NASA Technical Memorandum 4072

CSM Testbed Development and Large-Scale Structural Applications

N. F. Knight, Jr., and R. E. Gillian
Langley Research Center
Hampton, Virginia

S. L. McCleary and C. G. Lotts
PRC Kentron, Inc.
Aerospace Technologies Division
Hampton, Virginia

E. L. Poole and A. L. Overman
Awesome Computing, Inc.
Charlottesville, Virginia

S. C. Macy
PRC Kentron, Inc.
Aerospace Technologies Division
Hampton, Virginia

NASA
National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Division

1989

The use of trademarks or names of manufacturers in this report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.

Trademarks

UNIX is a registered trademark of AT&T. CRAY and UNICOS are registered trademarks and CRAY-2 and CFT77 are trademarks of Cray Research, Inc. VAX, MicroVAX, and VMS are trademarks of Digital Equipment Corp. PATRAN is a registered trademark of PDA Engineering. IRIS is a trademark of Silicon Graphics, Inc. FORGE is a trademark of Pacific Sierra Research.

Abstract

A research activity entitled computational structural mechanics (CSM) at the NASA Langley Research Center is described. This activity involves developing advanced structural analysis and computational methods that exploit high-performance computers. New methods are developed in the framework of the CSM Testbed software system and applied to representative complex structural analysis problems from the aerospace industry. An overview of the CSM Testbed methods development environment is presented and some new numerical methods developed on a CRAY-2 computer system are described. Selected application studies performed on the NAS CRAY-2 computer system are also summarized.

Introduction

Research in computational methods for structural analysis is encumbered by the complexity and cost of the software development. In addition, new computer architectures with vector and multiprocessor capabilities are being manufactured to provide increased computational power. Analysis and computational algorithms that can exploit these new computer architectures need to be developed. These new algorithms should be developed and evaluated in a standard, general-purpose, finite-element structural analysis software system rather than in an isolated research software system so they can be evaluated on large-scale application problems as well as on small verification problems.

At the NASA Langley Research Center (LaRC), a research effort is being directed towards developing advanced structural analysis methods and identifying the requirements for next-generation structural analysis software which will exploit multiple vector processor computers (ref. 1). This activity, called computational structural mechanics, or CSM (ref. 2), has resulted in the development of the CSM Testbed software system (e.g., Lotts et al. (ref. 3)) to aid in the definition of these requirements and to serve as a "proving ground" for new methods for large-scale structural application problems. This research activity makes extensive use of the computational facilities provided by the Numerical Aerodynamic Simulation (NAS) Program at the NASA Ames Research Center (ref. 4).

This paper describes the implementation experiences, the resulting capability, and the future directions for the CSM Testbed on supercomputers. The distributed nature of the computing hardware environment is described herein and its use demonstrated. The flexibility of the CSM Testbed, coupled with the computational facilities available through the NAS system, makes it possible for structural analysts, method developers, numerical analysts, and computer scientists to integrate their research in a common, shared computing environment. The powerful, problem-solving capability of this computing environment is demonstrated in the solution of several structural application problems involving linear and nonlinear stress analysis, buckling analysis, and transient dynamics analysis.

Overview of CSM Testbed

The field of computerized structural analysis is dominated by two types of computer programs. One type is the huge, 2000-subroutine, general-purpose program (ref. 5) that is the result of over 100 man-years of effort spanning more than a decade. The other type is the relatively small, special-purpose code resulting from a research environment that represents a 1- to 2-year effort for a specific research application. This dichotomy has resulted in long delays in making research technology available for critical structural analysis problems that NASA faces. To accelerate the introduction of successful research technology into large-scale applications programs, a modular, public-domain, machine-independent, architecturally simple software development environment has been constructed. This system is denoted the CSM Testbed. One goal of the CSM Testbed is to provide a common structural analysis environment for three types of users—engineers solving complex structures problems, researchers developing advanced structural analysis methods, and developers designing the software architecture to exploit multiprocessor computers.

The CSM Testbed software system is a highly modular and flexible structural analysis system for studying computational methods and for exploring new multiprocessor and vector computers. The CSM Testbed is used by a group of researchers from universities, industry, and government agencies. Unrestricted access to all parts of the code, including the data manager and the command language, is permitted. Research on these elements of software design is needed because deficiencies in the data management strategy can have a devastating impact on the performance of a large structural analysis code, totally masking the relative merits of competing computational techniques. Furthermore, software designs that exploit multiprocessor computers must be developed; in particular, techniques for handling parallel input/output (I/O) are required.

The CSM Testbed is public-domain software, and source code is available. The initial CSM Testbed, called NICE/SPAR, began with the integration of the NICE system (refs. 6 and 7) and Level 13 of

SPAR (ref. 8). Since then, new capabilities and improvements have been implemented in the CSM Testbed. A brief description of selected CSM Testbed processors is given in table 1. Each step of the evolution of the CSM Testbed provides improved structural analysis capabilities to structural analysts.

Distributed Computer Environment

Distributed computer environments are made up of stand-alone computers of different sizes, architectures, and vendors, with a common network protocol offering the user easy file transfer and remote login functions. Structural analysts require the diverse computer capabilities offered by a distributed environment (workstation-mainframe-supercomputer) but cannot afford the "overhead" of learning the operating system commands for each system they use. Applications developers have a similar problem, but at a lower level. They cannot afford the overhead of learning a new set of system calls for each computer on which they wish to implement their application code. The CSM Testbed, as depicted in figure 1, addresses these problems. The inner circle, the computer-specific operating system, is provided by the computer vendor and is different for each vendor. The outer ring, the applications development environment, insulates both the user and the applications developer from those differences by providing a consistent interface. The methods development environment of the CSM Testbed is described by Gillian and Lotts (ref. 9).

The computing environment of the CSM activity is currently a distributed environment, as shown in figure 2. Typically, a structural analyst will develop a finite-element model of the structure either by using a preprocessing software system such as PATRAN or by using CLAMP (command language for applied mechanics processor) for "parameterizing" the model. Run streams are the vehicle used to perform structural analyses with the CSM Testbed. The term run stream most commonly refers to the file (or files) of input data and commands used to perform a specific analysis, although it may also refer to input at an interactive session. Run streams for the CSM Testbed are usually developed, verified on a workstation, and then transferred to the NAS CRAY-2 computer system for complete processing. Following a successful execution, the computational data base may then be "unloaded" (i.e., converted from the binary format of the NAS CRAY-2 computer system to the ASCII format), transferred intact to LaRC using the NASnet wide-area network, and then "loaded" (i.e., converted from ASCII format to the binary format of the desired workstation) back into a computational data base which has the identical Testbed library format as on the NAS CRAY-2 computer system. Finally, postprocessing is done to help the structural analyst visualize the computed structural response. The sequence of steps just described depicts the computing environment to which the structural analyst must adapt in order to exploit the full potential of these computing systems.

To exploit this new computing environment, expertise is needed in the areas of computational strategies, numerical techniques, computer science, and communication networks, together with a firm understanding of the principles of structural mechanics. New computing hardware environments, like the NAS system, offer the computational power, memory, and disk space necessary for routine analysis of large structural models. New computing software environments, like the CSM Testbed, offer an integrated system with data management, a general command language, and many different application processors—features that enable the structural analyst to develop new analysis methods and to tailor the analysis for specific application needs.

CSM Testbed Architecture Features

The CSM Testbed is a Fortran program organized as a single, executable file (called a macroprocessor) which calls structural applications modules that have been incorporated as subroutines. The macroprocessor and applications modules interface with the operating system for their command input and data management functions through a set of common "architectural utilities." Processors access the Testbed utilities by calling entry points implemented as Fortran-77 functions and subroutines which are available to module developers in the Testbed object libraries. Applications processors do not communicate directly with each other, but instead communicate by exchanging named data objects in a data base managed by a data manager called GAL (global access library). The user controls the execution of applications processors using an interactive, or batch, command run stream written in a command language for applied mechanics processors (CLAMP), which is processed by the command language interpreter program (CLIP).

Command Language

The Testbed command language CLAMP is a generic language originally designed to support the NICE system and to offer program developers the means for building problem-oriented languages (ref. 10). It may be viewed as a stream of free-field command records read from an appropriate command source (the user's terminal, actual files, or

Table 1. Selected CSM Testbed Processors

Processor	Description
ELD	Element definition (connectivity, material properties, etc.)
LAU	Laminate analysis utility for 2-D and 3-D elements
E	Element-state initiation (builds element information packets)
EKS	Computes the element intrinsic stiffness matrices
TOPO	Analyzes the finite-element mesh topology and build tables to drive assembly and factorization of system matrices
RSEQ	Renumbers nodes for minimum fill or minimum bandwidth
AUS	Arithmetic utilities
K	Assembles unconstrained system stiffness matrix
M	Assembles unconstrained system mass matrix
INV	Applies constraints and factors assembled system matrix
SSOL	Performs forward reduction and back substitution
BAND	Factors and solves using profile or banded solvers
ITER	Factors and solves using iterative solvers
KG	Forms and assembles unconstrained system geometric stiffness matrix
EIG	Solves linear algebraic eigenproblems
ES	Generic element processor shell
VEC	Performs variety of vector algebra operations

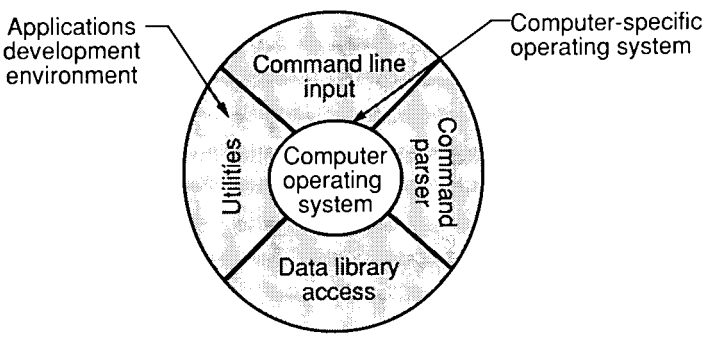


Figure 1. Organization of CSM Testbed software.

