NASA
Technical Memorandum 102735

AVSCOM
Technical Memorandum 90-B-017

High-Performance Equation Solvers and Their Impact on Finite Element Analysis

Eugene L. Poole
Awesome Computing Inc.
Hampton, Virginia

Norman F. Knight, Jr.
NASA Langley Research Center
Hampton, Virginia 23665

D. Dale Davis, Jr.
Aerostructures Directorate
U. S. Army Aviation Research and Technology Activity
Aviation Systems Command
Hampton, Virginia 23665

September 1990

NASA
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665-5225

US ARMY
AVIATION
SYSTEMS COMMAND
AVIATION R&T ACTIVITY

HIGH-PERFORMANCE EQUATION SOLVERS AND THEIR IMPACT ON FINITE ELEMENT ANALYSIS

Eugene L. Poole, Norman F. Knight, Jr. and D. Dale Davis, Jr.
NASA Langley Research Center
Mail Stop 244, NASA Langley Research Center, Hampton, VA 23665 USA

INTRODUCTION

Large-scale finite element analyses are commonly used in aerospace and other industries as part of the structural design and certification process. The computational cost of these analyses is most often dominated by the cost of solving the system of algebraic equations associated with the finite element model. New high-performance computer systems have become widely available, and new equation solvers (direct and iterative) have been developed (see Refs. 1-3) to exploit the vector capabilities of these high-performance computer systems. This paper describes the use of equation solvers in structural analysis and demonstrates the need for different types of equation solvers within a comprehensive structural analysis software system. All of the equation solvers compared in this work are incorporated in a large-scale structural analysis software system - the Computational Structural Mechanics (CSM) Testbed [4]. This feature is in contrast to many finite element software systems which provide only a single equation solver. The CSM Testbed facilitates the integration of new methods into a shared software system enabling researchers to test these new methods by solving real applications problems and immediately providing structural analysts the benefits of improved problem-solving capabilities. A variable-band Choleski solver, a sparse Choleski solver, and two preconditioned conjugate gradient solvers are compared by solving several representative structural analysis problems. The resulting CPU time and memory requirements demonstrate the importance of selecting an appropriate equation solver for each problem.

FINITE ELEMENT ANALYSIS

High-performance equation solvers are a key component of solution strategies for linear and nonlinear structural response calculations for static, dynamic, and eigenvalue problems in finite element analysis. The semi-discrete equations of motion for time $t + \Delta t$ may be written as

$$\mathbf{M}\ddot{\mathbf{u}}_{t+\Delta t} + \mathbf{C}\dot{\mathbf{u}}_{t+\Delta t} + \mathbf{f}^{int}(\mathbf{u}_{t+\Delta t}) = \mathbf{f}^{ext}_{t+\Delta t} \qquad (1)$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ the damping matrix, $\mathbf{f}^{int}$ and $\mathbf{f}^{ext}$ the internal and external force vectors, and $\ddot{\mathbf{u}}$, $\dot{\mathbf{u}}$ and $\mathbf{u}$ the acceleration, velocity and displacement vectors. The internal force vector is a function of the displacements at time $t + \Delta t$ and may also be written as

$$\mathbf{f}^{int}(\mathbf{u}_{t+\Delta t}) = \mathbf{K}_0\, \mathbf{u}_{t+\Delta t} + \mathbf{q}(\mathbf{u}_{t+\Delta t}) \qquad (2)$$

where $\mathbf{K}_0$ is the linear stiffness matrix, and $\mathbf{q}$ is the vector of nonlinear terms.

The time dependent equations (1) may be solved using a number of different algorithms. One such algorithm is the Newmark implicit direct integration algorithm, given by

$$\mathbf{u}_{t+\Delta t} = \mathbf{u}_t + \Delta t\dot{\mathbf{u}}_t + (\Delta t)^2\Big[\Big(\frac{1}{2}-\beta\Big)\ddot{\mathbf{u}}_t + \beta\ddot{\mathbf{u}}_{t+\Delta t}\Big] \tag{3a}$$

$$\dot{\mathbf{u}}_{t+\Delta t} = \dot{\mathbf{u}}_t + \Delta t[(1-\alpha)\ddot{\mathbf{u}}_t + \alpha\ddot{\mathbf{u}}_{t+\Delta t}] \tag{3b}$$

This algorithm may be used to solve for the acceleration and velocity vectors, $\ddot{\mathbf{u}}$ and $\dot{\mathbf{u}}$, at time $t+\Delta t$ using the displacements, $\mathbf{u}$, at time $t+\Delta t$, and the solution vectors, $\ddot{\mathbf{u}}$, $\dot{\mathbf{u}}$ and $\mathbf{u}$, from the previous time step, *i.e.*

$$\ddot{\mathbf{u}}_{t+\Delta t} = \frac{1}{\beta(\Delta t)^2}(\mathbf{u}_{t+\Delta t}-\mathbf{u}_t) - \frac{1}{\beta\Delta t}\dot{\mathbf{u}}_t - (\frac{1}{2\beta}-1)\ddot{\mathbf{u}}_t \tag{4a}$$

$$\dot{\mathbf{u}}_{t+\Delta t} = \frac{\alpha}{\beta\Delta t}(\mathbf{u}_{t+\Delta t}-\mathbf{u}_t) - (\frac{\alpha}{\beta}-1)\dot{\mathbf{u}}_t - (\frac{\alpha}{2\beta}-1)\Delta t\ddot{\mathbf{u}}_t \tag{4b}$$

Substituting equations (4) into (1), the equations of motion may be written in terms of the displacements at time $t+\Delta t$. That is,

$$\begin{aligned}\mathbf{f}^{int}(\mathbf{u}_{t+\Delta t}) + \Big(\frac{1}{\beta(\Delta t)^2}\mathbf{M} + \frac{\alpha}{\beta\Delta t}\mathbf{C}\Big)\mathbf{u}_{t+\Delta t} = \\ \mathbf{f}^{ext}_{t+\Delta t} + \mathbf{M}\Big(\frac{1}{\beta(\Delta t)^2}\mathbf{u}_t + \frac{1}{\beta\Delta t}\dot{\mathbf{u}}_t + (\frac{1}{2\beta}-1)\ddot{\mathbf{u}}_t\Big) \\ + \mathbf{C}\Big(\frac{\alpha}{\beta\Delta t}\mathbf{u}_t + (\frac{\alpha}{\beta}-1)\dot{\mathbf{u}}_t + (\frac{\alpha}{2\beta}-1)\Delta t\ddot{\mathbf{u}}_t\Big)\end{aligned} \tag{5}$$

