4$, $[HA]_{42} = 5$, and $[HA]_{43} = 11$ indicate that row 4 of matrix $[A]$ will have these nonzero terms. The exact locations (row and column numbers) of those three nonzero terms in $[A]$ can be referred to as

$$\{IR\}_4 = 4, \qquad \{JC\}_4 = 7, \qquad \{IR\}_5 = 4$$
$$\{JC\}_5 = 4, \qquad \{IR\}_{11} = 4, \qquad \{JC\}_{11} = 5 \tag{16}$$

Thus, the three nonzero terms of the fourth row of $[A]$ are located at row 4, column 7; row 4, column 4; and row 4, column 5, respectively [see Eq. (7)].

The integer matrix $[HA]$ and system matrix $[A]$ can be alternatively stored as one-dimensional vectors:

$$\{ha(M)\} = \{2, 3, 6, 7, 9, 8, 12, 14, 13, 15, 18, 19, 4, 5, 11, 10, 16, 17, 21, 1, 20\}^T \tag{17}$$

$$\{a(M)\} = \left\{A_{12}^{(4)}, A_{12}^{(3)}, A_{22}, A_{12}^{(7)}, A_{33}, A_{12}^{(8)}, A_{12}^{(12)}, A_{12}^{(6)}, A_{44}, A_{12}^{(2)}, A_{66}, A_{12}^{(13)}, A_{12}^{(5)}, A_{12}^{(1)}, A_{12}^{(11)}, A_{12}^{(9)}, A_{55}, A_{12}^{(10)}, A_{88}, A_{11}, A_{77}\right\}^T \tag{18}$$

The main ideas of algorithm 2 can be summarized by the following computational tasks:

1) After initializing $\{ha\}$ to a zero vector, process all elements (in ascending order) to obtain the integer vectors $\{IR\}$ and $\{JC\}$ while assembling $[A]$ into $\{a\}$.

2) Separate the diagonal and nonzero off-diagonal terms of $[A]$ from $\{a\}$ and store this information in $\{AD\}$ and $\{AN\}$. Separate the diagonal and off-diagonal terms in $\{IR\}$ and $\{JC\}$, and compute $\{JA\}$ and $\{IA\}$.

3. *Algorithm 3*

The nonzero patterns of the symmetrical system matrix $[A]$ can be completely described by $\{JA\}$ and the following one-dimensional integer vector:

$$\{IC(N+1)\} = \{1, 5, 7, 10, 12, 13, 14, 14, 14\}^T \tag{19}$$

where

$$\{IA\}_I = \{IC\}_{I+1} - \{IC\}_I \tag{20}$$

and the variable $N1$ can be conveniently computed as

$$N1 = \{IC\}_{N+1} - \{IC\}_1 \tag{21}$$

The element connectivity information for the two-dimensional truss sample problem (Fig. 1) is fully described by the element-DOF matrix

$$[E(NE, N)] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{22}$$

Each row of $[E]$ contains exactly two nonzeros because each element has two points of connection, or nodes, to the structure. Thus, $[E]_{IJ}$ is nonzero only if node J is a node for element I. For example, the first row of $[E]$ contains a unit value only in columns 1 and 7, indicating that the first element of the truss is connected to nodes 1 and 7 only

(Fig. 1). The concept of an element-DOF matrix is easily extended to q DOF per node by extending each of the unity coefficients in $[E]$ to a $q \times q$ identity matrix.

To minimize the RAM, it is convenient to describe the element-DOF matrix $[E]$ by the two one-dimensional vectors

$$\{IE(NE+1)\} = \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27\}^T \tag{23}$$

$$\{JE(NE \times NP)\} = \{7, 1, 7, 4, 4, 1, 1, 2, 1, 5, 4, 5, 2, 3, 5, 2, 5, 3, 5, 6, 6, 3, 3, 8, 6, 8\}^T \tag{24}$$

and the transpose of the element-DOF matrix $([E]^T)$ by the following two one-dimensional vectors:

$$\{IET(N+1)\} = \{1, 5, 8, 12, 15, 20, 23, 25, 27\}^T \tag{25}$$

$$\{JET(NE \times NP)\} = \{1, 3, 4, 5, 4, 7, 8, 7, 9, 11, 12, 2, 3, 6, 5, 6, 8, 9, 10, 10, 11, 13, 1, 2, 12, 13\}^T \tag{26}$$

The main ideas of algorithm 3 can be summarized by the following computational tasks:

1) Assume that $\{IE\}$, $\{JE\}$, $\{IET\}$, and $\{JET\}$ have already been defined from the connectivity information (see Fig. 1).

a) Compute $\{IC\}$ and $\{JA\}$ (symbolic assembly phase).

b) Compute $\{IA\}$ from Eq. (20).

2) Assume that vectors $\{IA\}$ and $\{JA\}$ have already been defined from the symbolic assembly (task 1). Compute $\{AN\}$ and $\{AD\}$ from $[A^{(e)}]$ (numerical assembly phase).

III. Sparse Algorithms for Solving Symmetrical Equations

In this section, the major tasks involved in solving sparse systems of linear equations are briefly explained. The success of the sparse solver is due to improved technologies (i.e., equation reordering, matrix decomposition, supernodes and loop unrolling, forward/backward solution phases) and bookkeeping strategies ideal for implementation on a digital computer. More detailed information on improved technologies can be obtained from Refs. 2–14.