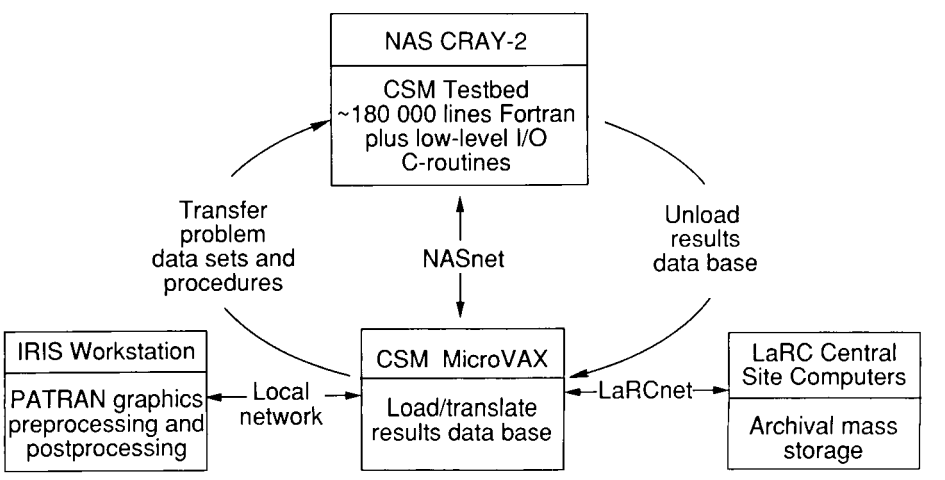


Figure 2. Distributed computing environment of CSM.

processor messages). The commands are interpreted by a "filter" utility called CLIP, whose function is to produce object records for use by its user program. The standard operating mode of CLIP is the processor-command mode. Commands are directly supplied by the user, retrieved from ordinary card-image files or extracted from the global data base, and submitted to the running processor. Special commands, called directives, are processed directly by CLIP; the processor is "out of the loop." Transition from processor-command to directive mode is automatic. Once the directive is processed, CLIP returns to processor-command mode. Directives are used to dynamically change run-environment parameters, to process advanced language constructs such as macrosymbols and command procedures, to implement branching and cycling, and to request services of the data manager. CLIP can be used in this way to provide data to a processor as well as to control the logic flow of the program through a single input stream. All command language directives are available to any processor that uses the CLIP-processor interface entry points.

Data Manager

The data manager within the CSM Testbed was derived from the global access library (GAL) concept developed at the Lockheed Palo Alto Research Laboratory (ref. 11). Methods for data management in structural analysis programs can be divided into three levels of complexity: file systems, file partition systems, and data-base systems (ref. 12). Since data-base files are subdivided or partitioned into data sets, the Testbed data manager is classified as a file partition manager. To a processor, a GAL data library is analogous to a file. It must be opened, written, read, closed, and deleted explicitly. The GAL resides on a direct-access disk file and contains a directory structure called a table of contents (TOC), through which specific data sets may be addressed. Low-level I/O routines access the GAL library file in a word-addressable scheme, as described by Felippa (ref. 13). The data management system is accessible to the user through the command language directives and to the running processors through the GAL-processor interface.

The global data base is made up of sets of data libraries (GAL's) residing on direct-access disk files. Data libraries are collections of named data sets, which are collections of data set records. The data library format supported by the Testbed is called GAL/82, which can contain nominal data sets made up of named records. Some of the advantages to using this form of data library are (1) the order in which records are defined is irrelevant, (2) the data contained in the records may be accessed from the command level, and (3) the record data type is maintained by the manager, and this simplifies context-directed display operations and automatic type conversion.

To provide the efficiency required to process the volume of data required for a complex structural analysis, all usual overhead associated with Fortran has been eliminated. The actual I/O interface between the GAL data manager and the UNIX operating system is accomplished through a set of block I/O routines written in the C programming language. For non-UNIX computer systems, this interface is accomplished through a set of assembly language routines which are unique to each computer system.

Interprocessor Control

The SuperCLIP capability of the Testbed architecture performs interprocessor control, which allows independent programs which use the Testbed architecture facilities (CLIP and GAL) to be executed from within a single Testbed run stream. SuperCLIP handles the interprocessor CLIP-state preservation and restoration so that the CLIP environment is maintained across independent program executions. These independent programs can be used in conjunction with the Testbed macroprocessor, with other independent Testbed processors, or entirely alone, as appropriate to accomplish the required task. The implementation of SuperCLIP is the most complex and machine-dependent element of the Testbed architecture software. To date, it has been implemented under the VAX/VMS and the UNIX operating systems.

User Interface

The user may develop run streams using the high-level command language CLAMP for a specific engineering problem (ref. 10). These run streams may contain CLAMP directives and CLAMP procedures which are processed by the command language interpreter CLIP. Applications processors are called using the `[XQT` command, or the GAL (e.g., ref. 11) may be interrogated. Engineers typically interact with the Testbed using simple run streams or through CLAMP procedures. Researchers interact using CLAMP procedures (e.g., to study nonlinear solution strategies) or through Fortran processors (e.g., to implement new element formulations). Developers interact with the entire Testbed architecture, including the design of the command language, the data handling techniques for large-scale analyses, and the strategy for I/O on parallel computers.

CSM Testbed Structural Analysis Features

The CSM Testbed presently provides structural analysis capabilities that permit an analyst to perform large-scale nonlinear stress analyses of shell-type structures. Three-dimensional stress analyses are presently limited to linear elastic orthotropic materials. Eigenvalue problems associated with either linear bifurcation buckling or linear vibration analyses may also be solved. Transient dynamic analyses are limited to linear elastic problems with either direct time integration or mode superposition used to obtain the transient response. Some of the newly developed engineering features of the CSM Testbed are the equation solvers, the element library, the material modeling, and the solution procedures. Interface utilities to and from the PATRAN graphics systems have been developed to support the modeling and analysis of large-scale structures. Access to such a preprocessing and postprocessing software system enhances the structural analyst's ability to understand the structural behavior through visualization of the computed results.

Equation Solvers

The linear system of equations that arise in static structural analysis applications has the form $\mathbf{Ku} = \mathbf{f}$, where $\mathbf{K}$ is the symmetric, positive-definite stiffness matrix, $\mathbf{f}$ is the load vector, and $\mathbf{u}$ is the vector of generalized displacements. Such linear systems can be as large as several hundred thousand degrees of freedom (dof) and often require significant computing resources (both memory and execution time). The structure of the stiffness matrices in these applications is often sparse, although in many applications an ordering of the nodes which minimizes the bandwidth makes banded or profile (skyline) type storage of the matrices practical. The choice of a particular method to solve $\mathbf{Ku} = \mathbf{f}$ will depend on the nonzero structure of $\mathbf{K}$ and, in the case of the iterative methods, the condition number of $\mathbf{K}$. In addition, the architecture of the computer, particularly for modern vector and parallel computers, influences both the choice and the implementation of methods used to solve these linear systems of equations. Ortega (ref. 14) presents a thorough description of these various methods and their implementations as applied to vector and parallel computers.

The data structure of the global stiffness matrix is a key factor in the design and implementation of equation solvers for the CRAY-2 architecture and the Testbed software (e.g., ref. 15). The generation of stiffness matrices is accomplished by several different processors producing element stiffness matrices, defining boundary conditions, applied loads, and ordering of nodes, and assembling the global stiffness matrix. The stiffness matrices are stored in a nodal-block sparse form. The original sparse out-of-core Choleski solver used by the Testbed code (processors INV and SSOL) factors and solves the stiffness matrices using this data structure. A major source of inefficiency for this equation solver on a CRAY-2 computer system is that the operations carried out in factoring the stiffness matrix and solving the resulting triangular systems are carried out using these small nodal blocks (usually 3×3 or 6×6 in size). The vector length of these operations is therefore six or less and the code is faster when run without vector optimization.

The new vectorized equation solvers (processors BAND and ITER) require $\mathbf{K}$ to be stored in one of several different sparse and banded storage schemes. Processor ITER contains three conjugate gradient iterative methods. These methods vary in their types of preconditioning, which include diagonal scaling, incomplete Choleski factorization with a sparse storage scheme, and incomplete Choleski factorization with a diagonal storage scheme. Processor BAND contains three basic algorithms that are all based on Choleski factorization of banded matrices. The first algorithm uses the standard LINPACK routines (ref. 16) for banded solvers, namely SPBFA and SPBSL. The second algorithm, *kji* Choleski, uses column storage of the lower triangular part of the symmetric matrix to take advantage of vectors with a constant stride of one and loop unrolling to a depth or level of four. Loop unrolling reduces the number of memory references by holding vectors longer in the registers and increases the amount of vector computations within a loop. As a result, many of the multiplication and subtraction operations and memory references will overlap, leading to greater performance. In addition, the local memory of the CRAY-2 computer system is used to store up to four columns of the factored matrix to further decrease execution time. The third algorithm uses profile storage of the matrix instead of banded storage, and this type of storage results in a significant reduction in memory requirements and in the number of operations.

The strategy used for the vectorized equation solvers involves four steps. First, the coefficients of the unconstrained stiffness matrix are read from the global data base into a temporary array. Second, the nodal constraint information and node ordering sequence information are retrieved from the global data base. Third, the appropriate pointer arrays for the new storage scheme are formed. Finally, the coefficients of $\mathbf{K}$ are placed in a singly dimensioned array and modifications are made to the right-hand side ($\mathbf{f}$) corresponding to any applied displacements. For the direct Choleski methods, an additional storage

scheme is included to reformat Testbed stiffness matrices into the standard LINPACK (ref. 16) banded storage format. The reformatting procedure is essentially sequential, but the time to reformat the matrices is small compared with the time to solve the equations for large problems.

The capability to reorder the nodes automatically is an important part of the equation solving process in general-purpose finite-element codes. The structure of the assembled stiffness matrices is determined by the node connectivities and the node numbering scheme used in the finite-element model. Although the node connectivity is fixed by the problem definition and discretization, many node orderings are possible. The Testbed software contains processor RSEQ, which uses four different algorithms to reorder nodes automatically. These algorithms are nested dissection, minimum degree, reverse Cuthill-McKee, and Gibbs-Poole-Stockmeyer. The first two algorithms are used by sparse solvers and minimize fill in the factorization process. The last two are profile and bandwidth minimizing routines, respectively. The direct banded solvers implemented in processor BAND are most efficient with node orderings which minimize bandwidth, while the sparse out-of-core Choleski equation solver in processor INV is most efficient with orderings which minimize fill. For the various preconditioned conjugate gradient methods in processor ITER, the preconditioner used determines which ordering is best. Although the precise relationship between node ordering and the convergence rate of the incomplete Choleski conjugate gradient (ICCG) is not known, preliminary results show that the ordering of nodes can have a great effect on the convergence rate. In the test problems used with the ICCG method, the convergence rate of ICCG is better for the sparse, minimum-fill orderings than for the bandwidth-minimizing orderings. However, in some cases, the ordering used to define the problem gives the best convergence rate. For the basic conjugate gradient method, the matrix structure has no effect on the convergence rate but the matrix structure is important for the storage requirements if diagonal storage is used. Orderings which minimize bandwidth also concentrate the coefficients near the main diagonal, thereby minimizing the number of diagonals required for matrix storage. As a result, the vector lengths of the diagonals are longer and the number of extra zeros added between nonzero coefficients is fewer; thus, the memory requirements are reduced and the computation speed is increased.

