Fully Vectorized EBE Preconditioners for Nonlinear Solid Mechanics: Applications to Large-Scale Three-Dimensional Continuum, Shell and Contact/Impact Problems*

THOMAS J. R. HUGHES† AND ROBERT M. FERENCZ†

Abstract. Computing solutions to systems of linear equations remains the dominant cost for large-scale implicit finite element calculations, especially in three-dimensional applications. Direct methods can easily require prohibitively large amounts of supercomputer CPU and storage, even for relatively coarse meshes. Iterative procedures avoiding the formation and factorization of a global system of equations can circumvent this difficulty. Element-by-element (EBE) preconditioning techniques are examined in the context of the production nonlinear solid mechanics code NIKE3D. The recently introduced concept of a fractal dimension of a finite element mesh is reviewed, and proves useful in characterizing the efficiency of this iterative algorithm with respect to a variable band, active column direct method. Sample calculations using a CRAY X-MP/48 with Solid-state Storage Device (SSD) illustrate the performance and range of applicability of the EBE algorithms considered. In addition to continuum models, results are also presented for analyses utilizing shell elements and contact/impact algorithms.

Introduction. Solution of linear systems of equations remains the dominant computational cost in large-scale implicit finite element calculations, particularly in three-dimensional applications. For linear analyses with a few ten-thousands of unknowns, direct equation solving typically consumes over ninety percent of total CPU usage and entails massive storage requirements. These well-known difficulties have inspired the study and development of iterative methods, and the ongoing growth of parallel computation further stimulates this research.

The application of iterative methods and preconditioners with a global data base, for example, Gauss-Seidel, SSOR and Incomplete Cholesky, suffer from a

* The research supported herein was sponsored by the Lawrence Livermore National Laboratory under grant 5435505, and the U.S. Office of Naval Research under Contract No. N00014-84-K-0600.

† Division of Applied Mechanics, Durand Building, Stanford University, Stanford, California, 94305.

major drawback for finite element calculations in solid and structural mechanics: mapping such global operators onto multiprocessor architectures remains a nontrivial problem for the complex meshes often required in engineering applications. Indeed, this problem remains a barrier to the enhancement of direct solution methods via parallel processing. A contrasting philosophy aspires to utilize the local data structure inherent in finite element formulations and implementations. Element-by-element (EBE) methods were first introduced by [*Hughes 83a*] as a means of implicit time-integration for heat conduction, and were viewed as a natural generalization of the operator splitting methods developed primarily for finite difference methods. Based upon this work, Ortiz, Pinsky and Taylor [*Ortiz 83*] developed a novel time-stepping scheme for second-order systems. While unconditionally stable and formally second-order accurate, these algorithms displayed unacceptable spatial truncation errors in some test problems. Subsequent efforts shifted to using EBE procedures for solving systems of linear equations emanating from standard time-integration schemes [*Hughes 83b*]. In particular, the EBE procedure has been recognized as an effective preconditioner for use with the conjugate gradients algorithm as an iterative driver [*Hughes 83c,84a,87, Nour-Omid 85, Winget 85*].

This paper presents an overview of two element-by-element preconditioned conjugate gradients (EBE/PCG) algorithms currently under study. These procedures are being tested within the production nonlinear stress analysis code NIKE3D [*Hallquist 84*] developed at Lawrence Livermore National Laboratory. NIKE's nonlinear equation solving strategy is first summarized and the Crout EBE/PCG algorithm reviewed. The extension of this methodology to NIKE's contact/impact algorithms is presented, and numerical examples computed on a CRAY X-MP/48 with 134 megaword Solid-state Storage Device (SSD) using both continuum and shell elements illustrate its performance. Motivated by a desire to reduce the element data base, we re-examine the symmetrized Gauss-Seidel EBE preconditioner, which requires only half the storage of the Crout version. Sample analyses contrast its performance with the other equation solving algorithms considered.

NIKE's Nonlinear Iteration Strategy. A standard weak form of the initial boundary value problem of solid mechanics serves as the basis for a finite element formulation in the spatial domain, leading to the semi-discrete, and in general nonlinear, matrix equation of motion:

$$\mathbf{Ma} + \mathbf{F}^{\text{int}} = \mathbf{F}^{\text{ext}},$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{a}$ is the nodal acceleration vector, and $\mathbf{F}^{\text{int}}$ and $\mathbf{F}^{\text{ext}}$ are the vectors of internal and external nodal forces, respectively. The internal forces depend upon the nodal displacement vector $\mathbf{d}$, typically via an inelastic, rate constitutive model, e.g., see [*Hughes 84b*]. Application of an implicit time-integration algorithm [*Newmark 59*] to the semi-discrete equation of motion gives rise to a nonlinear algebraic equation at each time step. NIKE utilizes a number of quasi-Newton algorithms to solve this nonlinear algebraic equation. All require the solution of the linear system

$$\tilde{\mathbf{K}}(\mathbf{d}_{n+1}^{(i)})\Delta\mathbf{d}_{n+1}^{(i+1)} = \mathbf{R}(\mathbf{d}_{n+1}^{(i)}), \tag{1}$$

where the subscripts refer to the time step number, the superscripts refer to the number of iterations within the current step, $\tilde{\mathbf{K}}(\mathbf{d}_{n+1}^{(i)})$ is a symmetric, positive-definite, approximate tangent matrix derived from quasi-Newton updates of a previously formed effective stiffness matrix $\mathbf{K}(\mathbf{d}_{n+1}^{(j)})$, $j < i$, $\Delta\mathbf{d}_{n+1}^{(i+1)}$ is a vector of unknown displacement increments, and $\mathbf{R}(\mathbf{d}_{n+1}^{(i)})$ is the residual, or out-of-balance force. Direct solutions are performed by FISSLE, a vectorized variable band, active column solver developed by R. L. Taylor and S. J. Sackett [*Taylor 81*]. Equation (1) is solved repeatedly with an auxiliary line search until $\|s_{i+1}\Delta\mathbf{d}_{n+1}^{(i+1)}\| < \delta_N \max_j \|s_j\Delta\mathbf{d}_{n+1}^{(j)}\|$, where s is the search parameter and typically $\delta_N \in [10^{-4}, 10^{-3}]$, at which time convergence is presumed and the displacement is updated: $\mathbf{d}_{n+1} = \mathbf{d}_n + \sum_i s_i\Delta\mathbf{d}_{n+1}^{(i)}$. If the residual begins to diverge, i.e., $\|\mathbf{R}(\mathbf{d}_{n+1}^{(i+1)})\| > \|\mathbf{R}(\mathbf{d}_{n+1}^{(i)})\|$, the stiffness matrix is reformed as a function of $\mathbf{d}_{n+1}^{(i)}$ and nonlinear iteration resumed.