A. Sparse Reordering Algorithms

After imposing the boundary conditions, the modified stiffness matrix $[\bar{A}]$ can be obtained from $[A]$ as indicated in the discussions before Eq. (12). Equation (12) should never be solved directly. To further simplify the discussions, we will assume that matrix $[\bar{A}]$ has the following numerical values:

$$[\bar{A}(N,N)] = \begin{bmatrix} 110 & 7 & 4 & 0 & 5 & 3 \\ 7 & 112 & 0 & 2 & 0 & 0 \\ 4 & 0 & 66 & 0 & 0 & 0 \\ 0 & 2 & 0 & 11 & 1 & 0 \\ 5 & 0 & 0 & 1 & 88 & 0 \\ 3 & 0 & 0 & 0 & 0 & 44 \end{bmatrix} \tag{27}$$

Thus, in this case $N = 6$ and $N1 = 6$. During the factorization phase, many of the zero-value terms appearing in Eq. (27) may become nonzero. For maximum efficiency of storage and solution time, the equations are reordered so that the number of nonzero terms that occur during factorization are minimized. These extra nonzero terms created during the factorization of $[\bar{A}]$ are referred to as fill ins and are denoted by the symbols F in the following equation:

$$[\bar{A}_F(N,N)] = \begin{bmatrix} X & X & X & 0 & X & X \\ & X & F & X & F & F \\ & & X & F & F & F \\ & & & X & X & F \\ & & & & X & F \\ & & & & & X \end{bmatrix} \tag{28}$$

In Eq. (28), one has eight extra (or new) nonzero fill ins. As a result,

$$NF = 8 \tag{29}$$

$$N2 = N1 + NF = 6 + 8 = 14 \tag{30}$$

In general, the number of nonzero coefficients in the upper triangular part of $[\bar{A}]$ after factorization ($N2$) is much larger than those before factorization ($N1$).

The purpose of reordering algorithms [multiple minimum degrees (MMD), nested dissection, or METIS algorithms] is to rearrange the nonzero terms of $[\bar{A}]$, defined in Eq. (27), to different locations so that $N2$ is minimized.[5,17–23] For example, applying the MMD reordering algorithm to $[\bar{A}]$ will result in the following permutation and inverse permutation vectors:

$$\{IP(N)\} = \{5, 6, 3, 1, 4, 2\}^T, \qquad \{IV(N)\} = \{4, 6, 3, 5, 1, 2\}^T \tag{31}$$

With the permutation array $\{IP\}$, the matrix $[\bar{A}]$ in Eq. (27) can be transformed into

$$[\bar{A}_R(N,N)] = \begin{bmatrix} 11 & 0 & 0 & 1 & 0 & 2 \\ 0 & 44 & 0 & 0 & 3 & 0 \\ 0 & 0 & 66 & 0 & 4 & 0 \\ 1 & 0 & 0 & 88 & 5 & 0 \\ 0 & 3 & 4 & 5 & 110 & 7 \\ 2 & 0 & 0 & 0 & 7 & 112 \end{bmatrix} \tag{32}$$

Now, if one factorizes $[\bar{A}_R]$, there will be only one fill in that occurs, as follows:

$$[\bar{A}_{RF}(N,N)] = \begin{bmatrix} X & 0 & 0 & X & 0 & X \\ & X & 0 & 0 & X & 0 \\ & & X & 0 & X & 0 \\ & & & X & X & F \\ & & & & X & X \\ & & & & & X \end{bmatrix} \tag{33}$$

B. Sparse Symbolic Factorization

The reordered matrix $[\bar{A}_R]$ can be described by the following four one-dimensional vectors:

$$\{IA(N+1)\} = \{1, 3, 4, 5, 6, 7, 7\}^T$$

$$\{JA(N1)\} = \{4, 6, 5, 5, 5, 6\}^T \tag{34}$$

$$\{AD(N)\} = \{11, 44, 66, 88, 110, 112\}^T$$

$$\{AN(N1)\} = \{1, 2, 3, 4, 5, 7\}^T \tag{35}$$

In this example, $N = 6$ and $N1 = 6$. Before performing the numerical factorization, it is necessary to go through the sparse symbolic factorization, so that the following hold true:

1) The nonzero pattern of $[\bar{A}_{RF}]$ can be determined (including the locations of fill ins).

2) The value of $N2$ can be determined so that adequate computer memory can be allocated for the subsequent sparse numerical factorization phase.

On completion of the sparse symbolic factorization phase, the nonzero patterns of $[\bar{A}_{RF}]$ are completely known, and the modified versions of Eqs. (34) and (35) for the factored matrix $[\bar{A}_{RF}]$ can be computed as

$$\{IA(N+1)\} = \{1, 3, 4, 5, 7, 8, 8\}^T$$

$$\{JA(N2)\} = \{4, 6, 5, 5, 5, 6, 6\}^T \tag{36}$$

In this case,

$$N2 = N1 + NF = 6 + 1 = 7 \tag{37}$$

Efficient sparse symbolic factorization algorithms and detailed FORTRAN coding can be found elsewhere.[2,5–7]

C. Finding Supernodes

To understand the concept of a supernode (or master node), notice that, in Eq. (33), rows 2–3 and 4–5 have the same nonzero patterns. That is, the nonzero terms in rows 2–3 correspond to the same column numbers. Equation (33) can be used to define a master DOF vector

$$\{MS(N)\} = \{1, 2, 0, 2, 0, 1\}^T \tag{38}$$

The master DOF vector $\{MS\}$ is based on the assumed system matrix $[\bar{A}_{RF}]$ defined in Eq. (33). Once Eq. (38) has been defined, effective loop-unrolling techniques[2,23] can be used to improve computational speed during the sparse numerical factorization phase.