Generic Element Processor Template

The generic element processor template shown in figure 3 provides the element developer with a standard outer software "shell" that handles all user-command input and all I/O to and from the global data base. In addition, a standard set of "shell-to-kernel" interface routines (e.g., ES_K, ES_M, and ES_F) are provided as cover routines for the element developer's "kernel" routines. The function of the interface routines is to perform the transformation between the standard argument lists of the outer software "shell" and those of the element developer's personal code. The element developer's kernel routines are integrated with these interface routines through the convention that the interface subroutine names and argument lists are standardized. The independent structural element processors (i.e., processor ESi, where i = 1,2,...) are installed and readily accessible to all CSM researchers for small benchmark problems as well as large-scale application problems.

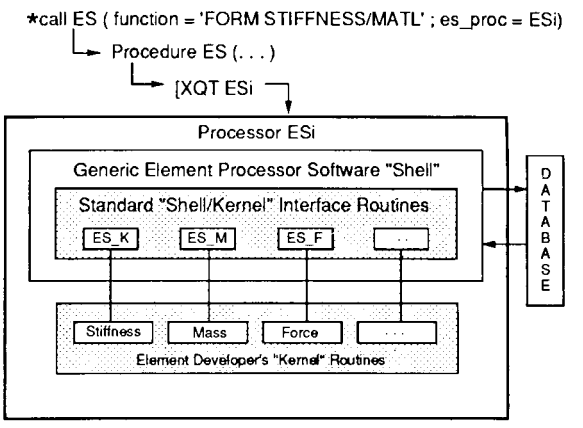


Figure 3. Generic element processor template.

The generic element processor features a standard high-level procedure ES that processes user commands such as

```
*call ES ( function='FORM STIFFNESS/MATL';--
    es_proc=ESi)
```

All the ESi processors are driven by a common set of commands through calls to the ES procedure and create the same data structure, regardless of how the element developer programmed the kernel routines (e.g., the element stiffness calculations and element stress recovery). This approach provides an extendible and easy-to-use vehicle for integrated finite-element research, development, and application within the CSM Testbed.

A key feature of the generic element processor shell is the easy access to the utilities associated with an element-independent corotational formulation (ref. 17). Through these utilities, element

developers may readily attempt geometric nonlinear problems which exhibit large rotations. Only the basic element characteristics associated with linear strain-displacement relations are required from the element developer in the kernel routines. Extensions to include the nonlinear strain-displacement relations require the element developer to provide additional kernel routines (e.g., internal force calculations).

Presently only two-dimensional shell elements and three-dimensional solid elements have been installed in the CSM Testbed with the generic element processor template. Processor ES1 contains a family of four- and nine-node continuum-based resultant (CBR) quadrilateral shell elements (ref. 18). This family of elements includes the assumed-natural coordinate strain quadrilateral shell elements (ref. 19) and the Lagrangian quadrilateral shell elements with selectively reduced integration. Processor ES2 contains a new hybrid curved four-node quadrilateral shell element (ref. 20). Processor ES3 contains a family of three-dimensional hybrid solid elements, including 8- and 20-node bricks (hexahedrons), 6- and 15-node wedges (pentahedrons), and 4- and 10-node pyramids (tetrahedrons). Processor ES4 contains a family of hybrid plate-shell elements, including four-node quadrilateral and three-node triangular elements. Processor ES5 contains a displacement-based, four-node quadrilateral plate-shell element, denoted the 410 element, from the STAGSC-1 computer code (ref. 21). Additional ES*i* processors are under development. In addition, elements in the original element library of Level 13 of SPAR (ref. 8) are currently still available for linear analyses.

Material Modeling

The material modeling features of the CSM Testbed are directed toward the analysis requirements of laminated composite structures. Constitutive relations for classical and shear flexible two-dimensional plate and shell models as well as for three-dimensional solids are evaluated and available to the element developer or structural analyst. Processor LAU is a laminate analysis utility for calculating the constitutive relations for two-dimensional and three-dimensional isotropic, orthotropic, and laminated structures. The formulation is based on the usual lamination theory (e.g., refs. 22 and 23) whereby the laminate constitutive relations are derived from the constitutive relations for each layer in the laminate. With the midplane strains and curvatures, the in-plane strains and corresponding stresses in each layer of the laminate may be calculated and used to evaluate selected stress- and strain-based failure criteria. The failure criteria implemented in the Testbed include the maximum stress criteria, the maximum strain criteria, and several quadratic polynomial failure criteria.

Solution Procedures

Various types of analysis may be performed with the CSM Testbed through the use of either ordinary run streams which execute various processors sequentially or CLAMP procedures which execute directives and processors and perhaps call other procedures. Linear stress analyses and eigenvalue analyses are both performed using simple analysis run streams. Solution procedures that require looping and branching are more complex procedures than linear analysis procedures. Two sets of solution procedures that require looping have been written and may be used to solve various application problems. The first solution procedure is named NEWMARK. Its function is to perform a linear transient dynamic analysis using the well-known Newmark method for direct time integration of the equations of motion. The second set of procedures is named NL_STATIC_1. These procedures are used to perform a geometric nonlinear static analysis using a modified Newton-Raphson algorithm with corotational updates and the Riks–linearized-Crisfield arc-length control strategy (refs. 24 and 25) for either applied force or applied displacement problems.

Application Studies Using CSM Testbed

Research in methods development for the CSM Testbed is driven in part by the analysis deficiencies identified in the solution of various application problems. The LaRC CSM activity uses the concept of focus problems to provide a common set of structural analysis problems for all CSM participants. Focus problems may be entire aerospace vehicles or various subcomponents that pose difficult computational and structural mechanics problems. These focus problems help guide methods research and development for generic classes of problems. New focus problems are selected as new technology evolves and computational structural mechanics methodology develops. A wide range of CSM application studies are presented in reference 26. Problems selected for presentation here are the following:

- Composite blade-stiffened panel with discontinuous stiffener
- Circular cylindrical shell with two rectangular cutouts
- Impulsively loaded truncated conical shell
- Space Shuttle solid rocket booster

These application studies demonstrate the structural analysis capabilities of the CSM Testbed. The

analyses presented herein utilize solution procedures implemented through the CLAMP language as well as various finite elements implemented through the generic element processor template. The execution times for selected CSM Testbed processors are compared for the various analysis problems considered. Postprocessing of the results, including both deflections and stress resultants, is performed with PATRAN to help the analyst visualize the computed results, and examples of this capability are also presented.

Composite Blade-Stiffened Panel With Discontinuous Stiffener

Discontinuities and eccentricities are usually present in practical structures. In addition, potential damage of otherwise perfect structures is often an important design consideration. Predicting the structural response in the presence of discontinuities, eccentricities, and damage is particularly difficult when the component is built from brittle composite materials or is loaded into the nonlinear range. The nonlinear response of a flat, blade-stiffened graphite-epoxy panel with a discontinuous stiffener loaded in axial compression (fig. 4(a)) was chosen as a focus problem and is summarized in this section. A more complete discussion of this problem is presented in reference 26.

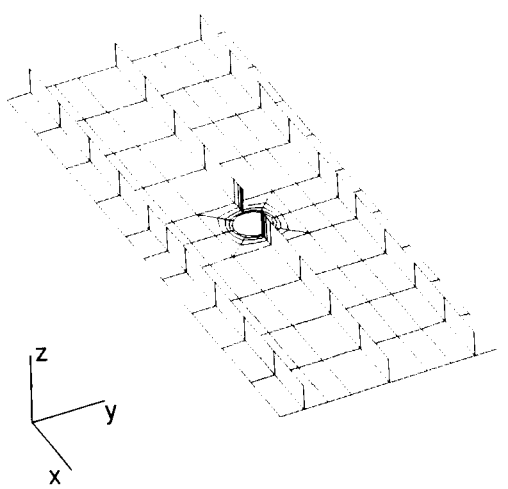
This problem represents a generic class of laminated composite structures with discontinuities in which the interlaminar stress state becomes important. This problem is characterized by a discontinuity (the hole), an eccentric loading, large displacements, large stress gradients, and a brittle material system. The geometry and laminate properties are given in reference 26. The loading is uniform axial compression. The loaded ends of the panel are clamped and the sides are free.

The finite-element model is shown in figure 4(b). A total of 144 9-node quadrilateral shell elements (processor ES1) are used in the nonlinear analysis. This model has 628 nodes and 2910 active degrees of freedom. The procedure NL_STATIC_1 is used to perform the nonlinear analysis.

End-shortening results are shown in figure 5 as a function of the applied compressive load. The end shortening u is normalized by the overall panel length L, and the applied load P is normalized by the panel prebuckling extensional stiffness EA obtained from the test. The blade-stiffened panel with a discontinuous stiffener was tested to failure. Local failures occurred prior to overall panel failure, as is evident from the end-shortening results shown in figure 5. Good agreement between test and analysis is shown up to the load where local failures occurred.

L-79-7347

(a) Stiffened panel.



(b) Finite-element model.

Figure 4. Composite blade-stiffened panel with discontinuous stiffener.