The terms on the right-hand side of equation (5) are independent of the displacement vector at time $t+\Delta t$. The system of equations (5) may involve nonlinear terms through the internal force vector $\mathbf{f}^{int}(\mathbf{u}_{t+\Delta t})$. The nonlinear system to be solved at time $t+\Delta t$ is represented by

$$\overline{\mathbf{F}}_{t+\Delta t} = \mathbf{f}^{int}(\mathbf{u}_{t+\Delta t}) + \Big(\frac{1}{\beta(\Delta t)^2}\mathbf{M} + \frac{\alpha}{\beta\Delta t}\mathbf{C}\Big)\mathbf{u}_{t+\Delta t} - \mathbf{F}_{t+\Delta t} = 0 \tag{6a}$$

with

$$\begin{aligned}\mathbf{F}_{t+\Delta t} = \mathbf{f}^{ext}_{t+\Delta t} + \mathbf{M}\Big(\frac{1}{\beta(\Delta t)^2}\mathbf{u}_t + (\frac{1}{2\beta}-1)\ddot{\mathbf{u}}_t\Big) \\ + \mathbf{C}\Big(\frac{\alpha}{\beta\Delta t}\mathbf{u}_t + (\frac{\alpha}{\beta}-1)\dot{\mathbf{u}}_t + (\frac{\alpha}{2\beta}-1)\Delta t\ddot{\mathbf{u}}_t\Big)\end{aligned} \tag{6b}$$

This nonlinear system of equations may be solved using the Newton-Raphson procedure at time $t+\Delta t$. For iteration $k+1$, the procedure used to solve the linearized system of equations follows:

$$\overline{\mathbf{K}}^k_{t+\Delta t}\ \Delta\mathbf{u}^{k+1}_{t+\Delta t} = -\overline{\mathbf{F}}^k_{t+\Delta t} \tag{7a}$$

$$\mathbf{u}^{k+1}_{t+\Delta t} = \mathbf{u}^k_{t+\Delta t} + \Delta\mathbf{u}^{k+1}_{t+\Delta t} \tag{7b}$$

The Jacobian, or effective tangent stiffness matrix, associated with the k^{th} iterate, $\mathbf{u}^k_{t+\Delta t}$, is given by

$$\overline{\mathbf{K}}^k_{t+\Delta t} = \left(\frac{\partial \mathbf{f}^{int}}{\partial \mathbf{u}}\right)^k_{t+\Delta t} + \frac{1}{\beta(\Delta t)^2}\mathbf{M} + \frac{\alpha}{\beta \Delta t}\mathbf{C} \tag{7c}$$

Once the displacement vector at time $t + \Delta t$ is obtained, the velocity and acceleration vectors at time $t + \Delta t$ may be readily obtained using equations (4).

For a nonlinear transient dynamic response, the effective tangent stiffness matrix, $\overline{\mathbf{K}}^k_{t+\Delta t}$, must be factored repeatedly. The number of factorizations may be reduced if a modified Newton-Raphson procedure is used. For a linear transient dynamic response, the left-hand-side of equation (5) only involves matrices of constant coefficients (*e.g.*, $\mathbf{K}_0$, $\mathbf{M}$, $\mathbf{C}$). Only one factorization is required if the time step size Δt is held constant. As such, the transient response is obtained by repeated back-substitutions and evaluations of the right-hand-side vector.

For static analysis problems, the inertia and damping terms are neglected in equation(5), and static equilibrium corresponds to the balance of internal and external forces. For a linear static response, the internal force vector is just the product of the linear stiffness matrix $\mathbf{K}_0$ and the displacement vector $\mathbf{u}$, and only a single factorization is required. For a nonlinear static response, a Newton-Raphson procedure (like equations (7)) is used and the tangent stiffness matrix ($\mathbf{K}_T$), given by

$$\left(\mathbf{K}_T\right)^k_{t+\Delta t} = \left(\frac{\partial \mathbf{f}^{int}}{\partial \mathbf{u}}\right)^k_{t+\Delta t} = \mathbf{K}_0 + \left(\frac{\partial \mathbf{q}}{\partial \mathbf{u}}\right)^k_{t+\Delta t} \tag{8}$$

may need to be factored repeatedly.

For structural eigenvalue problems (buckling or vibration analyses), a generalized eigenvalue problem of the form

$$\mathbf{A}\mathbf{x}_i - \lambda_i \mathbf{B}\mathbf{x}_i = 0 \tag{9}$$

must be solved. In linear vibration analyses, the matrices $\mathbf{A}$ and $\mathbf{B}$ correspond to the linear stiffness matrix $\mathbf{K}_0$ and the mass matrix $\mathbf{M}$, respectively, λ_i is the ith vibration frequency squared, and $\mathbf{x}_i$ is the corresponding ith vibration mode shape. This generalized eigenvalue problem is often transformed to a standard eigenvalue problem of the form

$$\tilde{\mathbf{A}}\tilde{\mathbf{x}}_i - \lambda_i \tilde{\mathbf{x}}_i = 0 \tag{9}$$

where $\tilde{\mathbf{A}} = \mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}$ using the Cholesky factors of $\mathbf{B} = \mathbf{L}\mathbf{L}^T$ or of a shifted matrix of the form $\mathbf{A} - \sigma\mathbf{B} = \mathbf{L}\mathbf{L}^T$.

FEATURES OF HIGH-PERFORMANCE COMPUTERS

Modern high-performance vector computers, such as the Convex C220, the CRAY-2, and the CRAY Y-MP, have from two to eight central processing units (CPU's). Each CPU has multiple vector arithmetic and logic functional units which access very large main memories through high-speed vector registers. The computation rate, commonly measured in units known as MFLOPS (millions of floating point operations per second), is maximum when both addition and multiplication vector functional units are operating simultaneously, thus producing two results every machine cycle. The actual computation rate achieved is often much lower than this theoretical peak due to several factors, the most important being the time required for memory access. Some delay is always incurred if the operands for a vector addition or multiplication must be transferred from main memory to the vector registers. In general, the rate of this transfer is maximum when array elements are stored contiguously in main memory. Once elements are in the vector registers they can be accessed at the maximum computation rate. Another factor which affects the computation rate is the type of operations required to carry out an algorithm. For example, vector SAXPY operations (*scalar* $\times$ *vector* + *vector*) are efficient on vector computers since they contain both addition and multiplication instructions which can be carried out nearly simultaneously. However, inner product instructions are somewhat less efficient since they require a summation of many vector elements into a single scalar value. The challenge to the software developer is to design algorithms which minimize the memory access delays while utilizing the full computing power of multiple vector functional units.