D. Sparse Numerical Factorization Phase

The strategies employed in this phase are quite similar to the ones used during the sparse symbolic factorization phase and have been well documented in the literature.[5,24] The reordered system matrix $[\bar{A}_R]$ can be decomposed or factorized as

$$[\bar{A}_R] = [\mathcal{L}][\mathcal{D}][\mathcal{L}]^T \tag{39}$$

Here, $[\mathcal{D}]$ is a diagonal and $[\mathcal{L}]$ is unit lower triangular matrix, and

$$[\mathcal{D}]_{II} = \left\{ \begin{array}{ll} [\bar{A}_R]_{11} & (I = 1) \\ [\bar{A}_R]_{II} - \displaystyle\sum_{K=1}^{I-1} [\mathcal{D}]_{KK}[\mathcal{L}]_{KI} & (I = 2, 3, \ldots, N) \end{array} \right\} \tag{40}$$

$$[\mathcal{L}]_{IJ} = \left\{ \begin{array}{ll} \dfrac{[\bar{A}_R]_{IJ}}{[\mathcal{D}]_{II}} & (I = 1, \quad J = 2, \ldots, N) \\ [\bar{A}_R]_{IJ} - \displaystyle\sum_{K=1}^{I-1} \frac{[\mathcal{D}]_{KK}[\mathcal{L}]_{KI}[\mathcal{L}]_{KJ}}{[\mathcal{D}]_{II}} & (I \neq 1, \quad J = I+1, \ldots, N) \end{array} \right\} \tag{41}$$

E. Solution to the System Matrix Equation

The solution to the system matrix equation [Eq. (12)] is obtained in three phases:

1) In the first phase (forward solution phase), an intermediate solution vector $\{\Phi_{FF}\}$ is computed from the solution of the matrix equation

$$[\mathcal{L}]\{\Phi_{FF}\} = \{\bar{F}_R\} \tag{42}$$

2) In the second phase (backward solution phase), a vector $\{\Phi_{BB}\}$ is computed from the matrix equation

$$[\mathcal{D}][\mathcal{L}]^T\{\Phi_{BB}\} = \{\Phi_{FF}\} \tag{43}$$

3) In the third phase (back transformation phase), the vector $\{\Phi_{BB}\}$ is transformed back to the original unknown vector $\{\Phi\}$ by utilizing the inverse permutation vector $\{IV\}$.

IV. Three-Dimensional Aeroacoustics Application

The developed algorithm will be exercised to study the propagation of acoustic pressure waves in a three-dimensional duct lined with sound absorbing materials (acoustic liners) as depicted in Fig. 2. The duct is spanned by axial coordinate z, transverse coordinate x, and spanwise coordinate y. The source plane is located at $z = 0$, and the source plane acoustic pressure p_s is assumed known. At the exit plane, the dimensionless exit acoustic impedance ζ_{exit} is assumed known. In the duct, air is flowing along the positive z axis at a subsonic speed of u_0, and the duct has acoustic liners along its upper, lower, and two sidewalls. The duct walls are assumed to be locally reacting so that the sound absorbing properties of the acoustic liners results from the dimensionless wall impedance ζ that is assumed known. The sound source pressure, dimensionless exit impedance, and dimensionless wall impedance are assumed functions of position along their respective boundaries.

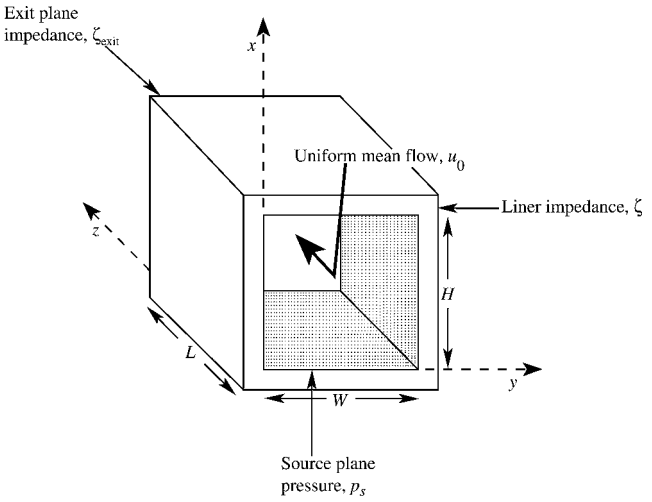

Fig. 2 Three-dimensional duct and coordinate system.

A. Mathematical Formulation

The mathematical formulation of the duct acoustics problem (Fig. 2) does not lead to a boundary value problem that is formally self-adjoint and will not lead to a symmetric system matrix when airflow is considered. Thus, the analysis in the foregoing discussion does not allow for airflow because the current paper focuses on symmetric systems. With zero airflow in the duct ($u_0 = 0$), the mathematical problem is to find the solution to Helmholtz's equation[15]

$$\nabla^2 p + k^2 p = 0 \tag{44}$$

Along the source plane of the duct ($z = 0$), the boundary condition is given in term of a Dirichlet boundary condition:

$$p = p_s \tag{45}$$

The wall boundary condition is

$$\frac{\partial p}{\partial n} = -ik\frac{p}{\zeta} \tag{46}$$

At the duct termination ($z = L$), the ratio of acoustic pressure to the axial component of acoustic particle velocity is proportional to the known dimensionless exit impedance. When expressed in terms of the acoustic pressure, this boundary condition is

$$\frac{\partial p}{\partial n} = -ik\frac{p}{\zeta_{\text{exit}}} \tag{47}$$

Equations (44–47) form a well-posed boundary value problem for which exact solutions for the acoustic pressure field are generally not known. A solution for the acoustic pressure field satisfying this boundary value problem is required to predict and reduce the radiated noise. An approximate solution for the acoustic pressure field can be obtained using numerical techniques such as the finite element method.