Oblique views of two deformed shapes with exaggerated deflections are shown in figure 6 for two values of applied compressive load. Load A corresponds to approximately half the value of load B (37 800 lb). Contour plots of the longitudinal in-plane stress resultant N_x are also shown in figure 6. These N_x distributions reveal several features of the global structural behavior of this panel. First, away from the discontinuity, the N_x distribution in the panel skin is nearly uniform and approximately half the value

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

ORIGINAL PAGE
COLOR PHOTOGRAPH

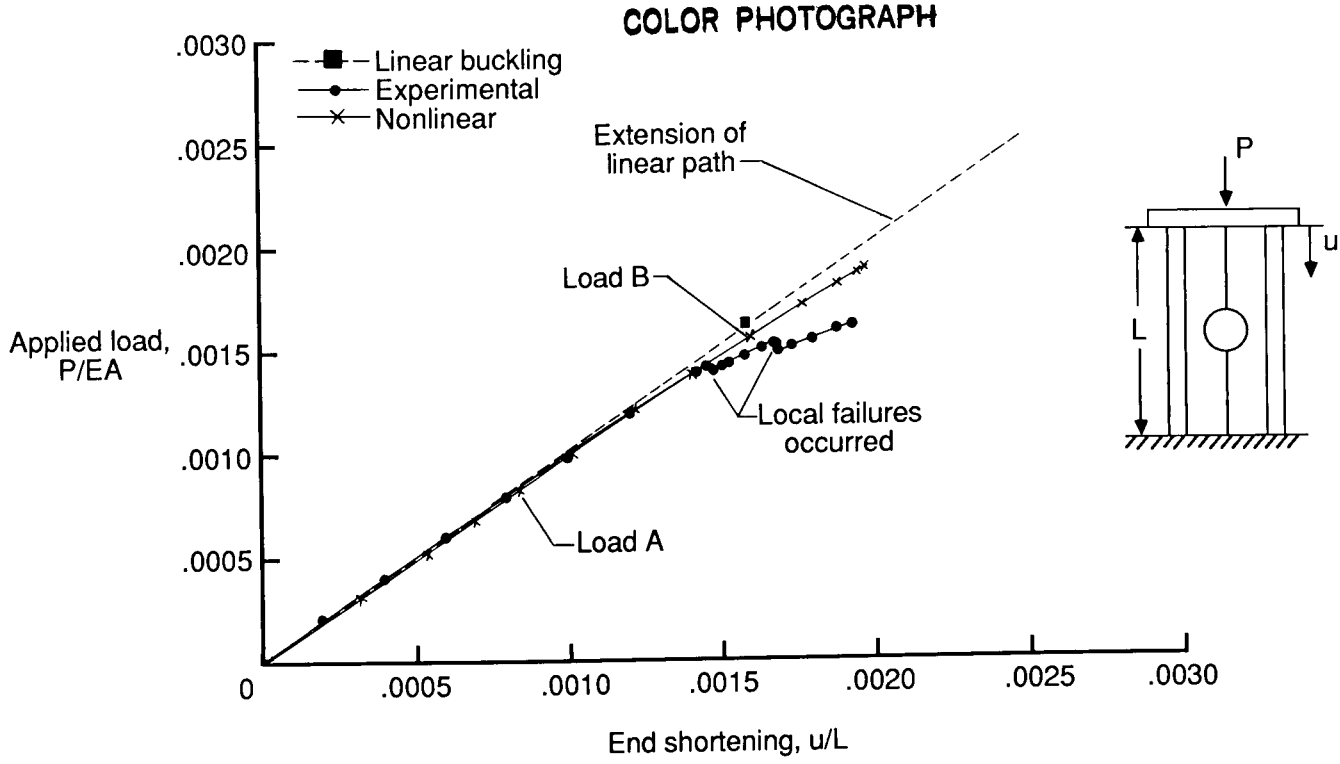


Figure 5. End-shortening results for blade-stiffened panel.

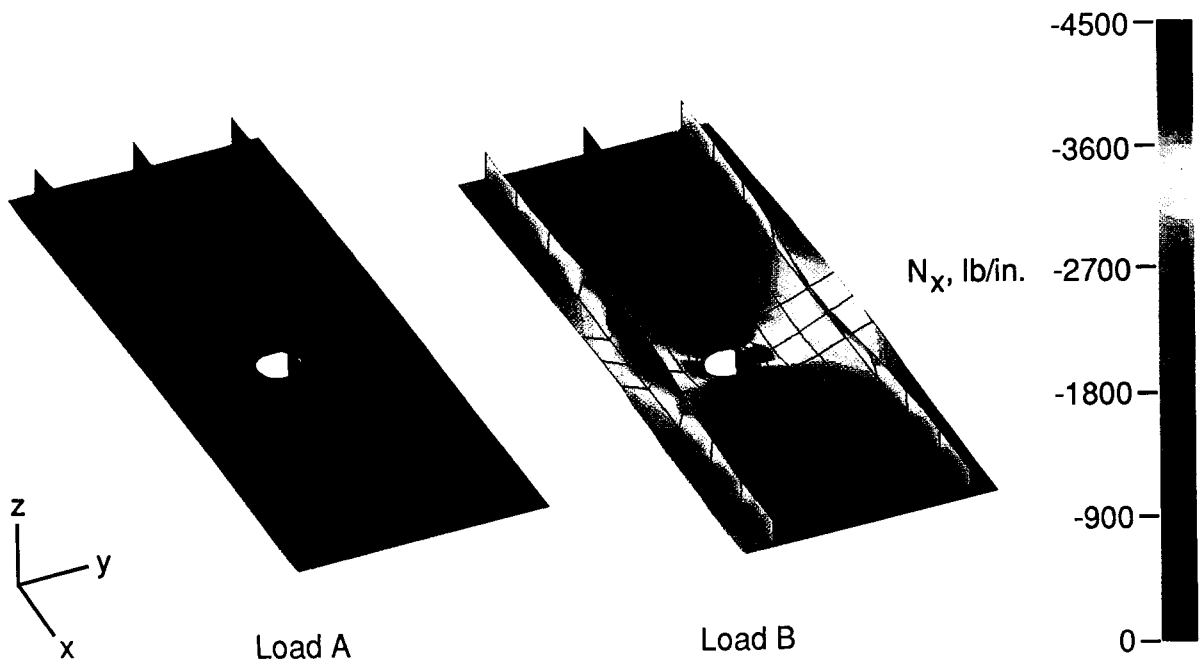


Figure 6. Longitudinal in-plane stress resultant N_x distributions.

of N_x in the two outer blade stiffeners. Second, the load is diffused from the center discontinuous stiffener into the panel skin rapidly such that the center stiffener has essentially no N_x load at the edge of the hole. Third, the N_x load in the two outer stiffeners increases towards the center of the panel and, because of panel bending, is concentrated in the blade tips. Fourth, the N_x load in the panel skin near the center of the panel is much greater than the N_x load in other portions of the panel skin.

Longitudinal in-plane stress resultant distributions at panel midlength are shown in figure 7 as a function of distance from the hole. The results indicate that high in-plane stresses and a high stress gradient exist near the hole. As the load increases, both the longitudinal in-plane stress resultant and the stress gradient increase near the hole and the blade stiffeners. These high in-plane stresses and stress gradients coupled with the large out-of-plane displacements at the free edge of the hole may cause material nonlinearities, local failures, and/or delaminations to develop.

Computation times for the nonlinear analysis of the composite blade-stiffened panel with a discontinuous stiffener are given in table 2 for selected Testbed processors. The ratio of the overall execution time on a VAX 11/785 computer system to the execution time on the NAS CRAY-2 computer system is 17.2 for the complete nonlinear analysis of the composite blade-stiffened panel with a discontinuous stiffener. The global tangent stiffness matrix was reevaluated and factored 15 times, and a total of 64 iterations were required to predict the nonlinear structural response of this panel. Most of the CPU time was spent in processors that perform computations on the element level such as processor ES1, which computes new elemental tangent stiffness matrices. The ratio of the execution times for the evaluation of the elemental stiffness matrices is lower than the ratio of the overall execution time. These processors (e.g., ELD and ES1) have not been modified to exploit the features of vector computers. However, processor INV has been modified and performs 41.8 times faster on the NAS CRAY-2 computer system than on a VAX 11/785 computer system.

Performance results obtained with various direct solvers implemented in processor BAND are shown in table 3. Increased performance is obtained by using "loop unrolling" to level 4, where a column of **K** is updated by four columns at a time rather than one, and by also exploiting the local memory of the CRAY-2 computer system. For this problem, only the profile method in processor BAND performs better than processor INV.

Circular Cylindrical Shell With Two Rectangular Cutouts

A common structural configuration is that of a cylindrical shell (e.g., storage tanks, pipelines, aircraft fuselages, and rocket motor cases). Shell-type structures are generally sensitive to initial geometric imperfections and to local discontinuities such as cutouts. Many aerospace vehicles contain large cutouts (e.g., access holes and windows). The strength of these structures is limited to the static collapse load. Predicting the nonlinear collapse behavior of these shell structures is a difficult and computationally intensive analysis problem.

The circular cylindrical shell with two rectangular cutouts loaded by uniform end shortening shown in figure 8 is representative of this class of structures. This problem has also been used as a benchmark problem by Hartung and Ball (ref. 27) for shell analysis computer codes and by Almroth and Brogan (ref. 28) for assessing shell elements. These researchers considered only one-eighth of the shell in their analyses. The results reported herein are compared with their results, and hence only one-eighth of the shell is modeled. The finite-element model is composed of 101 9-node quadrilateral shell elements (processor ES1), 449 nodes, and 2012 active degrees of freedom, as shown in figure 9(a).

A linear bifurcation buckling analysis was performed prior to the nonlinear collapse analysis. The buckling load computed in this study is 1016 lb which agrees with the results presented by Hartung and Ball (ref. 27). The buckling mode shape indicates that the vertical edges of the cutout buckle locally.

The nonlinear analysis of the cylinder with cutouts was performed with the procedure NL_STATIC_1. Out-of-plane deflections w are shown in figure 9(b) as a function of the applied load for two points (denoted as "a" and "b" in fig. 9(a)). The elastic collapse load predicted with the Testbed is 2846 lb—nearly three times the linear bifurcation buckling load. As the out-of-plane deflections near the vertical edges of the cutouts develop, the compressive stresses are redistributed away from these regions and the load is carried by the remaining portions of the shell, as shown in figure 10.

Hartung and Ball (ref. 27) reported a collapse load of 2109 lb using a finite-difference version of the STAGS computer code. Later, Almroth and Brogan (ref. 28), in a convergence study using the finite-element version of STAGS, reported a "nearly" converged collapse load of 2750 lb. Additional research in shell element technology is needed in order to provide analysts with reliable structural analysis tools.

Table 2. Selected Processor Execution Times for Blade-Stiffened Panel

[628 nodes; 2910 degrees of freedom; average semi-bandwidth of 439]

Solution phase	Processor name	NAS CRAY-2, CPU sec	VAX 11/785, CPU sec
Mesh generation	ELD	3.2	6.7
	E	.3	11.4
	TOPO	1.6	15.8
Global stiffness matrix formation and factoring	ES	50.1	821.0
	K	1.4	26.8
	INV	8.3	347.0
	VEC	2.9	18.4
	SSOL	.9	28.8
Each iteration	SSOL	0.9	35.3
	VEC	2.1	19.4
	ES	13.5	230.1

Table 3. Performance of Direct Solvers in Processor BAND

[628 nodes; 2910 degrees of freedom; average semi-bandwidth of 439]

Method	NAS CRAY-2, CPU sec	Compute rate, MFLOPS
LINPACK	27.1	64.1
kji Choleski	27.4	63.4
kji Choleski*	17.7	98.2
kji Choleski†	12.7	136.9
kji profile	12.7	57.1
kji profile*	7.9	92.9
kji profile†	5.6	129.4

*Loop unrolling to level 4.
†Loop unrolling to level 4 and use of local memory.