Modified Preconditioned Conjugate Gradients. The following *linear iteration* procedure, based upon the conjugate gradients algorithm [*Hestenes 52*], may be employed whenever NIKE's nonlinear iteration procedure requires the solution of a linear system. For conciseness, subsequent references to iteration will imply linear iteration. To simplify notation we consider the $N_{eq} \times N_{eq}$ matrix equation $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a symmetric, positive-definite coefficient matrix.

```
Step 1. Initialize and solve uncoupled equations:
            $m = 0$                                   (2)
            $\mathbf{x}_0 = \mathbf{0}$                                  (3)
            $\mathbf{r}_0 = \mathbf{b}$                                  (4)
        for $j = 1, 2, \ldots, N_{eq}$
            if $A_{ij} = 0$ for all $i \neq j$ then
                $x_j = A_{jj}^{-1} r_j$                        (5)
                $r_j = 0$                                (6)
            endif
            $\mathbf{p}_0 = \mathbf{z}_0 = \mathbf{B}^{-1}\mathbf{r}_0$                      (7)
Step 2. Line search to update solution and residual:
            $\alpha_m = \frac{\mathbf{r}_m \cdot \mathbf{z}_m}{\mathbf{p}_m \cdot \mathbf{A}\mathbf{p}_m}$                     (8)
            $\mathbf{x}_{m+1} = \mathbf{x}_m + \alpha_m \mathbf{p}_m$                (9)
            $\mathbf{r}_{m+1} = \mathbf{r}_m - \alpha_m \mathbf{A}\mathbf{p}_m$              (10)
Step 3. Check convergence:
        if $\|\mathbf{r}_{m+1}\| \leq \delta_L \|\mathbf{r}_0\|$ return
Step 4. Compute new conjugate search direction:
            $\mathbf{z}_{m+1} = \mathbf{B}^{-1}\mathbf{r}_{m+1}$                   (11)
            $\beta_m = \frac{\mathbf{r}_{m+1} \cdot \mathbf{z}_{m+1}}{\mathbf{r}_m \cdot \mathbf{z}_m}$                  (12)
            $\mathbf{p}_{m+1} = \mathbf{z}_{m+1} + \beta_m \mathbf{p}_m$              (13)
```

$$m = m + 1 \tag{14}$$

Go to *Step* 2.

Remark 1. The initialization operations of equations (5) and (6) arise from NIKE's treatment of nonzero displacement boundary conditions. These degrees of freedom are maintained as "active" diagonal equations in the system, which are then solved trivially when a direct method is used. Although a suitably defined preconditioner $\mathbf{B}$ will replicate this behavior during the solution of $\mathbf{B}\mathbf{z}_0 = \mathbf{r}_0$, the subsequent line search may scale the boundary displacements from the correct values. Experience has indicated very slow convergence–potentially to an incorrect solution–when this is the case. The present initialization serves to contract the search space to the remaining unspecified degrees of freedom. With this modification, the algorithm is applicable to systems with blocks of diagonal equations, as would arise from a mixed implicit-explicit time-integration scheme [*Hughes 79*].

Crout Element-by-Element Preconditioner. The choice of an appropriate preconditioner $\mathbf{B}$ is essential for the practical application of the PCG algorithm. We desire $\mathbf{B}^{-1}$ to approximate $\mathbf{A}^{-1}$ in the sense that the condition number $C(\mathbf{B}^{-1}\mathbf{A}) \to 1$ while retaining a computationally efficient structure. A particularly simple preconditioner is diagonal scaling, i.e., $\mathbf{B} = \operatorname{diag}(\mathbf{A})$, also known as Jacobi acceleration. This preconditioner may be realized by regularizing the system of equations with the global diagonal $\mathbf{W} = \operatorname{diag}(\mathbf{A})$:

$$\mathbf{W}^{-1/2}\mathbf{A}\mathbf{W}^{-1/2}\ \mathbf{W}^{1/2}\mathbf{x} = \mathbf{W}^{-1/2}\mathbf{b}, \tag{15}$$

resulting in the transformed system

$$\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}. \tag{16}$$

This system is solved with standard conjugate gradients, i.e, we take $\mathbf{B} = \mathbf{I}$, and then the solution to the original system is recovered: $\mathbf{x} = \mathbf{W}^{-1/2}\tilde{\mathbf{x}}$. In addition to automatically embedding diagonal scaling into the iterative driver, this regularization nondimensionalizes the equation system, ensuring that the residual norm used to monitor convergence is well-defined. Therefore, all of our iterative algorithms are poised in this transformed system.

Element-by-element preconditioners are motivated by a desire to maintain the element-based data structure of current finite element codes. For the *Crout EBE preconditioner*, we utilize the product decomposition

$$\mathbf{B} = \prod_{e=1}^{N_{el}} \mathcal{L}_p^e \times \prod_{e=1}^{N_{el}} \mathcal{D}_p^e \times \prod_{e=N_{el}}^{1} \mathcal{U}_p^e, \tag{17}$$

where

$$\mathcal{L}_p^e = \mathcal{L}_p^e(\bar{\mathbf{A}}^e, \mathbf{P}^e) = (\mathbf{P}^e)^T L_p[\mathbf{P}^e\bar{\mathbf{A}}^e(\mathbf{P}^e)^T]\mathbf{P}^e,$$
$$\mathcal{D}_p^e = \mathcal{D}_p^e(\bar{\mathbf{A}}^e, \mathbf{P}^e) = (\mathbf{P}^e)^T D_p[\mathbf{P}^e\bar{\mathbf{A}}^e(\mathbf{P}^e)^T]\mathbf{P}^e,$$
$$\mathcal{U}_p^e = \mathcal{U}_p^e(\bar{\mathbf{A}}^e, \mathbf{P}^e) = (\mathbf{P}^e)^T U_p[\mathbf{P}^e\bar{\mathbf{A}}^e(\mathbf{P}^e)^T]\mathbf{P}^e,$$

and N_{el} is the number of elements in the mesh. The reversed element order in the third product of (17) results in $\mathbf{B}$ being symmetric. If we denote the eth element's contribution to the global matrix $\tilde{\mathbf{A}}$ by $\tilde{\mathbf{A}}^e$, then $\bar{\mathbf{A}}^e$ is the *Winget regularized element matrix*:

$$\bar{\mathbf{A}}^e = \mathbf{I} + (\tilde{\mathbf{A}}^e - \tilde{\mathbf{W}}^e), \tag{18}$$

$$\tilde{\mathbf{W}}^e = \operatorname{diag}(\tilde{\mathbf{A}}^e). \tag{19}$$

The regularization ensures $\bar{\mathbf{A}}^e$ is positive-definite, hence the Crout factorization $(\mathbf{P}^e)^T L_p[\cdot] D_p[\cdot] U_p[\cdot] \mathbf{P}^e$ is well-defined. $\mathbf{P}^e$ is a permutation matrix whose role is to interchange rows and columns of $\bar{\mathbf{A}}^e$ such that they are consistent with the "local" nodal ordering. The element matrices $\mathbf{P}^e \bar{\mathbf{A}}^e (\mathbf{P}^e)^T$ and Crout factors $D_p[\cdot]$ and $U_p[\cdot]$ are computed each time $\mathbf{A}$ is formed in the nonlinear iteration.

A fully vectorized implementation of the Crout EBE preconditioner is operational within NIKE3D and described in detail in [*Hughes 87*]. Fundamental to successful vectorization is a reordering of the elements at the beginning of execution into a series of *internally disjoint* blocks, i.e., within each block no elements share common degrees of freedom. This permits the element forward reductions and backsubstitutions to be calculated "in parallel" within a block via vector pipelining. No optimization is performed with respect to average block size or inter-element communication, but the procedure appears applicable to any mesh topology. For logically regular meshes the algorithm produces a classic "checkerboard" reordering. This is analogous to the domain decomposition used to map EBE onto a Hypercube parallel processor [*Lyzenga 87*], so although the present PCG implementation does not multitask the four processors within the X-MP, this extension should be straightforward. Finally, it should be noted that the element matrices are naturally formed in their local ordering, i.e., $\mathbf{P}^e \mathbf{A}^e (\mathbf{P}^e)^T$ is computed directly prior to regularization and factorization. The remaining permutation matrices are a notational convenience indicating gather/scatter operations.