B. Finite Element Model

The approximate solution for the sound field in the duct is obtained by subdividing the duct and representing the acoustic field within each subdivision by relatively simple functions. Because the duct of interest is a rectangular prism, the computational domain is divided into a number of smaller rectangular prisms (or elements) as shown in Fig. 3. These elements are considered interconnected at joints called nodes. The most widely used method for locating the nodes in the discretization is to divide the physical volume into NX, NY, and NZ grid lines in the x, y, and z directions, respectively, as shown in Fig. 3. Each node of an element can be located by identifying an ordered triplet, (x_I, y_J, z_K). Similarly, each element in the assemblage can be identified by an ordered triplet of integers (I, J, K). A typical rectangular prism element (I, J, K) is shown in Fig. 4. Each element consists of eight local node numbers labeled

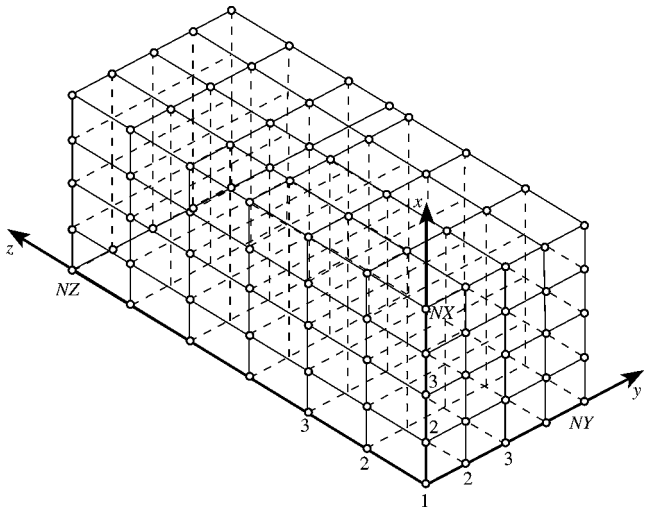

Fig. 3 Three-dimensional finite element discretization.

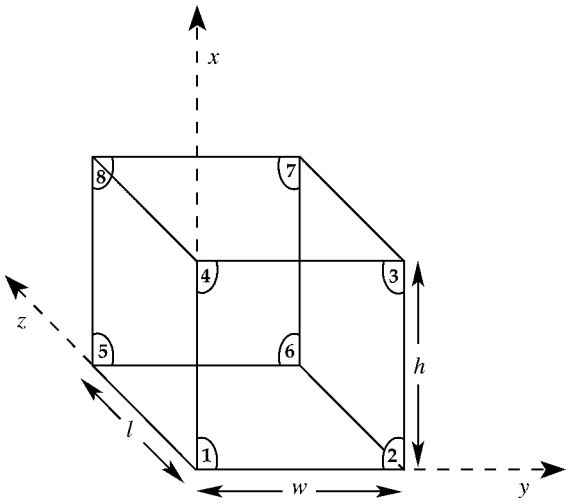

Fig. 4 Typical three-dimensional element and local node numbering system.

1, 2, . . . , 8. Each element is considered to have a dimension of h, w, and l in the x, y, and z directions, respectively, as shown.

C. Element Stiffness Matrix

Galerkin's finite element method is used to compute the element stiffness matrix. The field error function is defined as

$$E_r = \nabla^2 p + k^2 p \tag{48}$$

Within each element, p is represented as a linear combination of eight functions, $N_1, N_2, \ldots, N_8$,

$$p = \sum_{m=1}^{m=8} N_m p_m \tag{49}$$

$$N_1 = \left(1 - \frac{x}{h}\right)\left(1 - \frac{y}{w}\right)\left(1 - \frac{z}{l}\right)$$

$$N_2 = \left(1 - \frac{x}{h}\right)\left(\frac{y}{w}\right)\left(1 - \frac{z}{l}\right), \qquad N_3 = \frac{xy}{(wh)(1 - z/l)}$$

$$N_4 = \left(\frac{x}{h}\right)\left(1 - \frac{y}{w}\right)\left(1 - \frac{z}{l}\right)$$

$$N_5 = \left(1 - \frac{x}{h}\right)\left(1 - \frac{y}{w}\right)\left(\frac{z}{l}\right)$$

$$N_6 = \left(1 - \frac{x}{h}\right)\left(\frac{y}{w}\right)\left(\frac{z}{l}\right), \qquad N_7 = \frac{xyz}{whl}$$

$$N_8 = \left(\frac{x}{h}\right)\left(1 - \frac{y}{w}\right)\left(\frac{z}{l}\right) \tag{50}$$

The linear combination [Eq. (49)] comprises a complete set of basis functions.