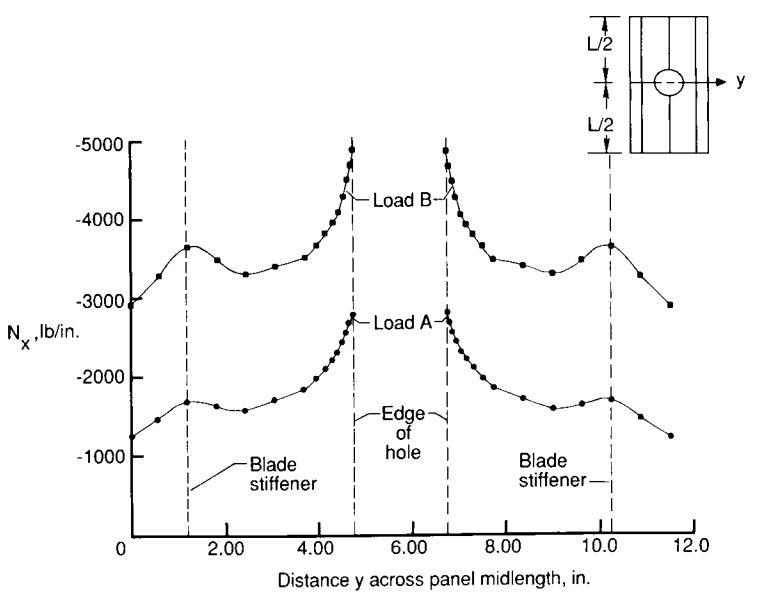


Figure 7. Longitudinal in-plane stress resultant N_x distributions at panel midlength.

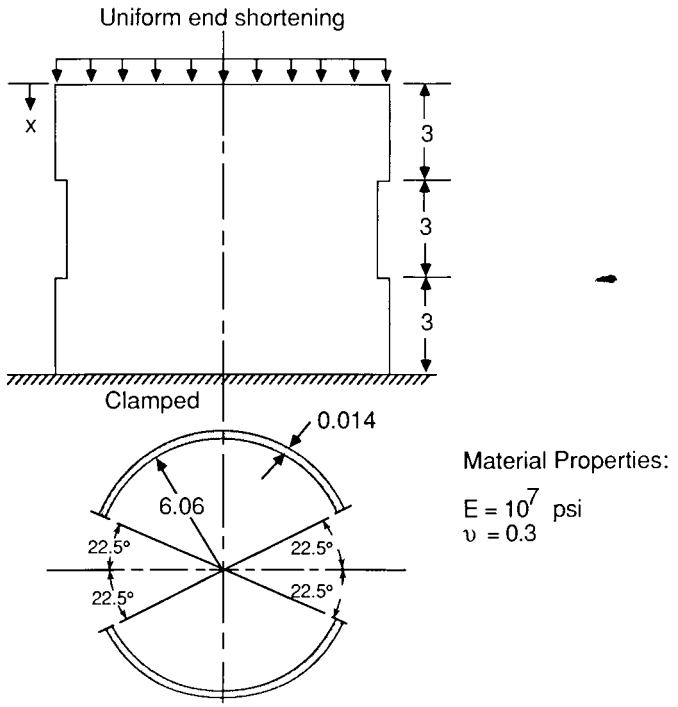
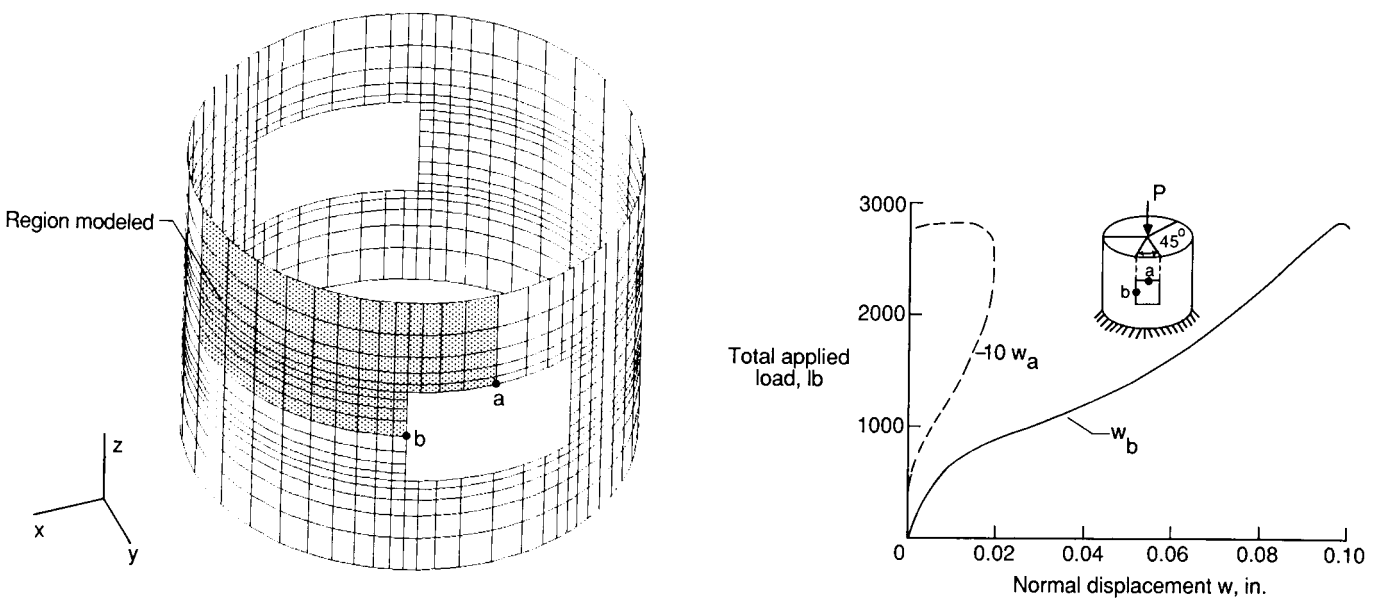


Figure 8. Circular cylinder with cutouts—geometry, properties, and loading. Dimensional quantities are in inches unless otherwise noted.



(a) Finite-element model.

(b) Out-of-plane deflections.

Figure 9. Nonlinear response of cylinder with cutouts.

ORIGINAL PAGE
COLOR PHOTOGRAPH

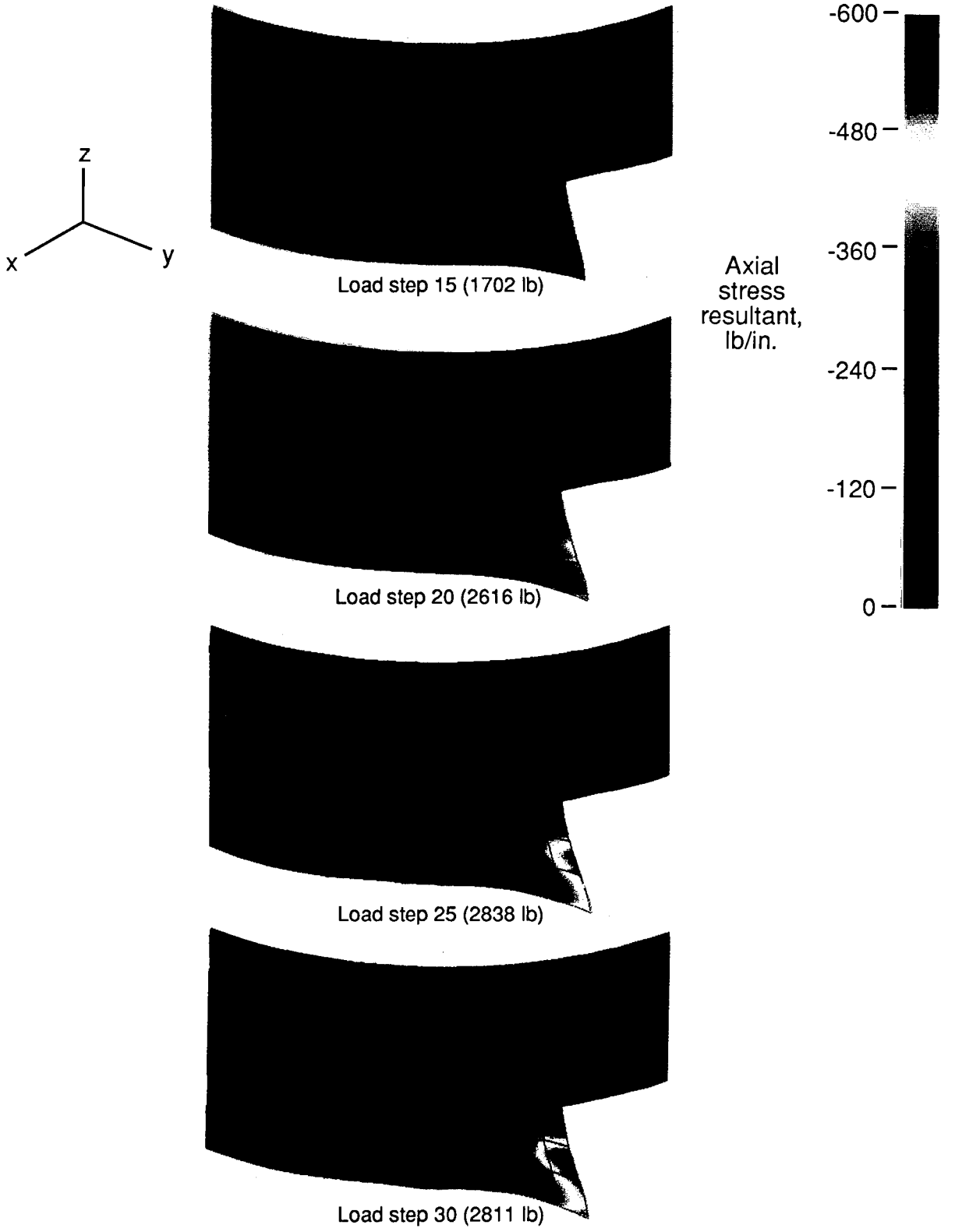


Figure 10. Deformed geometry for several load levels.