Extensions to Penalized Contact/Impact Algorithms. The ability to model contact/impact between unbonded material interfaces is essential for the realistic analysis of many mechanical assemblies. Combining a very flexible geometric representation with the algorithmic simplicity of penalized constraints, NIKE3D has a very general "slidesurface" capability able to represent tied interfaces (for bonded materials or mesh transitions) and interfaces opening and/or closing as dictated by the global equations of motion. For a detailed description see [*Hallquist 85*].

As an illustration we consider the symmetry plane option, which may be considered a "one-sided" tied slidesurface. The analyst defines the desired symmetry plane via a unit outward normal vector $\mathbf{n}$ whose tail lies on the plane and assigns a list of nodes to be constrained to lie upon the plane throughout the analysis. Each such node engenders a penalty element having the *local* representation

$$\mathbf{k}^s = 10^6 K\ \mathbf{n} \otimes \mathbf{n} \tag{20}$$

where K is a stiffness based upon the maximum bulk modulus among *all* materials in the mesh and the geometry of the largest and smallest elements lying on the symmetry plane. The outer product $\mathbf{n} \otimes \mathbf{n}$ generates a 3×3 matrix and the factor of 10^6 is NIKE's default penalty parameter.

General slidesurfaces generate 15×15 penalty elements; each node on one side of the surface couples to the four nodes of the nearest "master" element belonging to the opposing surface. The stiffness K is a function of the bulk modulus of that specific master element. For tied slidesurfaces NIKE uses a penalty parameter of 10^2, while opening/closing interfaces use a penalty of 1. Thus general slidesurfaces do not create nearly as much ill-conditioning as symmetry planes.

For direct solution methods, the penalty element matrix is assembled into the global coefficient matrix by the standard FEM procedure. The corresponding extension to the Crout EBE preconditioner leads to

$$\mathbf{B} = \prod_{e=1}^{N_{el}} \mathcal{L}_p^e \times \prod_{s=1}^{N_{sel}} \mathcal{L}_p^s \times \prod_{e=1}^{N_{el}} \mathcal{D}_p^e \times \prod_{s=1}^{N_{sel}} \mathcal{D}_p^s \times \prod_{s=N_{sel}}^{1} \mathcal{U}_p^s \times \prod_{e=N_{el}}^{1} \mathcal{U}_p^e, \tag{21}$$

where N_{sel} is the number of symmetry plane/slidesurface penalty elements and $\mathcal{L}_p^s \mathcal{D}_p^s \mathcal{U}_p^s$ is the Crout factorization of a locally-ordered, Winget regularized penalty element. We refer to the above definition of $\mathbf{B}$ as the *interior* Crout EBE preconditioner, as contrasted with the *exterior* version

$$\mathbf{B} = \prod_{s=1}^{N_{sel}} \mathcal{L}_p^s \times \prod_{e=1}^{N_{el}} \mathcal{L}_p^e \times \prod_{e=1}^{N_{el}} \mathcal{D}_p^e \times \prod_{s=1}^{N_{sel}} \mathcal{D}_p^s \times \prod_{e=N_{el}}^{1} \mathcal{U}_p^e \times \prod_{s=N_{sel}}^{1} \mathcal{U}_p^s \,. \tag{22}$$

The interior form was chosen for inclusion in NIKE3D due to its marginally better performance relative to the exterior version as observed in numerical experiments with the two-dimensional code NIKE2D [*Hallquist 83*].

Fractal Dimension of a Finite Element Mesh. Before discussing our computational experience with the EBE/PCG algorithm, we wish to review the recently introduced concept of the fractal dimension of a mesh [*Hughes 87*]. Traditionally, a mesh is said to be n-dimensional if it is composed of n-dimensional elements. Hence a mesh consisting of a long strip of three-dimensional continuum elements is labelled a three-dimensional mesh. Such nomenclature obscures a key measure of the relative efficiency of iterative and direct solution algorithms.

Definition. Given a mesh containing three-dimensional, eight-node brick continuum elements, consider three meshes with equal numbers of active degrees of freedom: a strip of elements connected on end, a square with one element through the thickness, and a cube. We refer to these as the one-, two- and three-dimensional reference meshes. For each reference mesh we compute the total profile storage, or "fill", needed to hold the associated global matrix $\mathbf{A}$. Then the fill for the given mesh determines a *fractal dimension* d_f:

If $fill \leq fill_{2\mathrm{D}}$

$$d_f = \frac{fill - fill_{1\mathrm{D}}}{fill_{2\mathrm{D}} - fill_{1\mathrm{D}}} + 1, \tag{23}$$

If $fill > fill_{2D}$

$$d_f = \frac{fill - fill_{2D}}{fill_{3D} - fill_{2D}} + 2. \tag{24}$$

For simplicity, the fill for each reference mesh is computed using its *peak* bandwidth. NIKE's four-node shell element has the same number of degrees of freedom as the eight-node continuum element, and in the general case identical nodal connectivity within an element, so it is consistent to use the same reference meshes to define the fractal dimension of a model containing these shell elements.

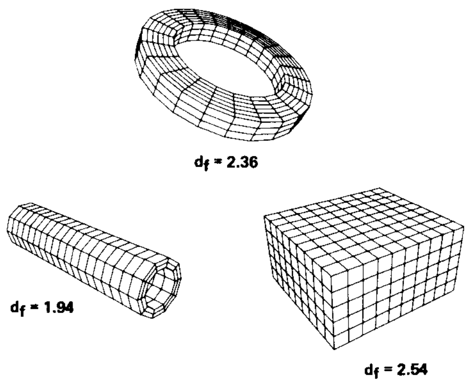

Fig. 1. Fractal dimension of three simple meshes.

Given that the number of equations N_{eq} is constant, the fractal dimension reflects the change in mean bandwidth of **A** as a function of mesh configuration. Fig. 1 presents the fractal dimension for three simple meshes: a hollow cylinder, an annulus and a truncated cube. Note that if the annulus is "cut" and opened into a column, then d_f decreases to 1.79. As d_f approaches 1.0, two effects combine to make direct solution more efficient. First, the bandwidth of **A** decreases, reducing the operations for direct factorization which are proportional to the bandwidth squared. Second, for the preconditioners discussed here, the number of iterations scales with the largest number of nodes along an edge, increasing the cost of PCG. For a well-conditioned linear problem, PCG/EBE has been observed to be more efficient than FISSLE for $d_f >\approx 1.5$; see [*Hughes 87*] for further discussion.

Numerical Examples. The following calculations illustrate our experience with applying EBE/PCG to problems utilizing the contact/impact algorithms within NIKE3D. Except where indicated, all computing was performed on a CRAY X-MP/48 with a 134 megaword Solid-state Storage Device (SSD) running under the Livermore Time Share System (LTSS). We are currently using the CFT 1.14 FORTRAN compiler, which utilizes both the bi-directional memory and hardware gather/scatter capabilities of the X-MP. Unless noted otherwise, all computations utilized nonlinear and linear iteration tolerances of $\delta_N = 10^{-3}$ and $\delta_L = 10^{-4}$, respectively. The CPU and I/O costs we present are for equation solving costs only, and include assembly of the global equations for the direct method, but exclude element formation for all methods. The I/O costs include I/O-related system charges, which can vary with the level of machine usage. Storage requirements are in units of 64-bit words.