For a typical element (I, J, K), contributions to the minimization of the field error function due to local node m over the computational volume V are

$$\int_V E_r N_m \, \mathrm{d}V = \int_V [\nabla^2 p + k^2 p] N_m \, \mathrm{d}V \tag{51}$$

The second derivative terms in Eq. (51) are reduced to first derivatives using Green's second identity

$$\int_V E_r N_m \, \mathrm{d}V = \int_V \left[-\{\nabla\} p \cdot \{\nabla\} N_m + k^2 p N_m\right] \mathrm{d}V + \int_S \frac{\partial p}{\partial n} N_m \, \mathrm{d}S \tag{52}$$

Elimination of the second derivative terms from the volume integral in Eq. (51) is required so that the linear basis functions N_m can be used. Elimination of the second derivative terms from the volume integral also has the advantage that all impedance boundary conditions can be incorporated into the surface integral of Eq. (52). This allows a choice of basis functions that do not have to satisfy explicitly any impedance boundary conditions. The contribution to the surface integral

$$\int_S \frac{\partial p}{\partial n} N_m \, \mathrm{d}S \tag{53}$$

is identically zero for all elements except those that lie along an impedance boundary. Substituting the exit boundary condition [Eq. (47)] into the surface integral in Eq. (53) gives

$$\int_S \frac{\partial p}{\partial n} N_m \, \mathrm{d}S = -ik \int_S \frac{p}{\zeta_{\text{exit}}} N_m \, \mathrm{d}S \tag{54}$$

along the exit boundary, whereas for elements that lie along the upper, lower, and sidewalls of the duct

$$\int_S \frac{\partial p}{\partial n} N_m \, \mathrm{d}S = -ik \int_S \frac{p}{\zeta} N_m \, \mathrm{d}S \tag{55}$$

The contribution to the minimization of the field error for each element, when collected for each of the eight local nodes m, is expressed in matrix form as

$$\begin{Bmatrix} \int_V E_r N_1 \, \mathrm{d}V \\ \int_V E_r N_2 \, \mathrm{d}V \\ \vdots \\ \int_V E_r N_8 \, \mathrm{d}V \end{Bmatrix} = \left[A^{(I,J,K)}\right]\left\{\Phi^{(I,J,K)}\right\} \tag{56}$$

In Eq. (56), $\{\Phi^{(I,J,K)}\}$ is an 8×1 column vector for each element containing the unknown acoustic pressures at the eight local nodes of the element

$$\left\{\Phi^{(I,J,K)}\right\}^T = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\} \tag{57}$$

The element matrix $[A^{(I,J,K)}]$ is an 8×8 complex symmetric matrix for each element (I, J, K). In the special case of a hard wall duct ($\zeta = \infty$),

$$\left[A^{(I,J,K)}\right] = \begin{Bmatrix} [\mathcal{P}], & K \neq (NZ-1) \\ [\mathcal{P}] + [B], & K = (NZ-1) \end{Bmatrix} \tag{58}$$

Here, $[\mathcal{P}]$ represents the contribution to $[A^{(I,J,K)}]$ due to the element volume V, whereas $[B]$ represents the contributions due to the exit plane boundary. The matrices $[\mathcal{P}]$ and $[B]$ are symmetric, and their coefficients have been computed explicitly:

$$[\mathcal{P}] = \frac{k^2 whl}{216}[\mathcal{A}] - \frac{wl}{36h}[\mathcal{B}] - \frac{hl}{36w}[\mathcal{C}] - \frac{wh}{36l}[\mathcal{F}] \tag{59}$$

$$[\mathcal{A}] = \begin{bmatrix} 8 & 4 & 2 & 4 & 4 & 2 & 1 & 2 \\ 4 & 8 & 4 & 2 & 2 & 4 & 2 & 1 \\ 2 & & 8 & 4 & 1 & 2 & 4 & 2 \\ 4 & 2 & 4 & 8 & 2 & 1 & 2 & 4 \\ 4 & 2 & 1 & 2 & 8 & 4 & 2 & 4 \\ 2 & 4 & 2 & 1 & 4 & 8 & 2 & 4 \\ 1 & 2 & 4 & 2 & 2 & 2 & 8 & 4 \\ 2 & 1 & 2 & 4 & 4 & 4 & 4 & 8 \end{bmatrix}$$

$$[\mathcal{B}] = \begin{bmatrix} 4 & 2 & -2 & -4 & 2 & 1 & -1 & -2 \\ 2 & 4 & -4 & -2 & 1 & 2 & -2 & -1 \\ -2 & -4 & 4 & 2 & -1 & -2 & 2 & 1 \\ -4 & -2 & 2 & 4 & -2 & -1 & 1 & 2 \\ 2 & 1 & -1 & -2 & 4 & 2 & -2 & -4 \\ 1 & 2 & -2 & -1 & 2 & 4 & -4 & -2 \\ -1 & -2 & 2 & 1 & -2 & -4 & 4 & 2 \\ -2 & -1 & 1 & 2 & -4 & -2 & 2 & 4 \end{bmatrix} \tag{60}$$

$$[\mathcal{C}] = \begin{bmatrix} 4 & -4 & -2 & 2 & 2 & -2 & -1 & 1 \\ -4 & 4 & 2 & -2 & -2 & 2 & 1 & -1 \\ -2 & 2 & 4 & -4 & -1 & 1 & 2 & -2 \\ 2 & -2 & -4 & 4 & 1 & -1 & -2 & 2 \\ 2 & -2 & -1 & 1 & 4 & -4 & -2 & 2 \\ -2 & 2 & 1 & -1 & -4 & 4 & 2 & -2 \\ -1 & 1 & 2 & -2 & -2 & 2 & 4 & -4 \\ 1 & -1 & -2 & 2 & 2 & -2 & -4 & 4 \end{bmatrix} \tag{61}$$