In addition to maximizing the vectorization capabilities on a single CPU, parallel processing is possible on multiple processor computers. Parallel processing in combination with highly vectorized algorithms can lead to impressive performance rates of over 1 billion operations per second (see Ref. 5). The potential benefit of parallel-vector algorithms on today's supercomputers is realized when multiple processors are available for use by a single user but this availability is often limited in current multi-user environments. Additionally, large comprehensive structural analysis software systems are not currently available for massively parallel computers like the hypercube and Connection Machine computers. Therefore, although parallelization of the solvers presented in this paper is possible, and of interest, this paper focuses on vectorized methods for a single CPU.

DESCRIPTION OF EQUATION SOLVERS

The equation solvers described in this paper are used to solve symmetric, positive definite linear (or linearized) systems of equations of the form given by equations (2a). The direct solvers are Choleski methods which consist of a factorization of the matrix into triangular factors ($\mathbf{LL}^T$), followed by the forward and backward solution of the resulting triangular systems. The iterative solvers are preconditioned conjugate gradient (PCG) methods which proceed from an initial guess, $\mathbf{u}^0$, for the solution of (2a) and, through an iterative process, refine the guess to a very close approximation, $\mathbf{u}^k$, of the exact solution.

Variable-Band Choleski Method

The factorization is by far the most computationally intensive portion of the Choleski method. Many algorithms have been developed for this method differing both in the order in which computations are carried out and in the scheme used to store the matrix coefficients. A thorough description of some implementations for vector and parallel computers is given in Ref. 6. The factorization algorithm used for this work, hereafter referred to as the variable-band method, is described in detail in Ref. 1 and is illustrated in Figure 1 for a matrix with n columns.

```
Loop1 $k = 1$ to $n - r - 1$ in steps of $r$
    Complete columns $k, k+1, ..., k+r-1$ of $\mathbf{L}$
    $lastrk = k + \text{len}(k) - 1$
    Loop2 $j = k + r$ to $lastrk$
      IF $\mathbf{L}_{j,k}, ..., \mathbf{L}_{j,k+r-1}$ are not all zero THEN
          Update column $j$ of $\mathbf{K}$ using $r$ columns of $\mathbf{L}$
          Loop3 $i = j$ to $lastrk$
              $\mathbf{K}_{i,j} = \mathbf{K}_{i,j} - \mathbf{L}_{i,k} * \mathbf{L}_{j,k} - \mathbf{L}_{i,k+1} * \mathbf{L}_{j,k+1} -$
                  $\ldots - \mathbf{L}_{i,k+r-1} * \mathbf{L}_{j,k+r-1}$
          EndLoop3
      ENDIF
    EndLoop2
EndLoop1
Finish any remaining columns of $\mathbf{L}$
```

Figure 1. Variable-Band Choleski Factorization for $n \times n$ Matrix,

There are two key features of this method - a novel variable-band data storage scheme and an implementation which computes multiple SAXPY operations in the innermost computation loop (*i.e*, loop unrolling). Loop 3 in the algorithm accounts for most of the computations and, by using r (typically, $r = 6$) SAXPY operations at a time in this loop, very efficient use of the multiple functional units is achieved. This implementation is in contrast to traditional skyline or profile solvers which use slower inner product operations. In addition, memory access delays are minimized by storing the matrix $\mathbf{L}$ by columns rather than by rows. This storage scheme allows the data to be accessed at the maximum rate and leaves each column of $\mathbf{K}$ that is updated in loop 3 in the vector registers for r SAXPY operations. Furthermore, the r columns of $\mathbf{L}$ that are computed in the outer loop are used many times and can therefore take advantage of fast local memory caches. The lower triangular part of $\mathbf{K}$ is stored by variable-length columns and during factorization $\mathbf{K}$ is modified and replaced by the matrix $\mathbf{L}$. Adjustments are made to the lengths of the columns of $\mathbf{K}$ to account for fill-in of coefficients during factorization and to insure that groups of r columns end in the same row. This storage scheme can require more total storage and operations than traditional skyline Choleski solvers but most of the extra operations are eliminated by skipping loop 3 whenever all r scalar multipliers are zero.

Since the SAXPY updates performed in loop 3 are all independent, each column update performed in loop 2 may be performed in parallel requiring only that n/r synchronization steps (one for each iteration of loop 1) are carried out. The synchronization step is required to insure that the r columns of $\mathbf{L}$, which are used for the column updates in loop 3, are finished. This minimal synchronization cost coupled with the high degree of vectorization for this variable-band algorithm makes very high computation rates possible on parallel-vector computer systems.

Sparse Choleski Method

In recent years much research has been devoted towards the development of Choleski algorithms which use sparse matrix storage schemes. The key feature of sparse methods is the emphasis on significantly reducing both the number of operations and the memory storage requirements for the factorization of $\mathbf{K}$. Sparse methods require sophisticated preprocessing and usually reorder the equations to minimize the fill-in of non-zero terms during factorization. On vector computers, the time required for this stage can be a significant portion of the time required to actually form matrix $\mathbf{L}$. In addition, the storage schemes used for these methods generally require some form of indirect addressing which causes significant memory access delays on vector computers. However, many advances have been made in the development of efficient vectorized sparse Choleski solvers (see Refs. 3,7) and these solvers must be considered for structural analysis problems. A sparse Choleski solver, which is part of the SPARSPAK software package, has been added to the CSM Testbed (Ref. 7). Compiler vectorization directives that were added to the SPARSPAK factorization algorithm used in the CSM Testbed improved the computation rate significantly.

```
Choose $\mathbf{u}^0$
Set $\mathbf{r}^0 = \mathbf{f} - \mathbf{K}\mathbf{u}^0$
Solve $\mathbf{M}\mathbf{q}^0 = \mathbf{r}^0$
Set $\mathbf{p}^0 = \mathbf{q}^0$
Loop $k = 0, 1, \ldots$
    $\alpha_k = -(\mathbf{r}^k, \mathbf{q}^k)/(\mathbf{p}^k, \mathbf{K}\mathbf{p}^k)$
    $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \mathbf{p}^k$
    $\mathbf{r}^{k+1} = \mathbf{r}^k + \alpha_k \mathbf{K}\mathbf{p}^k$
    Solve $\mathbf{M}\mathbf{q}^{k+1} = \mathbf{r}^{k+1}$
    Test for convergence
    $\beta_k = (\mathbf{r}^{k+1}, \mathbf{q}^{k+1})/(\mathbf{r}^k, \mathbf{q}^k)$
    $\mathbf{p}^{k+1} = \mathbf{q}^{k+1} + \beta_k \mathbf{p}^k$
Endloop
```

Figure 2. Preconditioned Conjugate Gradient Method

Iterative Methods

Iterative methods are attractive for many structural analysis problems. Since they do not require the expensive factorization of matrix $\mathbf{K}$, they require less storage than sparse or banded direct solvers. In addition, if a good approximate solution is available, iterative solvers may converge quickly to an accurate solution in far fewer numerical computations than required by direct solvers. A major disadvantage of using iterative solvers for structural analysis is that the iterative method may not always converge to the solution or may require far too many iterations. In addition, for their effective use, user interaction is often required. The iterative methods considered in this paper are preconditioned conjugate gradient (PCG) methods which have been found to be robust in many structural analysis problems (see Refs. 1,8).