- Curved Bar with Symmetry Plane

As a first example we consider the mesh illustrated in Fig 1. Physically, the problem represents a bar curving 90 degrees in a plane with both ends fixed and loaded transversely at midspan. Computationally, only 45 degrees are modeled with a symmetry plane specified at midspan; nodal forces consistent with the shear stress distribution of a straight beam are applied at this same location. The model contains 833 nodes, 576 continuum and 49 penalty elements, with 2,352 degrees of freedom and fractal dimension of 2.27. For such a small equation system we do not expect the iterative methods to be competitive with direct solution. However, the execution timings in Table 1 serve to illustrate the ability of the EBE preconditioner to control the ill-conditioning engendered by the large penalty stiffness of the symmetry plane elements. The graph in Fig. 3 further highlights the continuous degradation of diagonal scaling as a function of the penalty parameter, which has a default value of 10^6 (see Eq. (16)). Again, the EBE performance is highly insensitive to the choice of penalty parameter. This example justifies abandoning diagonal preconditioning for problems with symmetry planes.

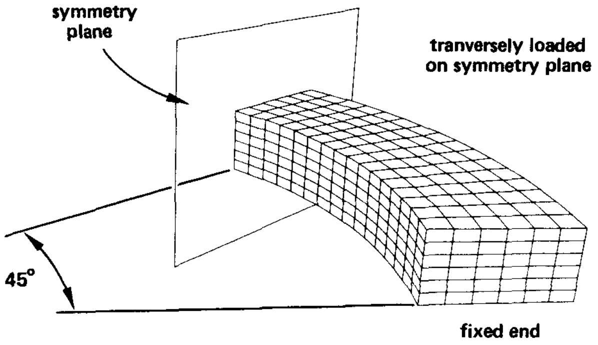

Fig. 2. Curved bar with penalized symmetry plane.

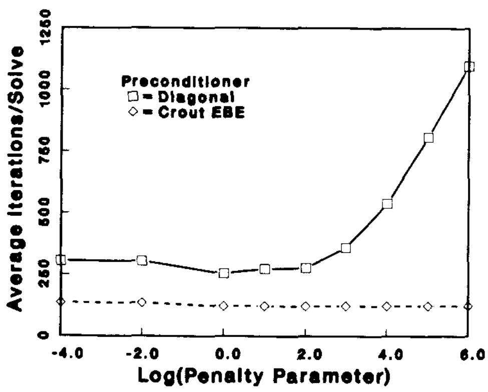

Fig. 3. PCG iterations for curved bar as function of symmetry plane penalty parameter.

Method	Linear Solves	Avg. Iter. per Solve	Max. Iter. per Solve	CPU (sec)
Direct[a]	3	—	—	5.3
Diagonal[a]	3	1097.7	1459	35.4
EBE[a]	3	123.7	124	8.5
Diagonal[b]	3	255.0	328	8.3
EBE[b]	3	122.7	129	8.3

[a] default penalty parameter, [b] penalty parameter reduced by 10^{-6}.

Table 1. Equation solving data for a curved bar with penalized symmetry plane.

- Pipe Whip Analysis

This analysis represents the first attempt at using NIKE3D's nonlinear shell element with PCG solution methods, in this case to model the impact of two steel pipes. As shown in Fig. 4, the target pipe is fixed at both ends while the other is rotating about a fixed pivot with a specified initial angular velocity. The model contains 915 nodes, 840 shell elements and 5,038 degrees of freedom with an initial fractal dimension of 2.23. The symmetry plane in Fig. 4 is normal to one of the global coordinate axes, hence the symmetry conditions are trivially enforced with nodal boundary conditions. Penalty elements arise only from the contact surface between the two pipes. The contact area remains small throughout the analysis, resulting in the fractal dimension never exceeding 2.24. The 0.01 second process is equally subdivided into 200 timesteps, and the magnitude of mesh deformation is illustrated by Fig. 5.

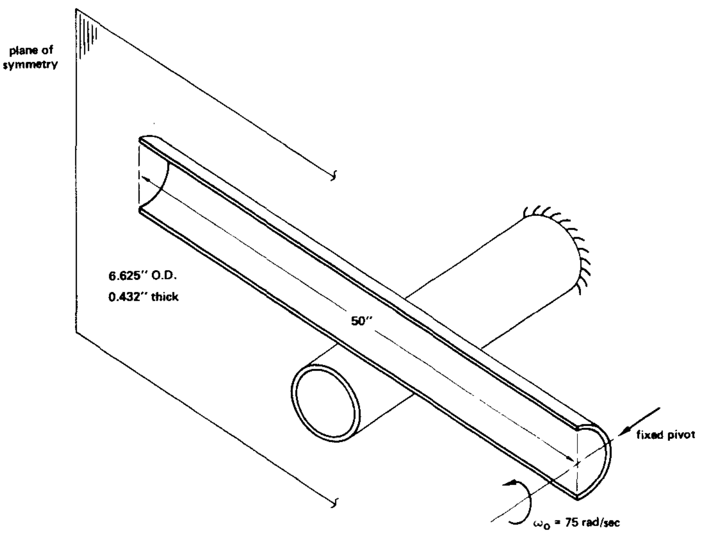

Fig. 4. Boundary and initial conditions for pipe whip analysis. Symmetry enforced with nodal boundary conditions.

All solutions were performed in core, and the results are summarized in Table 2. The maximum element aspect ratio h^e/t is 4.6, implying the ratio of membrane to bending stiffness is roughly 20. In many *linear* analysis applications, shell elements with $h^e/t > 100$ are utilized, and we conjecture that the resulting ill-conditioning may lead to an unacceptable number of iterations for convergence. However, the mesh refinement in this problem may be typical of that required to model *nonlinear* phenomena, and in this case iterative methods appear to offer improved efficiency for practical applications. The smaller penalty parameter of the contact elements prevents a catastrophic degradation of performance for diagonal preconditioning. Nonetheless, EBE uses 15 percent less CPU than diagonal preconditioning and 79 percent less than direct solution. The difference in maximum effective plastic strain e^p is less than two percent, which is acceptable for engineering accuracy, but a greater error between direct and EBE/PCG than we have typically observed. This discrepancy may arise from the large rigid-body motions which interact with the plastic buckling of the rotating pipe. PCG iteration converges from largest to smallest eigenvalue, hence the rigid-body response would be the least well resolved.

Method	No. **K** Forms	Linear Solves	Avg. Iter. per Solve	Max. Iter. per Solve	Max. e^p	CPU (sec)	Core Storage
Direct	200	413	—	—	.2954	2203	1,223,269
Diagonal	200	413	83.0	96	.2915	542	292,312
EBE	200	412	33.7	40	.2907	459	549,350

Table 2. Equation solving data for the pipe whip analysis.

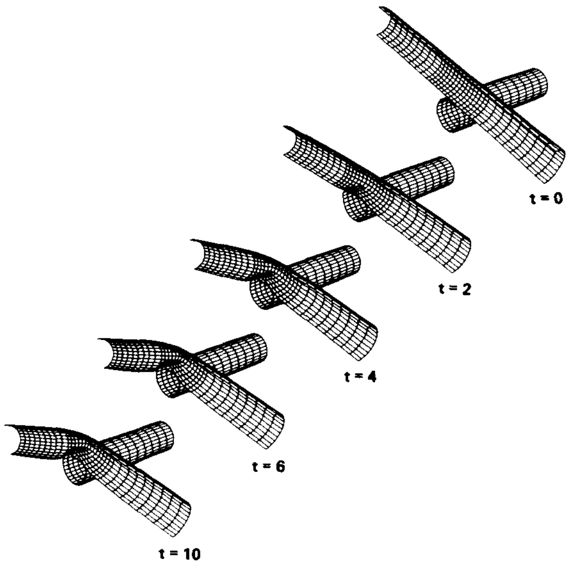

Fig. 5. Deformed shapes for pipe whip analysis. Time in milliseconds.