$$[\mathcal{F}] = \begin{bmatrix} 4 & 2 & 1 & 2 & -4 & -2 & -1 & -2 \\ 2 & 4 & 2 & 1 & -2 & -4 & -2 & -1 \\ 1 & 2 & 4 & 2 & -1 & -2 & -4 & -2 \\ 2 & 1 & 2 & 4 & -2 & -1 & -2 & -4 \\ -4 & -2 & -1 & -2 & 4 & 2 & 1 & 2 \\ -2 & -4 & -2 & -1 & 2 & 4 & 2 & 1 \\ -1 & -2 & -4 & -2 & 1 & 2 & 4 & 2 \\ -2 & -1 & -2 & -4 & 2 & 1 & 2 & 4 \end{bmatrix} \tag{62}$$

$$[B] = -ik\frac{wh}{144}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & f_1 & f_2 & f_3 & f_4 \\ 0 & 0 & 0 & 0 & f_2 & f_5 & f_6 & f_3 \\ 0 & 0 & 0 & 0 & f_3 & f_6 & f_7 & f_8 \\ 0 & 0 & 0 & 0 & f_4 & f_3 & f_8 & f_9 \end{bmatrix} \tag{63}$$

$$\begin{aligned}
f_1 &= 9\beta_{\text{exit}}(x_I, y_J) + 3\beta_{\text{exit}}(x_I, y_{J+1}) \\
&\quad + \beta_{\text{exit}}(x_{I+1}, y_{J+1}) + 3\beta_{\text{exit}}(x_I, y_{J+1}) \\
f_2 &= 3\beta_{\text{exit}}(x_I, y_J) + \beta_{\text{exit}}(x_I, y_{J+1})] \\
&\quad + \beta_{\text{exit}}(x_{I+1}, y_{J+1}) + \beta_{\text{exit}}(x_I, y_{J+1}) \\
f_3 &= \beta_{\text{exit}}(x_I, y_J) + \beta_{\text{exit}}(x_I, y_{J+1}) \\
&\quad + \beta_{\text{exit}}(x_{I+1}, y_{J+1}) + \beta_{\text{exit}}(x_I, y_{J+1}) \\
f_4 &= 3\beta_{\text{exit}}(x_I, y_J) + \beta_{\text{exit}}(x_I, y_{J+1}) \\
&\quad + \beta_{\text{exit}}(x_{I+1}, y_{J+1}) + 3\beta_{\text{exit}}(x_I, y_{J+1}) \\
f_5 &= 3\beta_{\text{exit}}(x_I, y_J) + 9\beta_{\text{exit}}(x_I, y_{J+1}) \\
&\quad + 3\beta_{\text{exit}}(x_{I+1}, y_{J+1}) + \beta_{\text{exit}}(x_I, y_{J+1}) \\
f_6 &= \beta_{\text{exit}}(x_I, y_J) + 3\beta_{\text{exit}}(x_I, y_{J+1}) \\
&\quad + 3\beta_{\text{exit}}(x_{I+1}, y_{J+1}) + \beta_{\text{exit}}(x_I, y_{J+1}) \\
f_7 &= \beta_{\text{exit}}(x_I, y_J) + 3\beta_{\text{exit}}(x_I, y_{J+1}) \\
&\quad + 9\beta_{\text{exit}}(x_{I+1}, y_{J+1}) + 3\beta_{\text{exit}}(x_I, y_{J+1}) \\
f_8 &= \beta_{\text{exit}}(x_I, y_J) + \beta_{\text{exit}}(x_I, y_{J+1}) \\
&\quad + 3[\beta_{\text{exit}}(x_{I+1}, y_{J+1}) + \beta_{\text{exit}}(x_I, y_{J+1})] \\
f_9 &= 3\beta_{\text{exit}}(x_I, y_J) + \beta_{\text{exit}}(x_I, y_{J+1}) \\
&\quad + 3\beta_{\text{exit}}(x_{I+1}, y_{J+1}) + 9\beta_{\text{exit}}(x_I, y_{J+1})
\end{aligned} \tag{64}$$

in which

$$\beta_{\text{exit}} = 1/\zeta_{\text{exit}} \tag{65}$$

V. Results and Discussion

The three-dimensional rigid wall acoustic element has been coupled with the sparse assembly and equation solver algorithms to provide assembly and solver statistics for a three-dimensional duct aeroacoustics application. Computations presented in this paper were run on a single processor with double-precision (64-bit) arithmetic on an ORIGIN 2000 computer platform. The sparse equation solver used MMD reordering. Computations are presented for a uniform grid and a geometry identical to that of the Langley Flow Impedance Tube. This three-dimensional duct has a square cross section 0.0508 m in width ($W = H = 0.0508$ m) and 0.812 m in length ($L = 0.812$ m). A more detailed description of the duct is given in Ref. 15. All calculations were performed at standard atmospheric conditions without flow, and the source frequency was chosen to span the full range of frequencies currently of interest in duct liner research. The sound was chosen as a plane wave ($p_s = 1$), and the dimensionless exit impedance was chosen as unity ($\zeta_{\text{exit}} = 1$). This exit impedance will simulate a nonreflecting termination for the plane wave source.