The basic PCG method is shown in Figure 2, with the notation (**a**,**b**) denoting the inner product of two vectors, **a** and **b**. The major computation steps for PCG methods at each iteration are a matrix-vector multiplication, three SAXPY operations, two inner products and the preconditioning step, (solve $\mathbf{M}\mathbf{q}^{k+1} = \mathbf{r}^{k+1}$). One of the simplest, yet often effective, preconditioning strategies is symmetric diagonal scaling of matrix **K**. This method is often referred to as Jacobi conjugate gradient (JCG) and is implemented in the CSM Testbed. For the JCG method, vectorizing the matrix-vector multiplication is a primary consideration for an efficient implementation. In the CSM Testbed JCG method, an efficient implementation is achieved by using a general sparse diagonal storage scheme for matrix **K** which increases vector lengths and eliminates the need for expensive indirect addressing.

When the convergence rate of the JCG method is slow or when the method will not converge at all, a modified incomplete Choleski conjugate gradient (ICCG) method may be used. The basic ICCG method performs an incomplete Choleski factorization of matrix **K** once to form the preconditioning matrix $\mathbf{M} = \mathbf{L}\mathbf{L}^T$. An important modification to the basic incomplete factorization used in the CSM Testbed ICCG method is the multiplication of the diagonal of matrix **K** by $1+\alpha$ prior to factorization, where α is a small positive constant which is increased automatically by the algorithm to insure that matrix **L** remains positive definite (see Ref. 8). The preconditioning step at each iteration requires the solution of two sparse triangular systems. The lower triangular, non-zero coefficients of **K** and **L** are stored by columns along with row index values for each non-zero coefficient. The incomplete factorization, matrix-vector multiplications, and sparse triangular solves are vectorized. However, the computation rate is much lower than that of the variable-band solver on vector computers due both to the indirect addressing required for the sparse data storage scheme and to short vector lengths (between 20 and 40 coefficients for typical structures matrices).

DESCRIPTION OF EXAMPLE PROBLEMS

These high-performance equation solvers are implemented within the CSM Testbed (Ref. 4) and are readily available to structural analysts. Several representative structural analysis problems are considered herein to assess the performance of these equation solvers. While the structural response determination is the primary goal of structural analysis, the goal of these studies is to assess the relative performance of these equation solvers. Hence, only a brief description of each structural problem is provided herein.

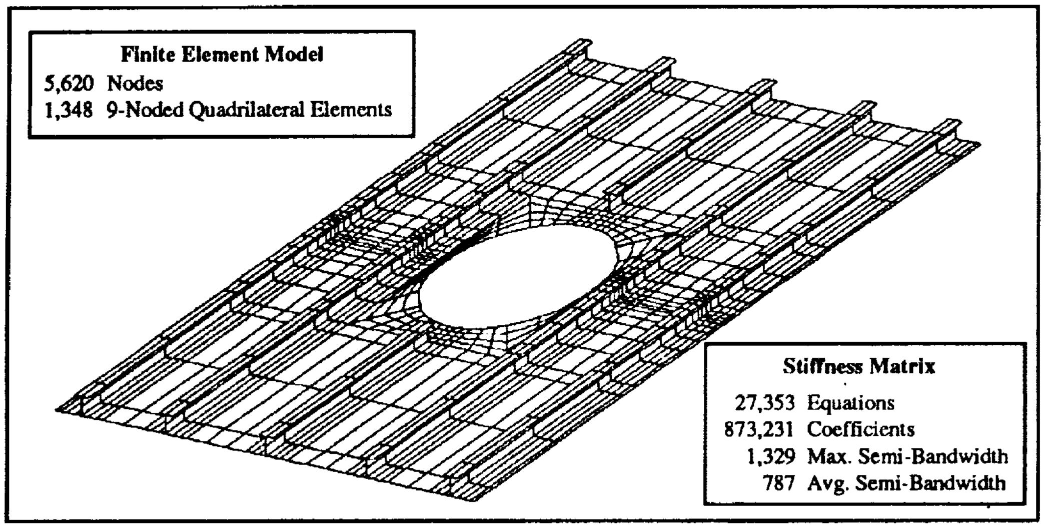

Figure 3. Composite Stiffened Panel with a Cutout

Composite Stiffened Panel with a Cutout

Predicting the structural response of aerospace structures in the presence of discontinuities, eccentricities, and damage is particularly difficult when the structural component is built from composite materials or is loaded into the nonlinear range. One such component is a flat, I-stiffened, composite panel with an elliptical cutout loaded in axial compression. Two-dimensional finite elements were used to model the entire structure (panel skin and stiffeners) and near discontinuities, the finite element mesh is refined. The finite element model, shown in Figure 3 has 5620 nodes and 1348 nine-noded quadrilateral shell elements. For the purposes of this paper, the linear static response was determined for the case of uniform axial compression.

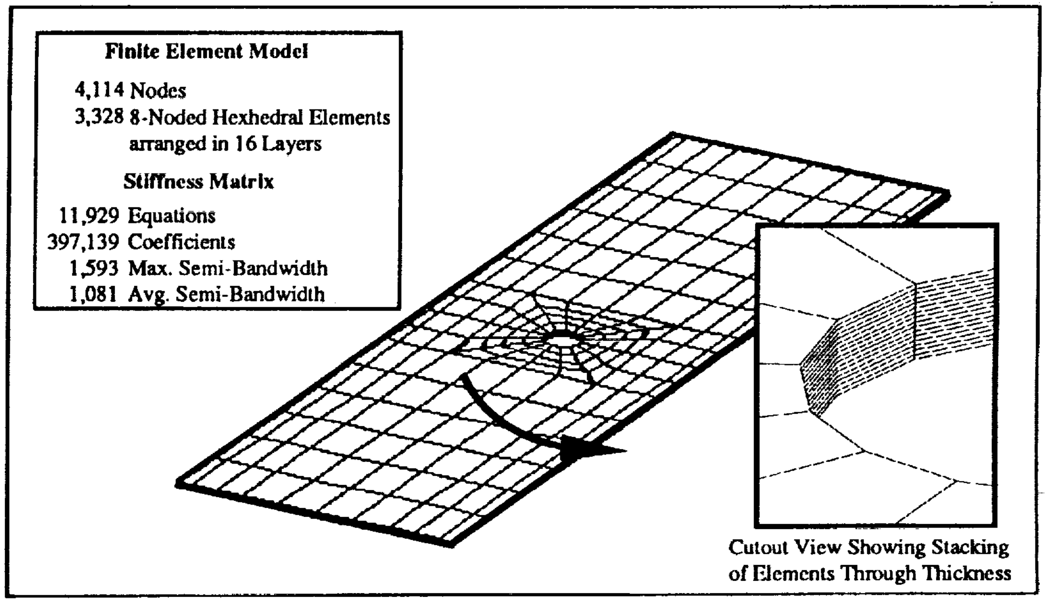
Finite Element Model
4,114 Nodes
3,328 8-Noded Hexhedral Elements arranged in 16 Layers
Stiffness Matrix
11,929 Equations
397,139 Coefficients
1,593 Max. Semi-Bandwidth
1,081 Avg. Semi-Bandwidth
Cutout View Showing Stacking of Elements Through Thickness