- O-ring Crush

This analysis concerns the seating of a metallic o-ring used to seal an assembly containing high-pressure gas. The mesh generated by R. A. Bailey (Fig. 6) contains 4,816 nodes, 3,304 continuum elements and 14,017 degrees of freedom ($d_f = 3.01$), and utilizes a penalized symmetry plane and several contact surfaces. The contact surface penalty was increased by a factor of ten and the penalty logic modified in order to control the potential rigid-body motion of the o-ring at the start of the analysis. This quasi-static analysis is divided into ten time steps and is driven both by displacement boundary conditions and concentrated forces applied to the upper flange. Fig. 7 displays the resulting deformation of the o-ring. The timings in Table 3 were obtained on an X-MP/416; the larger memory allocation available on this machine permitted the EBE analyses to be run in core. Initial results with EBE/PCG were disappointing, as the equation solving costs were equivalent to the direct method. This stems from the extremely inexpensive resolve capability of direct methods, i.e., for large problems, the vast majority of CPU cost occurs during Crout factorization of the assembled global stiffness matrix. In contrast, the EBE/PCG algorithm has approximately the same cost for each linear solve (depending upon the spectral character of the residual). The use of diagonal scaling was not attempted due to its poor performance with penalized symmetry planes as noted in the first example.

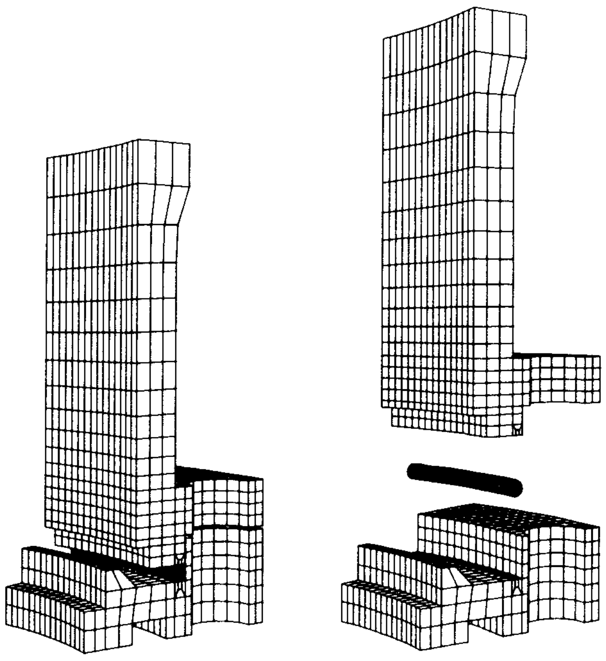

Fig. 6. Mesh for o-ring crush analysis with exploded view.

Method	No. **K** Forms	Linear Solves	Avg. Iter. per Solve	Max. Iter. per Solve	Max. e^p	CPU (sec)	I/O (sec)	Total CPU
Direct	21	159	—	—	.1357	4036	58	4629
EBE[(a)]	21	150	228.4	278	.1357	4053	0	4639
EBE[(b)]	21	81	214.4	280	.1355	2113	0	2453
EBE[(c)]	19	83	147.3	208	.1353	1482	0	1847

[(a)] 10 BFGS updates max., [(b)] 5 BFGS updates max., [(c)] 5 BFGS updates max. and PCG tolerance of 10^{-3}.

Table 3. Equation solving data for the o-ring crush analysis analysis.

By default, NIKE3D allows a maximum of ten BFGS updates in the nonlinear iteration before reforming the stiffness matrix. Thus the direct method has up to ten linear solves over which to amortize the cost of a factorization. For the EBE procedure, the regularization/factorization costs associated with a stiffness reformation are trivial, hence it is desirable to reform more frequently *if* this speeds convergence of the nonlinear iterations, thereby reducing the number of linear solves. A second EBE analysis was attempted with the nonlinear iteration limited to a maximum of five BFGS updates prior to reformation of the stiffness matrix. This strategy reduces the number of linear solves from 150 to 81 with a corresponding 47 percent reduction in equation solving cost. The third EBE analysis listed in Table 3 retains the revised BFGS strategy of the second, but relaxes the linear iteration tolerance δ_L from 10^{-4} to 10^{-3}. This leads to a 30 percent reduction in average iterations per solve and equation solving CPU. This final analysis requires 60 percent less CPU than the direct method. The larger PCG tolerance is still sufficient to maintain accuracy, as illustrated by the maximum effective plastic strain e^p. This numerical experiment underscores the need for automated adaptive selection of convergence tolerances and iteration parameters, e.g., number of Quasi-Newton updates, based on communication between the nonlinear and linear iteration procedures.

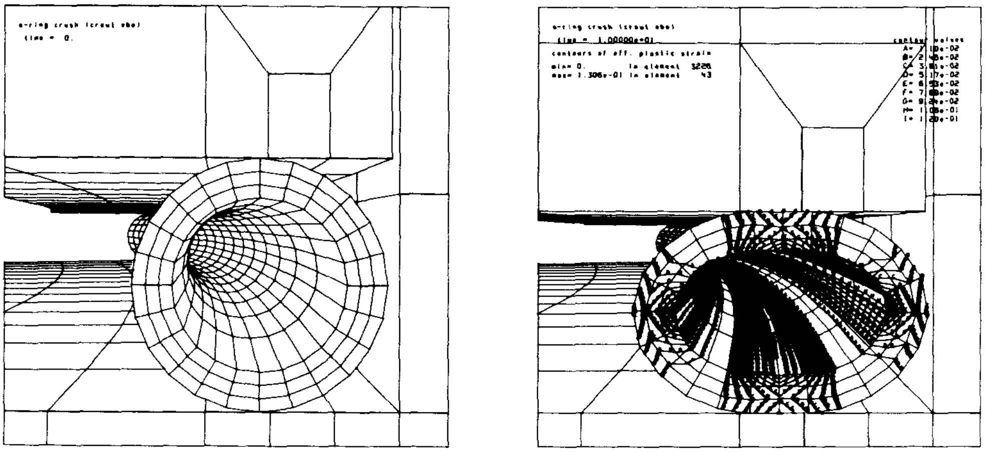

Fig. 7. Initial and final states of o-ring.

Symmetrized Gauss-Seidel EBE Preconditioner. The previous numerical examples, and those in [*Hughes 87*], indicate that Crout EBE/PCG has considerable capability to reduce the equation solving costs in three-dimensional applications. Indeed, the efficiency of EBE/PCG relative to direct solution increases with problem size and fractal dimension. One drawback with Crout EBE is the need to store *twice* as much element data as with diagonal preconditioning. To describe an alternative EBE preconditioner, recall the Winget regularized element matrix associated with the transformed system:

$$\bar{\mathbf{A}}^e = \mathbf{I} + \tilde{\mathbf{A}}^e - \tilde{\mathbf{W}}^e \tag{25}$$

$$= \mathbf{I} + \mathcal{L}_s(\tilde{\mathbf{A}}^e) + \mathcal{U}_s(\tilde{\mathbf{A}}^e), \tag{26}$$

where $\tilde{\mathbf{A}}^e = \mathcal{L}_s(\tilde{\mathbf{A}}^e) + \mathcal{D}_s(\tilde{\mathbf{A}}^e) + \mathcal{U}_s(\tilde{\mathbf{A}}^e)$ is the *sum* decomposition of the transformed element matrix. This gives rise to the *symmetrized Gauss-Seidel EBE preconditioner*:

$$\mathbf{B}_{G-S} = \prod_{e=1}^{N_{el}} \{\mathbf{I} + \mathcal{L}_s^e\} \times \prod_{e=N_{el}}^{1} \{\mathbf{I} + \mathcal{U}_s^e\}, \tag{27}$$

where

$$\mathcal{L}_s^e = \mathcal{L}_s^e(\tilde{\mathbf{A}}^e, \mathbf{P}^e) = (\mathbf{P}^e)^T L_s[\mathbf{P}^e \tilde{\mathbf{A}}^e (\mathbf{P}^e)^T]\mathbf{P}^e,$$
$$\mathcal{U}_s^e = \mathcal{U}_s^e(\tilde{\mathbf{A}}^e, \mathbf{P}^e) = (\mathbf{P}^e)^T U_s[\mathbf{P}^e \tilde{\mathbf{A}}^e (\mathbf{P}^e)^T]\mathbf{P}^e.$$