Computation times for the nonlinear analysis of the cylinder with cutouts are given in table 4 for selected Testbed processors. The ratio of the overall execution time on a VAX 11/785 computer system to the execution time on the NAS CRAY-2 computer system is 17.7 for the complete nonlinear analysis of the cylinder with cutouts. The global tangent stiffness matrix was reevaluated and factored 31 times and a total of 122 iterations were required to predict the nonlinear structural response. Most of the CPU time was spent in processors that perform computations on the element level, such as processor ES1. The ratio of the execution times for the evaluation of the elemental stiffness matrices is lower than the ratio of the overall execution time. These processors (e.g., ELD and ES1) have not been modified to exploit the features of vector computers. However, processor INV has been modified and performs 44.1 times faster on the NAS CRAY-2 computer system than on a VAX 11/785 computer system for this problem.

Impulsively Loaded Truncated Conical Shell

A number of important engineering problems are associated with the prediction of the response of a shell to high-energy, short-duration dynamic loads. Examples include reentry vehicles, space vehicles subjected to pyrotechnic separation loads, and vehicles subjected to blast or impulse environments (e.g., water impact). Sometimes these high-energy loads only generate a rapidly varying linear elastic stress state, but in other cases the loads may be sufficiently high or of sufficient duration that the structural response is nonlinear.

The linear elastic transient response of the truncated conical shell subjected to an impulse load (initial velocity) shown in figure 11 is selected as representative of these transient dynamic shell analysis problems. This problem also has been used as a benchmark problem by Hartung and Ball (ref. 27). The finite-element model is composed of 540 4-node quadrilateral elements, 589 nodes, and 2569 active degrees of freedom. The predicted transient response shown in figure 12 for the normal deflections at two points on the shell correlates well with the results presented in reference 26. Both points are located 6.5 in. from the clamped small-diameter edge, one at $\theta = 0°$ (point a) and one at $\theta = 180°$ (point b). The transient response was calculated for 1400 μsec using the Newmark method with a time step of 2 μsec. Oblique views of the deformed shape with exaggerated deflections from the transient analysis at various points in time T are shown in figure 13.

Space Shuttle Solid Rocket Booster

The basic elements of the Space Shuttle system are the orbiter, the external tank (ET), and the two reusable solid rocket boosters (SRB's). The SRB's provide the primary shuttle ascent boost for the first 2 minutes of flight with an assist from the three Space Shuttle main engines (SSME's) on the orbiter. A major subsystem of the SRB is the solid rocket motor (SRM), which consists of four lined, insulated rocket motor segments. These segments are connected with pinned tang-clevis joints. Each SRB is approximately 144 ft long and 12 ft in diameter.

The linear elastic static analysis of the SRB loaded by internal pressure was analyzed as representative of a large-scale structural analysis problem that is critical to NASA. The finite-element model shown in figure 14 involves 9205 nodes with 1273 2-node beam elements, 90 3-node triangular elements, and 9156 4-node quadrilateral elements. Although the finite-element model involves 54 870 degrees of freedom, it does not have the fidelity necessary to determine detailed stress distributions in particular SRB subsystems. In this global shell model, the field and factory joints are modeled with equivalent stiffness joints rather than detailed models of the joint. As such, local joint behavior cannot be obtained from this global model.

The linear stress analysis considered herein involves only the loading case of a uniform SRM internal pressure of 1000 psi. An oblique view of the deformed geometry with exaggerated deflections is shown in figure 15. The deflection pattern exhibits a "pressure pillowing" behavior in the vicinity of the joints. The influence of the partial (270°) SRB/ET attachment ring on the SRB shell response is shown in figure 16. An abrupt change in the deflection pattern near the ends of the ET attachment ring is exhibited.

Computation times for the SRB global shell analysis are given in table 5 for selected Testbed processors. The ratio of the overall execution time on a VAX 11/785 computer system to the execution time on the NAS CRAY-2 computer system is 35.6 for one linear stress analysis of the SRB global shell model. Most of the CPU time was spent in processor INV factoring the global stiffness matrix, a process which is 63.7 times faster on the NAS CRAY-2 computer system than on a VAX 11/785 computer system. However, several other processors (ELD, EKS, and TOPO) also used a sizeable amount of CPU time. All these processors need to be studied and improved for large-scale analysis problems.

The new Testbed equation solvers implemented in processors BAND and ITER have also been

Table 4. Selected Processor Execution Times for Cylinder With Cutouts

[449 nodes; 2012 degrees of freedom]

Solution phase	Processor name	NAS CRAY-2, CPU sec	VAX 11/785, CPU sec
Mesh generation	ELD	4.8	5.3
	E	.3	9.1
	TOPO	1.7	14.7
Global stiffness matrix formation and factoring	ES	33.4	549.7
	K	1.0	18.3
	INV	9.8	432.2
	VEC	3.9	15.5
	SSOL	.8	20.2
Each iteration	SSOL	0.8	20.6
	VEC	1.9	8.0
	ES	8.8	125.3

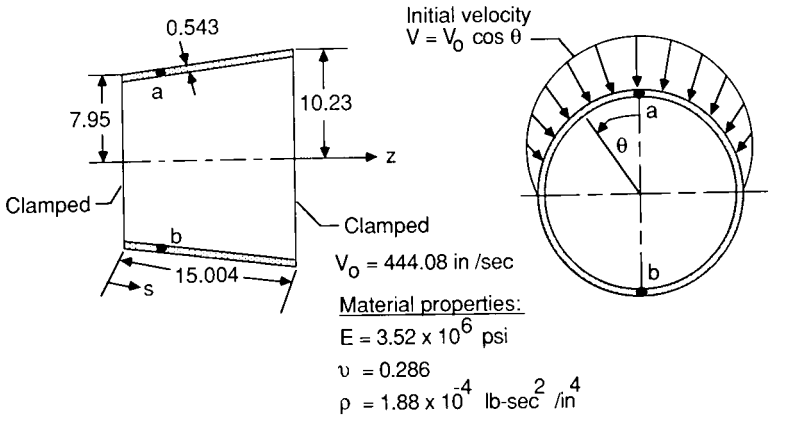


Figure 11. Truncated conical shell—geometry, properties, and loading. Dimensional quantities are in inches unless otherwise noted.

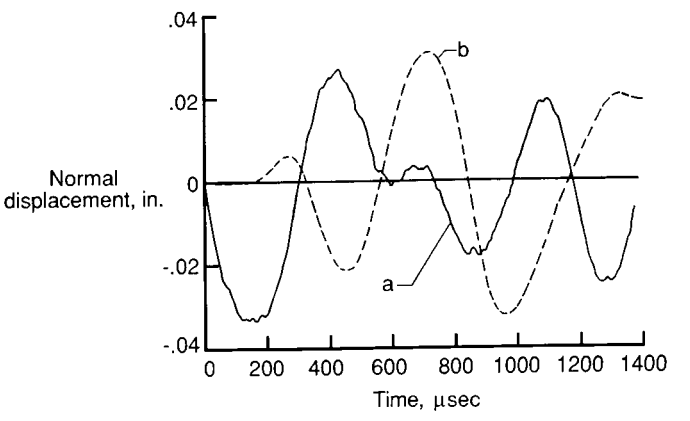


Figure 12. Normal deflections at points a and b on truncated conical shell.

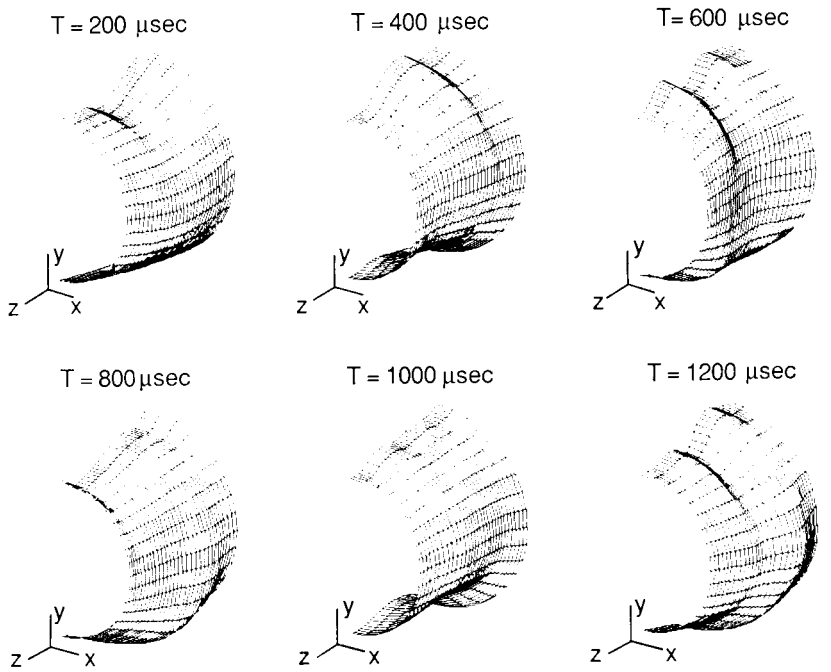


Figure 13. Deformed shapes for truncated conical shell during transient analysis.

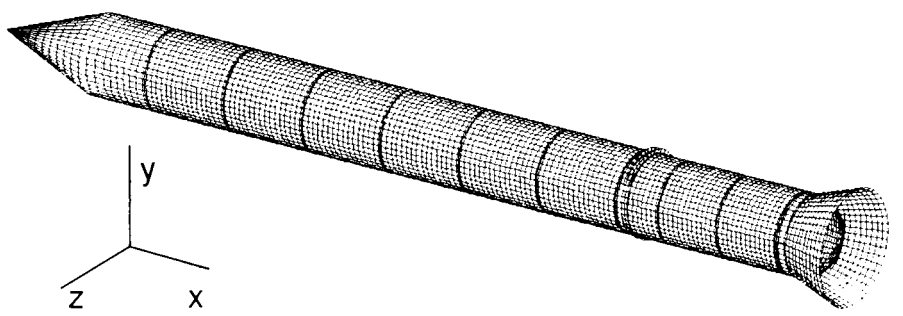


Figure 14. Finite-element model of SRB.

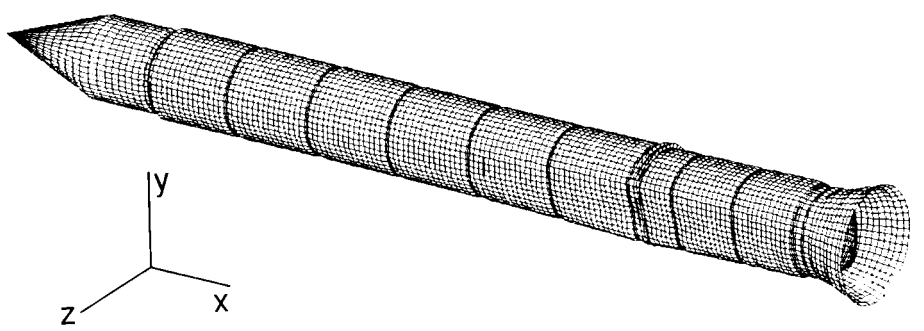


Figure 15. Deformed geometry plot of global SRB shell model.