Figure 4. Cross-Ply Laminate with a Hole

Cross-Ply Composite Laminate with a Hole

Detailed stress analysis of composite structures is often required to determine accurate through-the-thickness (or interlaminar) stress distributions. Some sources of interlaminar stress gradients include free-edge effects, holes, and ply drop-offs (*e.g.*, tapered stiffener attachment flanges). To study these effects, three-dimensional finite element models are frequently used. To study the performance of these solvers for three-dimensional finite element models, the overall structural response of an 8-ply cross-ply composite laminate with a central circular hole is considered. The finite element model, shown in Figure 4, has 4114 nodes and 3328 eight-noded hexahedral solid elements wherein two elements through-the-thickness of each ply of composite material are used. This finite element model is adequate for overall response characteristics but must be refined in order to determine accurate interlaminar stress states. However, the stiffness matrix corresponding to this finite element model is characteristic of larger discrete models, and therefore useful for examining the relative performance of different equation solvers.

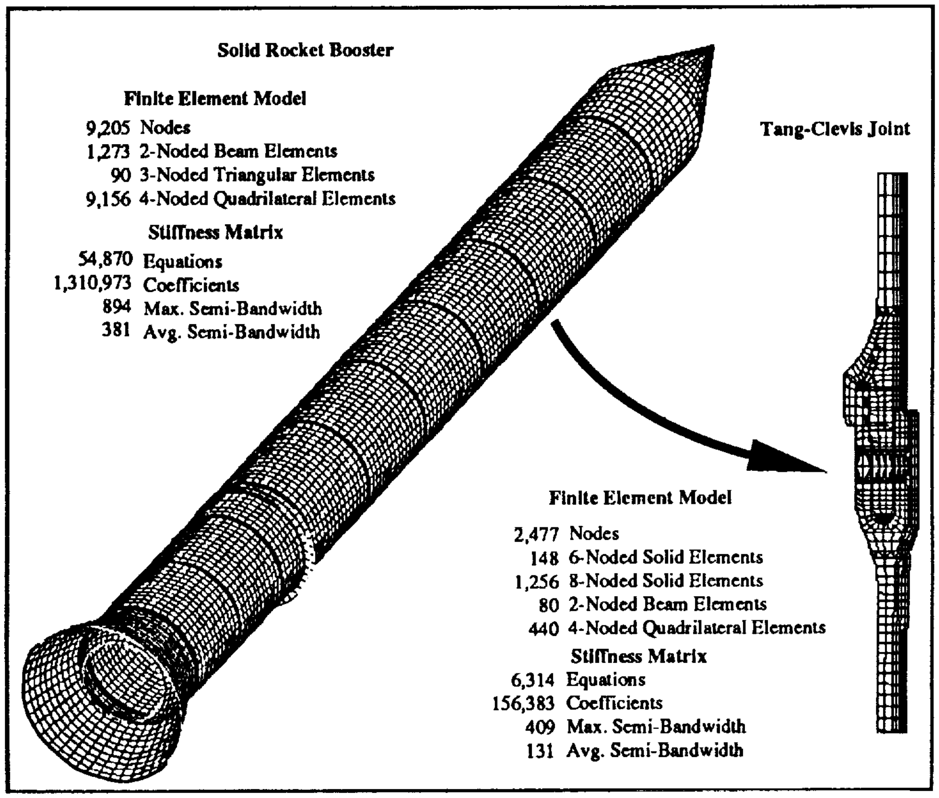

Figure 5. Space Shuttle Solid Rocket Booster and Tang-Clevis Joint

Space Shuttle Solid Rocket Motor Tang-Clevis Joint

The Space Shuttle Challenger accident investigation focused on the failure of a tang-clevis joint on the right Solid Rocket Motor (SRM). Finite element structural analyses were performed to predict both deflections and stresses in the joint under the primary pressure loading condition. The finite element model of the redesigned SRM tang-clevis joint, shown in Figure 5, has 2477 nodes, 148 six-noded solid elements, 1256 eight-noded solid elements, 80 two-noded beam elements, 440 four-noded quadrilateral elements, and 142 contact points (71 nonlinear spring elements). As part of a substructure approach based on a "unit motion" solution technique (see Ref. 9), the global system of equations must be solved repeatedly to obtain a displacement field for the unit motion applied at each contact point of the nonlinear spring elements (a total of 142 solutions). Each solution represents the displacement field corresponding to a single imposed unit motion at a specific contact point. This example problem differs from the other example problems in that even though 142 solutions are required only a single matrix decomposition is needed. This example problem illustrates the need for examining both the matrix decomposition stage and the forward-reduction/ back-substitution stage of direct equation solvers in assessing the relative performance of different equation solvers.

Space Shuttle Solid Rocket Booster

A preliminary assessment of the Space Shuttle Solid Rocket Booster (SRB) global shell response to selected prelaunch loads was presented in Reference 10. The two-dimensional shell finite element model used in this study was translated into a format compatible with the CSM Testbed. The finite element model, also shown in Figure 5, involves 9205 nodes with 1273 two-noded beam elements, 90 three-noded triangular elements, and 9156 four-noded quadrilateral elements. For the purpose of this paper, only a linear static analysis was performed for the internal SRM pressure loading case.