Table 1 presents CPU statistics (in seconds) for each of the three assembly algorithms and the sparse equation solver as a function of the source frequency f, in kilohertz. The CPU time for the solver (column 9) is that required to obtain the solution vector after the system matrix was assembled. Note that before obtaining the solution vector, the system matrices obtained from each assembly algorithm were compared to each other. Each assembly algorithm assembled the identical system matrix as expected. Also included in Table 1 are the number of grid lines NX, NY, and NZ and the matrix order N that were used to perform the computations at each frequency. Here we have used the generally accepted rule that 12 points per wavelength is required to resolve a cut-on mode in each coordinate direction. To establish the accuracy of the solver solutions, the relative error norm (Relerr), computed from the solver solution vector, was tabulated in the final column of Table 1. The relative error norm[2] is defined as

$$\text{Relerr} = \frac{\{EN\}^* \times \{EN\}^T}{\{\bar{F}\}^* \times \{\bar{F}\}^T} \tag{66}$$

where

$$\{EN\} = [\bar{A}]\{\Phi\} - \{\bar{F}\} \tag{67}$$

Table 1 CPU time (in seconds) and error statistics for the sparse algorithms

f	NX	NY	NZ	N	Algorithm 1	Algorithm 2	Algorithm 3	Solver	Relerr
4.00	6	6	114	4,104	49.20	0.34	0.22	6.00	6.5×10^{-15}
7.00	12	12	200	28,800	106.80	2.50	1.75	22.80	1.8×10^{-12}
11.00	18	18	313	101,412	1,123.80	9.05	2.28	487.80	7.5×10^{-12}
14.00	24	24	399	229,824	5,520.60	20.74	14.24	3,120.00	3.4×10^{-11}
17.00	30	30	484	435,600	19,488.00	39.78	26.94	10,440.00	3.2×10^{-11}
21.00	36	36	599	776,304	N/A	73.81	48.35	N/A	N/A

Table 2 RAM statistics (in megabytes) for the sparse algorithms

f	N	$N1$	Algorithm 2	Algorithm 3	Solver
4.00	4,104	41,468	0.47	0.46	4.00
7.00	28,800	331,244	21.00	11.00	80.00
11.00	101,412	1,216,118	72.00	37.00	640.00
14.00	229,824	2,812,838	165.00	83.00	2,140.00
17.00	435,600	5,396,600	317.00	158.00	8,100.00
21.00	776,304	9,696,158	551.00	283.00	N/A

Tabular results at 21 kHz are not presented for assembly algorithm 1 and the sparse equation solver because of the excessive CPU time required by these two algorithms.

Although algorithm 1 is extremely simple, its performance is extremely slow (Table 1). Note that algorithm 1 is 145 times slower than the other two algorithms at a frequency of 4 kHz and more than 490 times slower at 17 kHz. Tabular results also show that the CPU time required to assemble the system matrix using algorithm 1 exceeds that required to obtain the solution vector by 9048 s (or 87%) at 17 kHz. At low frequencies, algorithm 2 is only slightly slower than algorithm 3, but as the frequency increases to 17 kHz, algorithm 3 is 32% faster than algorithm 2. Generally, the higher the frequency, the better the performance of algorithm 3, relative to that of algorithm 2. Furthermore, in using algorithm 2, the user has to guess the maximum number of nonzero terms per row (MZ) to allocate the RAM for the matrix $[HA]$. Also, the CPU times required to assemble the system matrix using algorithm 2 or algorithm 3 are both more than two orders of magnitude less than the time required to obtain the solution vector. Finally, Relerr is small, indicating that the solver solution is accurate.

Table 2 shows the RAM (in megabytes) for algorithm 2, algorithm 3, and the sparse equation solver. RAM statistics for algorithm 1 were not tabulated because its performance was extremely slow when compared to algorithm 2 and algorithm 3 (as shown in Table 1). Values of the variables N and $N1$ are also given in Table 2. The results show that the number of off-diagonal nonzero coefficients ($N1$) is an order of magnitude larger than N. Table 2 also shows that algorithm 3 requires less memory than algorithm 2 because algorithm 2 must allocate RAM for storing vectors $\{IR\}$, $\{JC\}$, and $[HA]$ [see Eqs. (13–15)]. Note also that memory required by the sparse equation solver is substantially larger than that required for assembly algorithm 2 or algorithm 3. This is further verification that most of the RAM allocated is used during matrix factorization. Preliminary results from tests conducted by the authors have suggested that the performance of the sparse equation solver may improve if the solver were to use METIS instead of MMD reordering. For example, at 7 kHz the number of nonzeros after factorization ($N2$) was reduced from 4,736,991 with MMD reordering to only 4,376,496 when the METIS reordering algorithm was used.

VI. Conclusions

A template for symmetric sparse equation assembly and solutions on an unstructured grid has been presented. The accuracy and numerical performance of the sparse algorithms have been evaluated over the frequency range of interest in a three-dimensional aeroacoustics application. Based on the results of this study, the following conclusions are drawn:

1) Assembly algorithm 1 is impractical for system matrix assembly at high values of source frequency. It requires up to 87% more CPU time to assemble the system matrix than the sparse equation solver requires to obtain the solution vector.

2) Assembly algorithms 2 and 3 have nearly equal performances at low values of source frequency, but algorithm 3 gives savings in both CPU time (32%) and RAM (50%) at the higher values of source frequency.

3) Error norm statistics show that the sparse equation solver computes accurate acoustic solutions over the frequency range of interest for the three-dimensional aeroacoustics application.

4) At high frequency (17 kHz), the sparse equation solver requires low memory, but requires significant speed-up before optimization studies (either of the duct geometry or liner material properties) are practical. This research supports a recommendation, therefore, that a parallel version of the sparse solver be developed. The CPU time and RAM required by assembly algorithms 2 and 3 are two orders of magnitude smaller than that required by the sparse equation solver. These algorithms can, therefore, be conveniently incorporated into a substructuring (or domain decomposition) formulation (provided that each substructure is handled by different processors) to take advantage of parallel computation to further reduce CPU time and RAM.