In this way only one matrix must be saved for each element, thereby reducing storage requirements by 50 percent compared to Crout EBE. We emphasize that a *product* form has been retained for the operator splitting, but the regularized element matrices are *sum* decomposed rather than using the product decomposition $\mathcal{L}_p\mathcal{D}_p\mathcal{U}_p$ as in the Crout preconditioner. The permutation matrix $\mathbf{P}^e$ signifies the same global-to-local reordering and gather/scatter operations previously discussed for the Crout EBE preconditioner. The proposed symmetrized Gauss-Seidel EBE preconditioner requires *fewer* floating-point operations per iteration than its Crout counterpart. Not only is there no element factorization, but the *element* diagonal scaling is eliminated from each preconditioning.

The symmetrized Gauss-Seidel (G-S) EBE preconditioner was previously compared to the Crout EBE preconditioner in [*Hughes 83c*] and [*Winget 85*] on a small test problem. The Crout EBE preconditioner proved superior. Nevertheless, given the fact that storage is a prime concern in the analysis of large systems and that the G-S EBE preconditioner is optimal in this regard, and further that the present implementation actually decreases operations per iteration compared with Crout EBE, it seemed worthwhile to re-evaluate it in the context of large-scale problems. Credit for the original proposal of the G-S EBE preconditioner is due to Nour-Omid and Parlett [*Nour-Omid 85*]. Nour-Omid and Parlett refer to it as the "element splitting" preconditioner. They do not employ the transformed system or the Winget regularized element array in their approach. Recently, Nour-Omid and Raefsky have also been employing element splitting in their Lanczos-driven EBE algorithm [*Nour-Omid 86*].

Numerical Examples. The following results compare the performance of the proposed symmetrized Gauss-Seidel EBE preconditioner with the methods previously discussed. Timing data for several of these problems appear in [*Hughes 87*]. Since that work was completed, NIKE3D has undergone revision, in particular a rewrite of FISSLE's global assembly algorithm by D. Stillman and the EBE/PCG procedure has been altered to utilize the transformed equation system (see Eq. (15)). The use of the transformed residual to monitor convergence has led to small changes in the iteration counts previously reported. Also, a pronounced gain in efficiency has resulted from the installation of the CFT FORTRAN version 1.14 compiler at the L.L.N.L. Computer Center, which can utilize the bi-directional memory and hardware gather/scatter capabilities of the X-MP. In total, we have observed average speed-ups of 35, 56 and 61 percent for direct solution, diagonal preconditioning and Crout EBE/PCG, respectively.

- Three-dimensional Boussinesq Problem

This classical linear problem concerns the application of a point load to the surface of a homogeneous, isotropic elastic halfspace. The simple material model and use of a uniform cubic mesh ($d_f = 3$) leads to a very well-conditioned problem. Results from a series of meshes are reported in [*Hughes 87*]; here we only consider a 24 × 24 × 24 element mesh with 45,000 degrees of freedom. Table 4 illustrates the overwhelming CPU and storage advantages of all the iterative methods. Each requires less than one minute of CPU as compared to about 35 minutes for direct solution, and even Crout EBE uses an order-of-magnitude less disk storage. The G-S EBE preconditioner uses five percent less CPU than Crout EBE, reflecting the absence of any factorization process. The two EBE methods use an average of 15 percent less CPU than diagonal scaling.

Method	Max. Shear $\frac{1}{2}(\sigma_1 - \sigma_3)$	Iterations	CPU (sec)	I/O (sec)	Core Storage	Disk Storage
Direct	910.9	—	2079	20.0	1,325,000	82,727,421
Diagonal	910.9	139	36.4	8.5	511,776	4,478,976
Crout EBE	910.9	57	31.7	10.5	556,776	8,957,952
G-S EBE	910.9	57	30.2	11.1	511,776	4,478,976

Table 4. Equation solving data for 24 × 24 × 24 Boussinesq problem.

- Thermal Stress Analysis of a Slab Laser Lens

The model in Fig. 8 was generated by Susarla Murty to study the thermal stress response of a proposed lens for the High Average Power (HAP) laser under development at L.L.N.L. Containing 18,029 nodes, 14,720 elements and 54,084 degrees of freedom, the model has a fractal dimension of 2.40. Although still linear, this problem presents a greater challenge to the iterative methods because of the distorted element geometries, and material moduli which vary over three orders of magnitude. Table 5 summarizes the computational cost of the analyses, and highlights the increased robustness of the EBE preconditioners relative to diagonal scaling. In this case the G-S EBE method shows only a marginal decrease in CPU usage, and the increased I/O charge is apparently an artifact of system overhead.

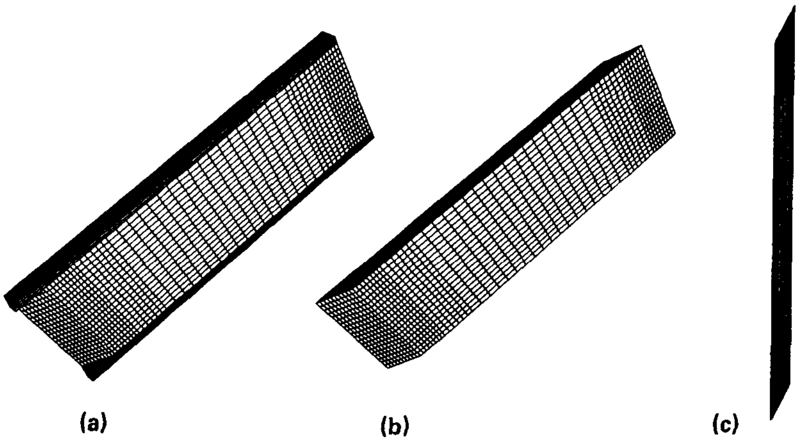

Fig. 8. Various views of HAP laser lens model: (a) full mesh, (b) and (c), rhombic lens with support rails removed.

Method	Max. $\sigma_1 \times 10^{-6}$	Iterations	CPU (sec)	I/O (sec)	Core Storage	Disk Storage
Direct	3.614	—	1159	11.5	817,348	64,073,517
Diagonal	3.615	1744	482	106	602,196	4,769,280
Crout EBE	3.618	670	358	117	656,280	9,538,560
G-S EBE	3.616	645	349	120	602,196	4,769,280

Table 5. Equation solving data for slab laser lens thermal stress analysis.

This problem illustrates the importance of the SSD for efficient solution of large problems with EBE/PCG. Using conventional CRAY DD49 rotating disks, the EBE I/O cost would approach 2.5 hours for this problem. With competition from other users for the available I/O ports, wall time of order 10 hours might be expected—a high price for less than nine minutes of CPU. Besides the equation solving CPU costs given in Table 5, it is important to note that the bandwidth minimization operation for direct solution requires an *additional* 442 seconds of CPU, while the element blocking for EBE uses less than 1 second.