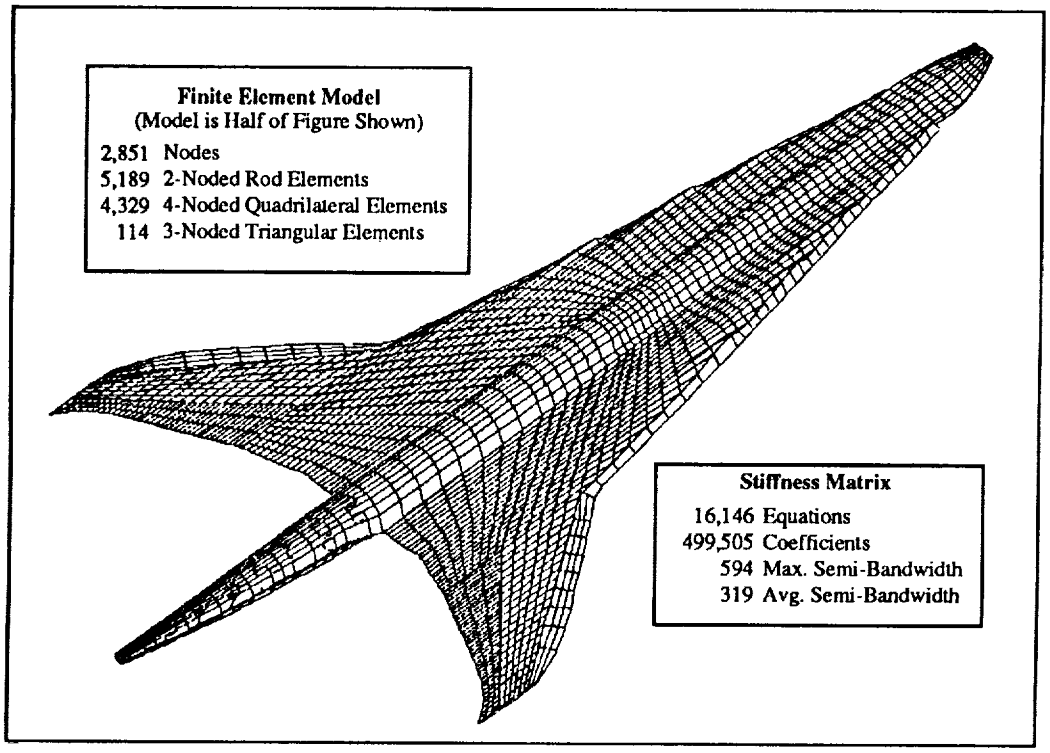

Figure 6. High-Speed Civil Transport Aircraft

High-Speed Civil Transport Aircraft

Projected trends in travel to the Pacific Basin has led to a renewed national interest in the development of a high-speed civil transport (HSCT) aircraft (see Ref. 11). The design of such a vehicle will require an integrated analysis approach involving both structures and aerodynamics. To accelerate the development of mathematical models of various structural configurations, a parameterized finite element model which represents the symmetric half of the HSCT aircraft has been developed (see Ref. 12). The finite element model for the entire HSCT aircraft is shown in Figure 6. The symmetric half-model involves 2851 nodes, 5189 two-noded rod elements, 4329 four-noded quadrilateral elements and 114 three-noded triangular elements. A linear static analysis is performed for the case of a wingtip loading.

NUMERICAL RESULTS

In this section numerical results are given comparing the runtimes (measured by CPU seconds), the number of operations (measured by the total number of additions and multiplications), the computation rate (measured by MFLOPS), and the memory requirements (measured by the number of 64-bit words required for matrix storage) for the direct and iterative solvers. A Convex C220 minisupercomputer was used for three analyses (see Table 1) and the NASA Langley CRAY-2 was used for three analyses (see Table 2). All analyses were performed using the CSM Testbed to generate the finite element models, solve the linear systems, and postprocess the output data. Times shown for the sparse solver, SPK, are given with and without the overhead costs associated with the use of the SPARSPAK software. Since most sparse solvers require extensive pre-processing which includes the input of the non-zero structure of the matrix, reordering of the equations, and the input of the actual matrix coefficients, this time is part of the total solution time. The data presented in Tables 1-3 also include normalized values which make comparisons between the solvers more clear.

The results in Tables 1 and 2 show that both the number of computations and the rate at which the computations can be performed must be considered to determine the fastest solver for a given problem. The variable-band solver has the highest computation rate for all problems considered herein and also has the lowest CPU time for most of the problems. This solver is the most effective at exploiting vector computer architectures, particularly on CRAY-2 computers, where rates as high as 269 MFLOPS were obtained. However, on all problems, the variable-band solver requires the most memory of all the solvers compared and always requires more operations than the sparse solver.

The effectiveness of the sparse solver in reducing the number of operations relative to the variable-band solver varied widely for the problems considered. For the composite stiffened panel problem (shown in Table 1), the sparse solver required over 17 times fewer operations than the banded solver. However, on the remaining problems the sparse solver required typically less than 2 times fewer operations than the banded solver. The computation rate for the sparse solver is much lower than the variable-band solver, although much faster rates have been reported for sparse solvers which use special machine coded routines. For example, Ref.3 reports a time of 23 seconds for the factorization only, of the matrix used in the Space Shuttle SRB problem on a CRAY Y-MP. However, this solver is not currently available for other computers such as the CONVEX C220 and relies on special machine-coded routines to improve the computation rate, while the variable-band solver uses a

Table 1. Comparison of Equation Solvers for Example Problems
Runs Made on Convex C220 Computer, normalized data in [ ]

Equation Solver	Time (sec)	Operations (adds/multiplies)	Rate (MFLOPS)	Memory (64-bit Words)
Stiffened Panel with Cutout, 9-Noded Shell Elements, 27,353 Equations 873,231 Coefficients, Max. Semi-Bandwidth=1,329, Avg. Semi-Bandwidth=787				
BAND	436.8 [1.9]	8,701,085,825 [17.5]	20.0 [9.5]	21,527,763 [8.1]
SPK: factor/solve	97.3 [.4]	496,998,128 [1.0]	5.1 [2.4]	3,684,396 [1.4]
with overhead	231.7 [1.0]	496,998,128 [1.0]	2.1 [1.0]	3,684,396 [1.4]
ICCG (1,041) γ=.067	1295.2 [5.6]	7,709,129,853 [15.5]	6.0 [2.9]	2,647,046 [1.0]
Cross-Ply Composite Laminate with a Hole, 8-Noded Brick Elements, 11,929 Equations 397,139 Coefficients, Max. Semi-Bandwidth=1,593, Avg. Semi-Bandwidth=1,081				
BAND	506.3 [1.0]	13,150,259,538 [4.2]	26.0 [3.9]	12,901,825 [10.7]
SPK: factor/solve	1085.7 [2.2]	7,832,183,056 [2.5]	7.2 [1.1]	7,518,911 [6.2]
with overhead	1192.2 [2.4]	7,832,183,056 [2.5]	6.6 [1.0]	7,518,911 [6.2]
ICCG (948) γ=.02	488.2 [1.0]	3,137,874,415 [1.0]	6.6 [1.0]	1,203,346 [1.0]
Space Shuttle SRM Joint with 142 R.H.S. Vectors, Mixed Element Types, 6,314 Equations 156,383 Coefficients, Max. Semi-Bandwidth=409, Avg. Semi-Bandwidth=131				
BAND	30.2 [1.0]	397,063,697 [1.3]	13.1 [3.2]	830,505 [1.7]
SPK: factor/solve	57.8 [1.9]	310,510,946 [1.0]	5.4 [1.3]	669,731 [1.5]
with overhead	76.4 [2.5]	310,510,946 [1.0]	4.1 [1.0]	669,731 [1.5]
ICCG (14,253) $\alpha = 0$	3578.7 [118.5]	18,918,520,552 [60.9]	5.3 [1.3]	475,463 [1.0]