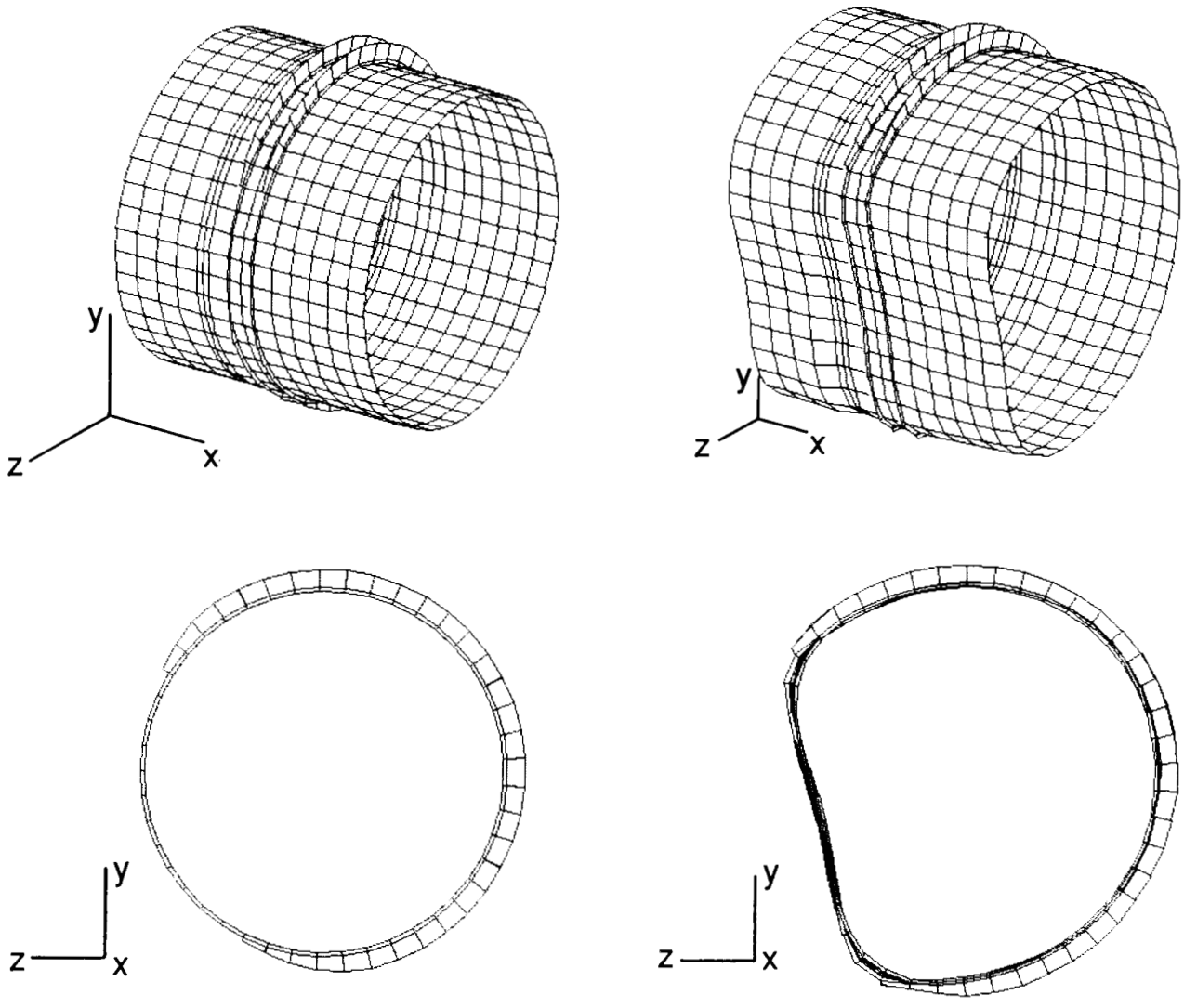


Figure 16. Close-up of undeformed and deformed geometries at SRB/ET interface.

Table 5. Selected Processor Execution Times for SRB Global Model

[9205 nodes; 54 870 degrees of freedom; average semi-bandwidth of 382]

Processor name	NAS CRAY-2, CPU sec	VAX 11/785, CPU sec
ELD	248.6	460.8
E	2.0	70.7
EKS	168.3	1625.0
TOPO	94.3	1678.4
K	46.9	472.3
INV	804.1	51185.1
SSOL	17.2	295.6

applied to this problem. Using the skyline method in processor BAND with loop unrolling to level 4 and exploiting local memory results in the solution time to factor and solve being 74.8 CPU sec on the NAS CRAY-2 computer system (a compute rate of 127.9 MFLOPS). Processor BAND factors the global stiffness matrix in less than one-tenth the CPU time required by processor INV on the NAS CRAY-2 computer system. With use of the incomplete Choleski conjugate gradient method with a sparse storage scheme, the solution is obtained after 562 iterations. The solution time is 455 sec which corresponds to a computation rate of 20 MFLOPS.

CSM Research Directions

The broad objective of the CSM activity is to develop advanced structural analysis and computational methods that exploit modern and emerging scientific computers, such as computers having vector and/or parallel processing capabilities. The evolving computational environment (both hardware and software) is providing new opportunities to structural analysts that enable them to study the structural behavior of complex nonlinear systems.

Command Language Enhancements

The CLIP enhancements include the implementation of a table-driven parser and lexical analyzer. The UNIX utilities LEX and YACC are being used to implement an easily extendable language. This language will be primarily the CLAMP language with modifications to remove context sensitive constructs from the language. As a side benefit, the resulting interpreter should be more efficient and maintainable and should provide the required extendability. This extendability will be tested by the addition of language directives to control processor-task allocation and synchronization at a high level through CLAMP directives. The resulting capability should provide a convenient research environment for the structural analyst to investigate parallelism without relying on computer-dependent coding.

Parallel Data Management

In addition to providing machine-independent control of parallel mechanisms, it will be necessary for the Testbed to provide machine-independent parallel data management. The data management components for engineering applications are reasonably well understood for sequential computers. However, multiple-instruction–multiple-data (MIMD) computers are a different matter. These new architecture computers, with the nondeterministic nature of processes running in parallel, create new requirements for maintaining data integrity across processors. For example, if data are written by processor one and will be needed by processor two, a mechanism must exist to ensure that processor one has written the data before processor two successfully returns from a read operation on that same data. The simple solution of blocking all subsequent I/O operations until processor one is complete must be avoided because that solution would eliminate the advantages offered by MIMD computers in the first place. In addition, if the structural analyst is to benefit from the capability of parallel I/O on the MIMD computers, the implementation details and internal workings of such a system must be hidden in a methods development environment such as the CSM Testbed.

Advanced Numerical Algorithms

To complement the transition to MIMD computers, numerical analysts are preparing a wealth of new algorithms designed to take advantage of the vector processing capability offered by many modern computers. In the past, the sparse nature of the matrices that dominate the structural analysis task has made vector processors of limited use. It is anticipated that work will continue on the development of numerical algorithms that will take full advantage of both the vector capabilities and the MIMD capabilities of future computer architectures. Such algorithms will be developed within the Testbed framework and will be evaluated on challenging structural analysis focus problems.

Structural Analysis Technology

The LaRC CSM activity for structural analysis technology is currently focused on methodology for predicting the nonlinear structural response of large-scale composite primary aircraft structures. Many of the structural analysis software systems available today can predict the nonlinear structural response of composite components. However, the lack of progressive failure analysis techniques in large-scale structural analysis systems limits the analyst in the design of composite aerospace structures. A capability to model and analyze damaged composite structures is needed in the aerospace community. In addition, designers need analysis tools that can be used to assess the sensitivity of variations in material properties or loads on selected response parameters for complex structural systems. Finally, error sensing and control strategies for finite-element solutions are needed in order to provide quantitative as well as qualitative information about the quality of the results from such calculations.

Summary

The computational structural mechanics (CSM) Testbed is a powerful methods development environment for developing structural analysis and computational methods. With enhancements and extensions for multiple-instruction–multiple-data (MIMD) computers, the Testbed should continue to be a useful research environment for the forseeable future. It is currently being used by researchers developing structural analysis methods and numerical algorithms and evaluating MIMD I/O strategies. The Testbed application environment provides the mechanism to allow researchers concentrating on different parts of the structural analysis problem to communicate on solutions to problems directly related to current NASA needs. The transfer of technology among researchers in structural engineering, computer science, and numerical analysis can now be accomplished more effectively than was previously possible.

An overview of the CSM activity at the NASA Langley Research Center has been presented. The CSM Testbed software system serves as a framework for structural analysis and computational methods research for high-performance computers. The CSM Testbed has been described and its use demonstrated through solution of selected structural analysis problems. Future directions for CSM research using the Testbed have been outlined. These future developments will take full advantage of both vector processors and parallel methods on the NAS CRAY-2 computer system and on anticipated supercomputers of the 1990's.

Appendix

CRAY-2 Implementation

The source code for the Testbed, both architecture and application modules, is maintained in text files with embedded preprocessing commands which allow selective conditional precompilation by machine-independent utility programs. The architecture code is made up of approximately 650 modules totaling about 83 000 lines of source code and include files. The application code is made up of approximately 1300 modules with about 95 000 lines in source code and include files. Distribution of the code in the UNIX environment is accomplished with the UNIX utility TAR to package the source code, makefiles, and scripts in a single file; this distribution file occupies approximately 8 megabytes of disk space. Installation of the Testbed on the NAS CRAY-2 computer was accomplished over a period of about 1 month in 1987 shortly after the computer was made available to LaRC users. The Testbed software system was the largest software system to be ported to the NAS CRAY-2 computer system, and consequently many problems which had not been experienced by other users had to be diagnosed and overcome.

Compilation Problems

Because the Testbed Fortran code uses character variables heavily, the CFT77 compiler had to be used for compilation. Most problems encountered with this compiler were related to its handling of character variables and formatting of output. These problems were encountered only at execution time. Most of these were resolved by inserting code blocks for the CRAY/UNICOS version into the master source files so that the modifications could be carried along into future versions of the code. The porting of the Testbed to the CRAY-2 computer system was accomplished using a very early version of CFT77 under UNICOS. Although several errors in the compiler were discovered, these errors could be worked around easily. These errors in the compiler have been corrected in subsequent releases of the CFT77 compiler.