- Rod Impact

This dynamic nonlinear analysis models the impact of a copper bar with an initial velocity of 227 m/sec onto a rigid wall. The model contains 3,367 nodes, 2,700 elments, 9,196 degrees of freedom and has a fractal dimension of 2.08. The mesh does not include any penalty slidesurfaces: nodal boundary conditions constrain the impacting face to lie upon the global X-Y plane. The computation is divided into 80 time steps of 1 microsecond and each method utilizes 81 stiffness formations. A linear iterative tolerance $\delta_L = 10^{-3}$ was specified. Execution data are summarized in Table 6. None of the iterative methods requires more than 25 percent of the direct method's CPU usage. The iterative data bases can be held in core, while FISSLE requires a modest 20 seconds of SSD I/O. In this example, G-S

EBE is only marginally better than diagonal scaling, and uses 19 percent *more* CPU than Crout EBE. This degradation appears to result from the substantial mesh distortion that develops with time (see Fig. 9). Over the first six time steps the Crout and G-S EBE methods average 14.9 and 17.9 iterations per linear solve, respectively. Subsequently, these averages rapidly increase to the values in Table 6 as the rod is crushed.

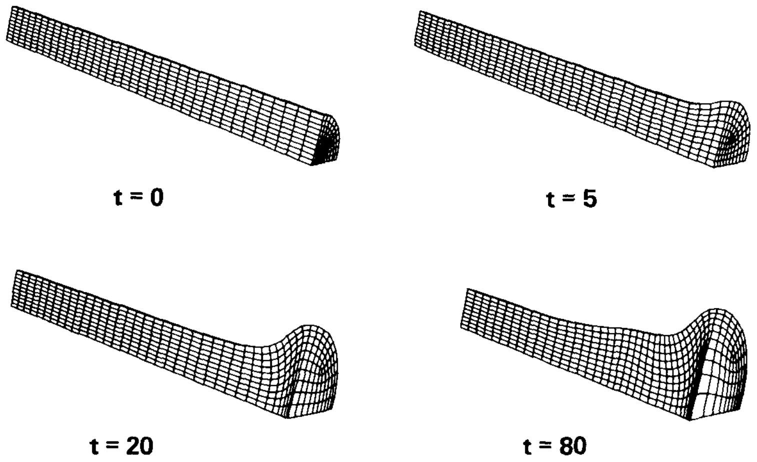

Fig. 9. Initial, intermediate and final deformation states for the rod impact analysis. Time in microseconds.

Method	Linear Solves	Avg. Iter. per Solve	Max. Iter. per Solve	Max. e^p	CPU (sec)	Core Storage	Disk Storage
Direct	193	—	—	2.249	1795	1,958,872	2,526,617
Diagonal	193	44.1	57	2.245	434	920,255	—
Crout EBE	195	16.8	22	2.248	371	1,739,976	—
G-S EBE	192	23.0	33	2.249	441	920,255	—

Table 6. Equation solving data for the rod impact analysis.

- Tunnel Intersection

The stress field about intersecting tunnels in geological material is under study at Los Alamos National Laboratory. The mesh in Fig. 10 was developed by R. Rosinsky of L.L.N.L. as a prototype to evaluate our iterative algorithms. Triple symmetry is used to model two 130 inch radius tunnels in a $600 \times 600 \times 600$ inch block of rock subjected to a uniform hydrostatic pressure of 100 psi. The mesh contains 11,713 nodes, 10,240 elements, 34,272 degrees of freedom with $d_f = 3.02$. The problem was first analyzed using a mesh with uniformly spaced elements; the resulting timings are listed in Table 7. With a well-conditioned problem, we would anticipate the number of EBE iterations to be on order of three times the maximum number of nodes along an edge (and six times for diagonal scaling). For the tunnel mesh this predicts $3 \times 25 = 75$ iterations, agreeing well with the Crout EBE data.

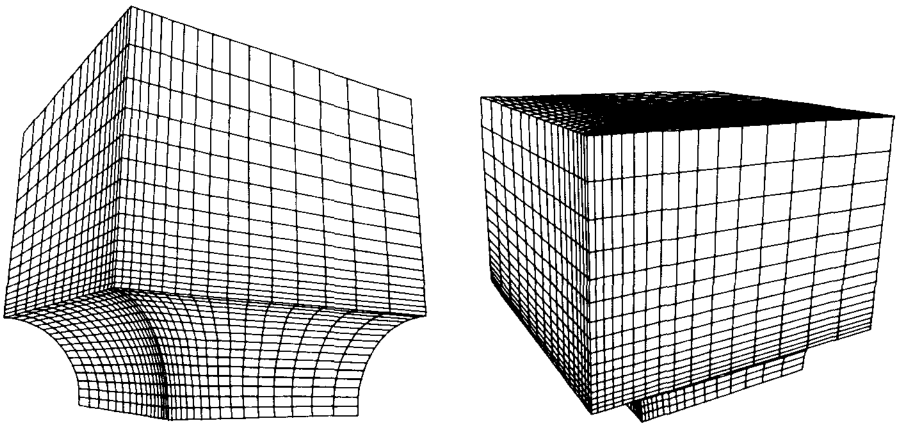

Fig. 10. Two views of tunnel intersection mesh with 10:1 element aspect ratio.

Method	Min. σ_3	Iterations	CPU (sec)	I/O (sec)	Core Storage	Disk Storage
Direct	−476.5	—	1836	8.8	1,432,928	53,973,073
Diagonal	−476.5	204	39.9	7.8	468,864	3,317,760
Crout EBE	−476.5	76	31.6	9.8	503,136	6,636,520
G-S EBE	−476.5	83	32.1	9.7	468,864	3,317,760

Table 7. Equation solving data for tunnel intersection with unitary element aspect ratios.

The poorer performance of G-S EBE and diagonal preconditioning is thought to arise from the modest element distortions resulting from the intersection geometry.

To study the effect of mesh distortion, a series of analyses were performed on meshes with evenly graded mesh transitions from the tunnel intersection outward. The measure of distortion was taken as the largest to smallest element length along an edge, which is nearly equivalent to the maximum element aspect ratio in the mesh. The mesh in Fig. 10 corresponds to an aspect ratio of ten, while we emphasize that the data in Table 7 are for an aspect ratio of one. The performance of the iterative methods is summarized in Fig. 11. The number of Crout EBE iterations increases at a slightly faster rate than diagonal scaling. G-S EBE displays the most distortion sensitivity. The graph of CPU time shows a crossover at an aspect ratio of 20, where diagonal scaling becomes the most efficient solver. G-S EBE is the least efficient method for aspect ratios greater than five. Although very large aspect ratios are not advisable in practice, the "survival" of diagonal scaling as the most efficient method runs counter to our expectations and previous experience. One may speculate that for such heavily distorted elements the regularized element matrices convey too little information to justify the expense of element forward reduction and back substitution.

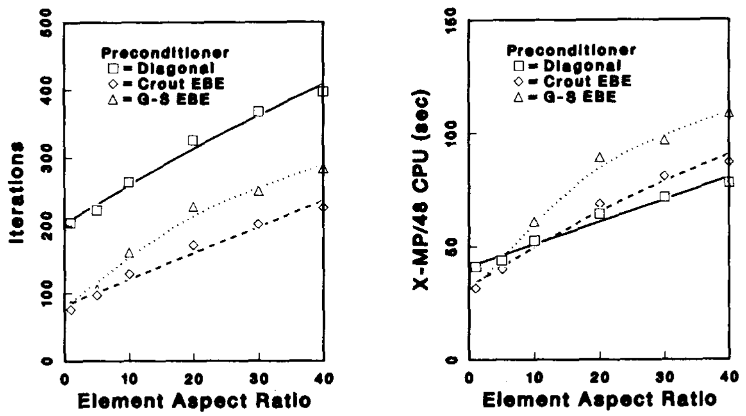

Fig. 11. Iterations and CPU cost for tunnel intersection analysis as a function of element aspect ratio.