BAND -	Choleski, variable-band matrix storage, loop unrolling to level 6, uses zero-checking option, CRAY-2 version exploits its local memory
SPK -	Choleski, sparse matrix storage, SPARSPAK-A with vectorization directives added, with overhead time (includes time to input non-zero structure, reorder equations and input non-zero coefficients)
ICCG -	Incomplete Choleski Preconditioned Conjugate Gradient, sparse matrix storage, diagonal multiplied by $1 + \gamma$ prior to incomplete decomposition, number of iterations in parenthesis

standard FORTRAN routine. Higher computation rates for sparse solvers relative to the variable-band solver would make them faster for some of the problems in this study.

The performance of the iterative solvers on both computers was limited by both a low computation rate and a high operation count (due to slow convergence rates) for most of the example problems. The solution was considered converged if the inner product in Figure 2 was less than a specified tolerance: $(\mathbf{r}^{k+1}, \mathbf{q}^{k+1}) < 10^{-10} \times n$. For the SRM joint problem (see Table 1), each of the 142 solutions computed by the ICCG iterative solver required approximately 100 iterations and very little benefit was gained in this case by using the previous solution as an initial guess. Using the direct solvers, only one factorization was required, and 142 right hand side vectors were solved. The solution time

Table 2. Comparison of Equation Solvers for Example Problems
Runs Made on NASA-Langley CRAY-2 Computer, normalized data in [ ]

Equation Solver	**Time** (sec)	**Operations** (adds/multiplies)	**Rate** (MFLOPS)	**Memory** (64-bit Words)
Composite Cross-Ply Laminate with Hole, 8-Noded Brick Elements, 11,929 Equations 397,139 Coefficients, Max. Semi-Bandwidth=1,593, Avg. Semi-Bandwidth=1,081				
BAND	48.9 [1.0]	13,150,259,538 [4.2]	269 [22.4]	12,901,825 [11.2]
SPK: factor/solve	223.5 [4.6]	7,832,183,056 [2.5]	35 [2.9]	7,518,911 [6.5]
with overhead	250.2 [5.1]	7,832,183,056 [2.5]	31 [2.6]	7,518,911 [6.5]
ICCG (950) γ =.02	260.5 [5.3]	3,137,874,415 [1.0]	12 [1.0]	1,203,346 [1.0]
JCG (2,864)	223.8 [4.6]	12,843,778,146 [4.1]	57 [4.8]	1,149,180 [1.0]
Space Shuttle Solid Rocket Booster, Mixed Element Types, 54,870 Equations 1,310,973 Coefficients, Max. Semi-Bandwidth=894, Avg. Semi-Bandwidth=381				
BAND	34.6 [1.0]	7,582,850,299 [1.5]	219 [19.9]	20,925,813 [8.1]
SPK: factor/solve	162.8 [4.7]	4,942,898,728 [1.0]	30 [2.7]	13,033,299 [5.0]
with overhead	218.3 [6.3]	4,942,898,728 [1.0]	23 [2.1]	13,033,299 [5.0]
ICCG (1,426) γ = .14	1534.0 [44.3]	16,110,861,248 [3.3]	11 [1.0]	3,987,789 [1.5]
JCG (6,758)	968.0 [28.0]	73,853,772,956 [14.9]	76 [6.9]	2,594,332 [1.0]
High-Speed Civil Transport Aircraft, Mixed Element Types, 16,146 Equations 499,505 Coefficients, Max. Semi-Bandwidth=594, Avg. Semi-Bandwidth=319				
BAND	6.8 [1.0]	1,362,079,199 [1.3]	200 [16.7]	5,160,591 [5.5]
SPK: factor/solve	37.1 [5.5]	1,083,225,624 [1.0]	29 [2.4]	3,360,105 [3.6]
with overhead	62.6 [9.2]	1,083,225,624 [1.0]	17 [1.4]	3,360,105 [3.6]
ICCG (1,124) γ =.048	395.7 [58.2]	4,726,562,076 [4.4]	12 [1.0]	1,514,661 [1.6]
JCG (8,310)	448.7 [66.0]	31,750,850,466 [29.3]	71 [5.9]	936,913 [1.0]

JCG - Jacobi Preconditioned Conjugate Gradient (Diagonal scaling), diagonal storage of matrix, number of iterations in parenthesis

for both direct solvers would be considerably higher if the same problem required repeated factorizations of the input matrix, while the iterative solution time would remain about the same. A computation rate of over 70 MFLOPS on the CRAY-2 for the JCG iterative method (see Table 2) was achieved by the use of diagonal storage of the system matrix. However, the number of iterations required for the JCG method was too high for the problems considered to make it competitive with the direct solvers.

One factor that significantly affects the convergence rate of iterative solvers for structural analysis problems is element aspect ratio. Table 3 demonstrates this effect for a prismatic solid, with physical dimensions $l \times l \times lz$, modeled using 8-noded brick elements. The solid is constrained at the four corners and uniformly loaded along one face. In this example, the aspect ratio is varied by changing the height (or thickness, lz) of the prismatic solid. For the model with an element aspect ratio of 1, the JCG method was nearly three times faster than the variable-band solver. However, as the element aspect ratio is increased, the number of iterations increases dramatically and the iterative solver is no longer competitive with the variable-band solver. A similar effect has been observed for shell elements where the element thickness enters the stiffness matrix through the constitutive relations. The number of iterations required for a given convergence tolerance decreases dramatically as the ratio of element planar dimension to the thickness approaches one. The ICCG method requires increasingly fewer iterations than the JCG method as the element aspect ratio increases, but the computation rate is also much slower. This example suggests that the iterative solvers can be expected to perform most efficiently for problems with very fine meshes where aspect ratios are closer to unity.