Fortran-C Interface

One problem related to CFT77 character handling which had to be resolved twice (once under UNICOS 1.0 and 2.0 and again under UNICOS 3.0) was the difference in data structures for CFT77 character arguments and C compiler character string arguments. This problem arises where the Fortran code for the data management function calls low-level C language I/O functions. In this respect, the CFT77 compiler does not conform to the same standard as the Fortran compilers on other UNIX systems. To overcome the problem, in the C functions a C structure was defined to correspond to the CFT77 character argument; upon entry to the C function, a transformation was performed from the argument structure to a C character string. This structure was initially defined to be compatible with the compilers used under UNICOS 1.0 and 2.0. When version 3.0 of UNICOS was installed with a new CFT77 compiler, the CFT77 character variable structure was changed without documentation, so the C functions had to be modified to accommodate the new structure once the problem was identified.

Loader Problems

The initial installation procedures used the LD loader for linking the executable file. When the optimization options for the CFT77 compiler had been used in compilation, all subroutine argument addresses and some temporary variables were defined in

local memory by the compiler. The LD loader concatenates the local memory segments for all modules, so attempting to link all the application modules and libraries with the macroprocessor resulted in overflow of local memory ($40\,000_8$ words) and failure of the load. The LMSTAK utility to enable overlaying local memory segments was used, but the resulting program would not execute. In order to check the operation of the software before resolving the local memory overflow problem, all the code was recompiled without optimization, linked successfully, and tested.

Later, following a suggestion by the NAS analysts, the segmentation loader (SEGLDR) was used. This loader performed the local memory overlay correctly, so the optimized object code could be used. No execution errors were encountered as a result of using the optimizing compiler. Performance was improved by a factor of 3 in CPU usage with the optimized code for most of the demonstration problems executed. Installation of a new CFT77 compiler with options to enable the user to control the allocation of local memory has since eliminated the requirement to use SEGLDR for the Testbed to overlay local memory. However, vectorization is not used efficiently in this version of the code because of short vector lengths actually used (≤ 6 in a critical area). Much greater improvements should be gained by tailoring the matrix operations in the code to take advantage of vectorization.

Optimization

In order to identify the most promising areas for performance improvement, two utilities were used. First, the UNICOS utility FLOW was used, after recompilation of the code with the CFT77 flowtrace (-ef) option. The resulting executable file was executed with several demonstration problems performing different types of analysis functions. The FLOW utility analyzed the output files and identified the modules which were using most of the CPU time for the executions. A calling tree diagram was also obtained in the FLOW output, which was helpful in analyzing the execution path of the program.

After identifying the biggest CPU users, the Fortran source code for those modules was sent to an IRIS workstation on which the FORGE software was installed. FORGE is a software system for optimizing Fortran programs for CRAY computer systems. FORGE attempts to exploit many of the intricate details of CRAY hardware and software and to restructure the Fortran programs for faster execution on CRAY computer systems. FORGE was used to insert timing function calls into the modules, which were then sent back to the CRAY computer, compiled, and linked into the executable file. The demonstration problems were executed again and very detailed analyses of the execution of the modules of interest were obtained. These analyses led to replacement of some code with UNICOS library function calls and some other minor revisions. This work resulted in an improvement of about 12 percent in the performance of the affected analyses.

NASA Langley Research Center
Hampton, VA 23665-5225
February 3, 1989

References

1. Blankenship, Charles P.; and Hayduk, Robert J.: Potential Supercomputer Needs for Structural Analysis. *Proceedings Supercomputing '87—Industrial Supercomputer Applications and Computations, Volume II,* International Supercomputing Inst., Inc., 1987, pp. 180–202.
2. Knight, Norman F., Jr.; and Stroud, W. Jefferson: *Computational Structural Mechanics: A New Activity at the NASA Langley Research Center.* NASA TM-87612, 1985.
3. Lotts, C. G.; Greene, W. H.; McCleary, S. L.; Knight, N. F., Jr.; Paulson, S. S.; and Gillian, R. E.: *Introduction to the Computational Structural Mechanics Testbed.* NASA TM-89096, 1987.
4. Bailey, F. R.: NAS—Current Status and Future Plans. *Supercomputing in Aerospace,* NASA CP-2454, 1987, pp. 13–21.
5. McLean, Donald M., ed.: *MSC/NASTRAN Programmer's Manual—MSC/NASTRAN Version 63.* MSR-50, MacNeal-Schwendler Corp., Oct. 1983.
6. Felippa, Carlos A.: Architecture of a Distributed Analysis Network for Computational Mechanics. *Comput. & Struct.,* vol. 13, no. 1–3, June 1981, pp. 405–413.
7. Felippa, C. A.; and Stanley, G. M.: NICE: A Utility Architecture for Computational Mechanics. *Finite Element Methods for Nonlinear Problems,* P. G. Bergan, K. L. Bathe, and W. Wunderlich, eds., Springer-Verlag, c.1986, pp. 447–463.
8. Whetstone, W. D.: *SPAR Structural Analysis System Reference Manual—System Level 13A. Volume I: Program Execution.* NASA CR-158970-1, 1978.
9. Gillian, Ronnie E.; and Lotts, Christine G.: *The CSM Testbed Software System: A Development Environment for Structural Analysis Methods on the NAS CRAY-2.* NASA TM-100642, 1988.
10. Felippa, Carlos A.: *The Computational Structural Mechanics Testbed Architecture. Volume I—The Language.* NASA CR-178384, 1988.
11. Wright, Mary A.; Regelbrugge, Marc E.; and Felippa, Carlos A.: *The Computational Structural Mechanics Testbed Architecture. Volume IV—The Global-Database Manager GAL-DBM.* NASA CR-178387, 1989.
12. Hurst, Patricia W.; and Pratt, Terrence W.: Executive Control Systems in the Engineering Design

Environment. *Collection of Technical Papers, Part 1—AIAA/ASME/ASCE/AHS 26th Structures, Structural Dynamics and Materials Conference,* Apr. 1985, pp. 96–105. (Available as AIAA-85-0619.)

13. Felippa, Carlos A.: Fortran-77 Simulation of Word-Addressable Files. *Adv. Eng. Softw.,* vol. 4, no. 4, Oct. 1982, pp. 156–162.
14. Ortega, James M.: *Introduction to Parallel and Vector Solution of Linear Systems.* Plenum Press, c.1988.
15. Poole, Eugene L.; and Overman, Andrea L.: *The Solution of Linear Systems of Equations With a Structural Analysis Code on the NAS CRAY-2.* NASA CR-4159, 1988.
16. Dongarra, J. J.; Bunch, J. R.; Moler, C. B.; and Stewart, G. W.: *LINPACK Users' Guide.* SIAM, 1979.
17. Rankin, C. C.; and Brogan, F. A.: An Element Independent Corotational Procedure for the Treatment of Large Rotations. *J. Pressure Vessel Technol.,* vol. 108, no. 2, May 1986, pp. 165–174.
18. Stanley, Gary Mitchel: *Continuum-Based Shell Elements.* Ph.D. Diss., Stanford Univ., 1985.
19. Park, K. C.; and Stanley, G. M.: A Curved C^0 Shell Element Based on Assumed Natural-Coordinate Strains. *J. Appl. Mech.,* vol. 53, no. 2, June 1986, pp. 278–290.
20. Kang, David S.; and Pian, Theodore H. H.: A Versatile and Low Order Hybrid Stress Element for General Shell Geometry. *A Collection of Technical Papers, Part 1—AIAA/ASME/ASCE/AHS 28th Structures, Structural Dynamics and Materials Conference,* Apr. 1987, pp. 633–641. (Available as AIAA-87-0840.)
21. Rankin, C. C.; Stehlin, P.; and Brogan, F. A.: *Enhancements to the STAGS Computer Code.* NASA CR-4000, 1986.
22. Jones, Robert M.: *Mechanics of Composite Materials.* McGraw-Hill Book Co., c.1975.
23. Whitney, James M.: *Structural Analysis of Laminated Anisotropic Plates.* Technomic Publ. Co., Inc., c.1987.
24. Riks, Eduard: *On the Numerical Solution of Snapping Problems in the Theory of Elastic Stability.* AFOSR 70-2258TR, U.S. Air Force, Aug. 1970.
25. Crisfield, M. A.: A Fast Incremental/Iterative Solution Procedure That Handles "Snap-Through." *Comput. & Struct.,* vol. 13, no. 1–3, June 1981, pp. 55–62.
26. Knight, Norman F., Jr.; McCleary, Susan L.; Macy, Steven C.; and Aminpour, Mohammad A.: *Large-Scale Structural Analysis: The Structural Analyst, the CSM Testbed, and the NAS System.* NASA TM-100643, 1989.
27. Hartung, Richard F.; and Ball, Robert E.: *A Comparison of Several Computer Solutions to Three Structural Shell Analysis Problems.* AFFDL-TR-73-15, U.S. Air Force, Apr. 1973. (Available from DTIC as AD 762 946.)
28. Almroth, B. O.; and Brogan, F. A.: Computational Efficiency of Shell Elements. *Nonlinear Finite Element Analysis of Plates and Shells,* Thomas J. R. Hughes, A. Pifko, and A. Jay, eds., AMD-Vol. 48, American Soc. of Mechanical Engineers, 1981, pp. 147–165.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TM-4072	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle CSM Testbed Development and Large-Scale Structural Applications		5. Report Date April 1989
		6. Performing Organization Code
7. Author(s) N. F. Knight, Jr., R. E. Gillian, S. L. McCleary, C. G. Lotts, E. L. Poole, A. L. Overman, and S. C. Macy		8. Performing Organization Report No. L-16499
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665-5225		10. Work Unit No. 505-63-01-10
		11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001		13. Type of Report and Period Covered Technical Memorandum
		14. Sponsoring Agency Code

15. Supplementary Notes

N. F. Knight, Jr., and R. E. Gillian: Langley Research Center, Hampton, Virginia.
S. L. McCleary, C. G. Lotts, and S. C. Macy: PRC Kentron, Inc., Aerospace Technologies Division, Hampton, Virginia.
E. L. Poole and A. L. Overman: Awesome Computing, Inc., Charlottesville, Virginia.

16. Abstract

A research activity entitled computational structural mechanics (CSM) at the NASA Langley Research Center is described. This activity involves developing advanced structural analysis and computational methods that exploit high-performance computers. New methods are developed in the framework of the CSM Testbed software system and applied to representative complex structural analysis problems from the aerospace industry. An overview of the CSM Testbed methods development environment is presented and some new numerical methods developed on a CRAY-2 computer system are described. Selected application studies performed on the NAS CRAY-2 are also summarized.

17. Key Words (Suggested by Authors(s)) Computational structural mechanics Structural analysis	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 22	22. Price A03