References

[1]Stead, D., "A Best Choice Numerical Method for Large-Scale Computations in Aeroacoustics," M.S. Thesis, Joint Inst. of Acoustics and Flight Science, George Washington Univ., Hampton, VA, June 1999.

[2]Nguyen, D., *Parallel-Vector Equation Solvers for Finite Element Engineering Applications*, Kluwer/Plenum, Norwell, MA, 2001, Chap. 10.

[3]Nguyen, D., Hou, G., Runesha, H., and Han, B., "Alternative Approach for Solving Sparse Indefinite Symmetrical System of Equations," *Advances in Engineering Software*, Vol. 31, Nos. 8–9, 2000, pp. 581–584.

[4]Chen, P., Runesha, H., Nguyen, D., Tong, P., and Chang, T., "Sparse Algorithms for Indefinite Systems of Linear Equations," *Computational Mechanics Journal*, Vol. 25, No. 1, 2000, pp. 33–41.

[5]George, A., and Liu, J., *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall, Englewood Cliffs, NJ, 1981, Chaps. 5 and 10.

[6]Pissanetzsky, S., "Gauss Elimination with Supersparse Matrices," Brookhaven National Lab., Rept. BNL 26773, Upton, NY, 1979.

[7]Simon, H., Vu, P., and Yang, C., "Performance of a Supernodal General Sparse Solver on the Cray-YMP: 1.68 GFLOPS with Autotasking," Applied Mathematics TR, Boeing Computer Services, SCA-TR-117, Seattle, WA, March 1989.

[8]Liu, J., "The Role of Elimination Trees in Sparse Factorization," *SIAM Journal of Matrix Analysis Application*, Vol. 11, No. 1, 1990, pp. 134–172.

[9]Duff, I., and Reid, J., "MA27-A Set of Fortran Subroutines for Solving Sparse Symmetric Sets of Linear Equations," Atomic Energy Research Establishment, TR R-10533, Harwell, England, U.K., 1982.

[10]Duff, I., and Reid, J., "The Multifrontal Solution of Indefinite Sparse Symmetric Linear Systems," *Association for Computing Machinery Transactions Mathematical Software*, Vol. 9, No. 3, 1983, pp. 302–325.

[11]Duff, I., Gould, N., Reid, J., Scott, J., and Turner, K., "Factorization of Sparse Symmetric Indefinite Matrices," *Institute of Mathematics and Its Applications Journal of Numerical Analysis*, Vol. 11, No. 9, 1991, pp. 181–204.

[12]Duff, I., Grimes, R., and Lewis, J., "Users' Guide for the Harwell–Boeing Sparse Matrix Collection (Release I)," Rutherford Appleton Lab., TR 92-086, Chilton, England, U.K., 1992.

[13]Duff, I., and Reid, J., "MAAT, a Fortran Code for Direct Solution of Indefinite Sparse Symmetric Linear Systems," Rutherford Appleton Lab., RAL-95-001, Chilton, England, U.K., Jan. 1995.

[14]Ng, E., and Peyton, B., "Block Sparse Choleski Algorithm on Advanced Uniprocessor Computer," *SIAM Journal of Scientific Computing*, Vol. 14, No. 5, 1993, pp. 1034–56.

[15]Watson, W., "Three-Dimensional Nacelle Aeroacoustic Code with Application to Impedance Eduction," AIAA Paper 2000-1956, June 2000.

[16]Chandrakant, S., and Abel, J., *Introduction to the Finite Element Method*, Van Nostrand Reinhold, New York, 1972, pp. 75–147.

[17]Ashcraft, C., "Compressed Graphs and the Minimum Degree Algorithm," *SIAM Journal of Scientific Computing*, Vol. 16, No. 6, 1995, pp. 1404–411.

[18]Gamma, E., Helm, R., Johnson, R., and Vlissides, J., "Design Patterns: Elements of Reusable Object-Oriented Software," *Addison Wesley Professional Computing Series*, Addison Wesley Longman, Reading, MA, 1995, pp. 25–30.

[19]George, J., and Liu, J., "The Evolution of the Minimum Degree Algorithm," *Society for Industrial and Applied Mathematics*, Vol. 31, No. 1, 1989, pp. 1–19.

[20]Liu, J., "Modification of the Minimum-Degree Algorithm by Multiple Elimination," *Association for Computing Machinery, Transactions on Mathematical Software*, Vol. 11, No. 2, 1985, pp. 141–53.

[21]Kumfert, G., and Pothen, A., "An Object-Oriented Collection of Minimum Degree Algorithms: Design, Implementation, and Experiences," NASA CR-1999-208977 1999; also Inst. for Computer Applications in Science and Engineering, Rept. 99-1, Hampton, VA, Jan. 1999.

[22]Amestoy, P., Davis, T., and Duff, I., "An Approximate Minimum Degree Ordering Algorithm," Computer and Information Science Dept., TR-94-039, Univ. of Florida, Gainesvine, FL, Dec. 1994.

[23]Karypis, G., and Kumar, V., "METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering," Ver. 2.0, Univ. of Minnesota, Minneapolis, MN, 1995.

[24]Runesha, H., and Nguyen, D., "Vectorized Sparse Unsymmetrical Equation Solver for Computational Mechanics," *Advances in Engineering Software*, Vol. 31, Nos. 8–9, 2000, pp. 563–570.

P. J. Morris
Associate Editor