Table 3. Effect of Element Aspect Ratio on Performance of Iterative Solvers
Runs Made on NASA-Langley CRAY-2 Computer, Normalized Data in [ ]

Prismatic Solid with $10 \times 10 \times 10$ Nodes, Physical Dimensions $l \times l \times lz$
8-Noded Brick Elements, 2,988 Equations, 99,661 Coefficients
Max. Semi-Bandwidth=336, Avg. Semi-Bandwidth=314

Equation Solver	Iterations	Time (sec)	Operations (adds/multiplies)	Rate (MFLOPS)
BAND	n/a	1.47 [4.0]	311,461,433 [10.0]	212 [17.7]
JCG ($l/lz = 1$)	52 [1.3]	.37 [1.0]	31,121,268 [1.0]	84 [7.0]
JCG ($l/lz = 10$)	449 [11.0]	3.12 [8.4]	262,244,050 [8.4]	84 [7.0]
JCG ($l/lz = 20$)	1,016 [24.8]	6.99 [18.9]	592,292,340 [19.0]	84 [7.0]
ICCG ($l/lz = 1$)	41 [1.0]	3.43 [9.3]	40,282,578 [1.3]	12 [1.0]
ICCG ($l/lz = 10$)	123 [3.0]	9.57 [25.9]	99,308,664 [3.2]	12 [1.0]
ICCG ($l/lz = 20$)	382 [9.3]	26.65 [26.7]	320,010,996 [10.3]	12 [1.0]

CONCLUDING REMARKS

The availability of a variety of vectorized direct and iterative equation solvers within a common large-scale finite element software system leads to new research areas and expanded capabilities in finite element analysis. Equation solution times for problems requiring several billion operations can now be performed in seconds on today's high-performance computers. The use of both direct and iterative equation solvers to solve representative structural analysis problems shows that the relative performance of the solvers depends both on the amount of computations as well as the rate at which the operations can be carried out on a given computer. The variable-band Choleski solver is very fast for many structural analysis problems. Sparse solvers are most attractive for models composed of higher order finite elements, where they can benefit most from a greatly reduced operation count relative to the variable-band solvers. Iterative methods require much less memory than the direct solvers, but their effective use depends upon a fast convergence rate. The convergence rate is best for elements with low aspect ratios and may also be improved if a good initial guess is available. The effective use of high-performance computers in solving challenging structural analysis problems requires interaction between structural engineers and numerical analysts in order to match the characteristics of a given problem with the capabilities of a variety of equation solvers.

REFERENCES

1. Poole, E.L.; and Overman, A.L.: *The Solution of Linear Systems of Equations with a Structural Analysis Code on the NAS CRAY-2.* NASA CR-4159, December, 1988.
2. Axelsson, O.: *A Survey of Vectorizable Preconditioning Methods for Large Scale Finite Element Matrix Problems.* Center for Numerical Analysis, University of Texas at Austin, Report no. CNA-190, 1984.
3. Simon, H.; Vu, P.; and Yang, C.: *Performance of a Supernodal General Sparse Solver on the CRAY Y-MP: 1.68 GFLOPS with Autotasking.* Boeing Computer Services, SCA-TR-117, 1989.
4. Stewart, C.B. (compiler): *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644. 1989.
5. Agarwal, T.K.; Storaasli, O.O.; and Nguyen, D.T.: A Parallel-Vector Algorithm for Rapid Structural Analysis on High-Performance Computers. AIAA Paper No. 90-1149.
6. Ortega, J.M.: *Introduction to Parallel and Vector Solution of Linear Systems.* Plenum Press, New York, 1988.
7. Chu, E.; and George, J.A.: *Sparse Matrix Methods Research Using the CSM Testbed Software System.* NASA CR-4219, March, 1989.
8. Mantueffel, T.A.: An Incomplete Factorization Technique for Positive Definite Linear Systems. *Mathematics and Computation*, Vol. 34, 1980, pp 473-497.
9. Greene, W.H.; Knight, N.F.; and Stockwell, A.E.: Structural Behavior of the Space Shuttle SRM Tang-Clevis Joint. *Journal of Propulsion and Power.* Vol. 4, No. 4, 1988, pp.317-327.
10. Knight, N.F.; Gillian, R.E.; and Nemeth, M.P.: *Preliminary 2-D Shell Analysis of the Space Shuttle Solid Rocket Boosters.* NASA TM-100515, March 1988.
11. Robins, A. W.; Dollyhigh, S. M.; Beissner, F. L., Jr.; Geiselhart, K.; Martin, G. L.; Shields, E. W.; Swanson, E. E.; Coen, P. G.; and Morris, S. J., Jr.: *Concept Development of a Mach 3.0 High-Speed Civil Transport.* NASA TM-4058, 1988.
12. Kao, P. J.; Wrenn, G. A.; and Giles, G. L.: *Comparison of Equivalent Plate and Finite Element Analysis of a Realistic Aircraft Structural Configuration.* AIAA Paper No. 90-3293.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TM-102735 AVSCOM TM-90-B-017	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle High-Performance Equation Solvers and Their Impact on Finite Element Analysis		5. Report Date September 1990
		6. Performing Organization Code
7. Author(s) Eugene L. Poole, Norman F. Knight, Jr., D. Dale Davis, Jr.		8. Performing Organization Report No.
		10. Work Unit No. 505-63-01-10
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665-5225 and Aerostructures Directorate, USAARTA-AVSCOM Hampton, VA 23665-5225		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001 and U.S. Army Aviation Systems Command St. Louis, MO 63120-1798		14. Sponsoring Agency Code

15. Supplementary Notes

Eugene L. Poole, Awesome Computing Inc., Hampton, Virginia

Norman F. Knight, Jr., Formerly NASA Langley Research Center, Hampton, Virginia 23665
Currently Clemson University, Clemson, South Carolina

D. Dale Davis, Jr., Aerostructures Directorate, U.S. Army Aviation Research and Technology Activity; currently assigned to Computational Mechanics Branch, NASA Langley Research Center

Published in *FEM in the Design Process*, ed. John Robinson, proceedings of Sixth World Conference on Finite Element Methods, October 1-5, 1990, Banff, Alberta, Canada

16. Abstract

The role of equation solvers in modern structural analysis software is described. Direct and iterative equation solvers which exploit vectorization on modern high-performance computer systems are described and compared. The direct solvers are two Choleski factorization methods. The first method utilizes a novel variable-band data storage format to achieve very high computation rates and the second method uses a sparse data storage format designed to reduce the number of operations. The iterative solvers are preconditioned conjugate gradient methods. Two different preconditioners are included; the first uses a diagonal matrix storage scheme to achieve high computation rates and the second requires a sparse data storage scheme and converges to the solution in fewer iterations than the first. The impact of using all of the equation solvers in a common structural analysis software system is demonstrated by solving several representative structural analysis problems.

17. Key Words (Suggested by Authors(s)) Equation solvers, Choleski method iterative methods, preconditioned conjugate gradient	18. Distribution Statement Unclassified—Unlimited Subject Category 39

19. Security Classif.(of this report) Unclassified	20. Security Classif.(of this page) Unclassified	21. No. of Pages 16	22. Price A03

NASA FORM 1626 OCT 86

For sale by the National Technical Information Service, Springfield, Virginia 22161-2171