Conclusions. We have presented two EBE/PCG algorithms capable of very significant reductions in the CPU and storage charges for equation solving compared with techniques based upon Gauss elimination in implicit finite element calculations. By utilizing a local, element-based data structure, these preconditioners should simplify the task of mapping complex meshes arising from engineering applications onto emerging multiprocessor architectures.

The Crout EBE procedure has been extended to the shell element and contact/impact algorithms within NIKE3D, and results to date indicate considerable increases in efficiency are attainable. The Crout EBE preconditioner has shown superiority over the diagonal preconditioner in most situations. It has been noted that diagonal preconditioning degrades markedly for symmetry plane elements whereas the Crout EBE preconditioner is essentially insensitive to the value of the penalty parameter. The solution of problems requiring many nonlinear iterations for convergence exposes a need to begin developing an adaptive iteration strategy that seeks to minimize the total cost of the nonlinear and linear iterative procedures.

The symmetrized Gauss-Seidel EBE preconditioner appears competitive with Crout EBE for moderately ill-conditioned problems while utilizing no more storage than diagonal preconditioning. It displays more distortion sensitivity than its Crout counterpart and presently this behavior is not understood. The Gauss-Seidel preconditioner has not yet been extended to contact/impact and shell problems. Given its more rapid degradation when faced with ill-conditioning, we are not optimistic that it will be successful in these situations.

The development of EBE-based methods is continuing within the production code environment of NIKE3D. We believe use of fractal dimension, among other parameters, will eventually provide an *a priori* indication of the regime for which iteration is more cost-effective than direct solution and will lead to automatic criteria for selecting the most appropriate equation solving algorithm. We also feel that considerable gains in efficiency can be registered by adopting the preconditioned Lanczos algorithms in place of PCG, as indicated in the recent studies of Nour-Omid and Raefsky [*Nour-Omid 86*].

Acknowledgement. The authors would like to express their sincere appreciation to Dr. Gerald L. Goudreau, Group Leader, Methods Development Group, L.L.N.L., for his support of the research presented herein. The first author also gratefully notes the support of Dr. Alan Kushner, Scientific Officer, U.S. Office of Naval Research.

References

[*Hallquist 83*] J. O. Hallquist, *NIKE2D—A Vectorized, Implicit, Finite Deformation, Finite Element Code for Analyzing the Static and Dynamic Response of 2-D Solids,* University of California, UCID-19677, 1983.

[*Hallquist 84*] J. O. Hallquist, *NIKE3D: An Implicit, Finite Deformation, Finite Element Code for Analyzing the Static and Dynamic Response of Three-Dimensional Solids,* University of California, UCID-18822, Rev. 1, 1984.

[*Hallquist 85*] J. O. Hallquist, G. L. Goudreau and D. J. Benson, *Sliding Interfaces with Contact-Impact in Large-Scale Lagrangian Computations,* Computer Methods in Applied Mechanics and Engineering, 51 (1985), pp. 107–135.

[*Hestenes 52*] M. R. Hestenes and E. Stiefel, *Method of Conjugate Gradients for Solving Linear Systems,* Journal of Research of the National Bureau of Standards, 49 (1952), pp. 409–436.

[*Hughes 79*] T. J. R. Hughes, K. S. Pister and R. L. Taylor, *Implicit-Explicit Finite Elements in Nonlinear Transient Analysis,* Computer Methods in Applied Mechanics and Engineering, 17/18 (1979), pp. 159–182.

[*Hughes 83a*] T. J. R. Hughes, I. Levit and J. M. Winget, *Implicit, Unconditionally Stable Algorithms for Heat Conduction Analysis,* Journal of the Engineering Mechanics Division, ASCE, 109 (1983) pp. 576–585.

[*Hughes 83b*] T. J. R. Hughes, I. Levit and J. M. Winget, *An Element-by-Element Solution Algorithm for Problems of Structural and Solid Mechanics,* Computer Methods in Applied Mechanics and Engineering, 36 (1983), pp. 241–254.

[*Hughes 83c*] T. J. R. Hughes, J. Winget, I. Levit and T. Tezduyar, *New Alternating Direction Procedures in Finite Element Analysis based upon EBE Approximate Factorizations,* in Computer Methods for Nonlinear Solids and Structural Mechanics, S. N. Atluri and N. Perrone eds., AMD-Vol. 4, ASME, New York, 1983, pp. 75–109.

[*Hughes 84a*] T. J. R. Hughes, A. Raefsky, A. Muller, J. M. Winget and I. Levit, *A Progress Report on EBE Solution Procedures in Solid Mechanics,* in Numerical Methods for Nonlinear Problems, Vol. 2, C. Taylor et al., eds., Pineridge Press, Swansea, U.K., 1984, pp. 18–26.

[*Hughes 84b*] T. J. R. Hughes, *Numerical Implementation of Constitutive Models: Rate-independent Deviatoric Plasticity,* in Theoretical Foundations for Large-Scale Computations for Nonlinear Material Behavior, S. Nemat-Nasser et al., eds., Martinus Nijhoff, Dordrecht, The Netherlands, 1984, pp. 29–57.

[*Hughes 87*] T. J. R. Hughes, R. M. Ferencz and J. O. Hallquist, *Large-scale Vectorized Implicit Calculations in Solid Mechanics on a CRAY X-MP/48 Utilizing EBE Preconditioned Conjugate Gradients,* Computer Methods in Applied Mechanics and Engineering, 61 (1987), pp. 215–248.

[*Lyzenga 87*] G. A. Lyzenga, A. Raefsky and B. H. Hager, *Finite Elements and the Method of Conjugate Gradients on a Concurrent Processor,* in Solving Problems on Concurrent Processors, Volume 2: Scientific and Engineering Applications, J. Fox and G. A. Lyzenga eds., Prentice-Hall, Englewood Cliffs, in press.

[*Newmark 59*] N. M. Newmark, *A Method of Computation for Structural Dynamics,* Journal of the Engineering Mechanics Division, ASCE, 85 (1959), pp. 67–94.

[*Nour-Omid 85*] B. Nour-Omid and B. N. Parlett, *Element Preconditioning Using Splitting Techniques,* SIAM Journal on Scientific and Statistical Computing, 6 (1985), pp. 761–770.

[*Nour-Omid 86*] B. Nour-Omid and A. Raefsky, *private communication*, 1986.

[*Ortiz 83*] M. Ortiz, P. M. Pinsky and R. L. Taylor, *Unconditionally Stable Element-by-Element Algorithms for Dynamics Problems,* Computer Methods in Applied Mechanics and Engineering, 36 (1983), pp. 223–239.

[*Taylor 81*] R. L. Taylor, E. L Wilson and S. J. Sackett, *Direct Solution of Equations by Frontal and Variable Band, Active Column Methods,* in Nonlinear Finite Element Analysis in Structural Mechanics, W. Wunderlich et al., eds., Springer-Verlag, Berlin, 1981, pp. 521–552.

[*Winget 85*] J. M. Winget and T. J. R. Hughes, *Solution Algorithms for Nonlinear Transient Heat Conduction Analysis Employing Element-by-Element Iterative Strategies,* Computer Methods in Applied Mechanics and Engineering, 52 (1985), pp. 711–815.