NASA Contractor Report 198502

Concurrent Probabilistic Simulation of High Temperature Composite Structural Response

Frank Abdi
Alpha STAR Research Corporation
Los Angeles, California

July 1996

Prepared for
Lewis Research Center
Under Contract NAS3–26997

National Aeronautics and
Space Administration

Acknowledgement

This report was prepared under NASA Small Business Innovative Research Program (SBIR) Phase II (Contract No. NAS3-26997) funding. Dr. F. Abdi is the Alpha STAR program manager . Other member of the team include Mr. B. Littlefield, Mr. B Golod, Dr. J. Yang, Dr. J. Hadian, Mr. Y. Mirvis, and Mr. B. Lajevardi.

We also would like to acknowledge special contribution by NASA program manager Dr. C. C. Chamis, and Dr. P. L. N. Murthy for useful discussion and suggestions for improvement.

We acknowledge the invaluable contribution of Mr. D. Collins and Dr. B. Nour-Omid. Our special thanks to Mr. R. Lorenz whose help was invaluable in putting the report together.

Contents

Contents

Contents

Illustrations

Illustrations

Illustrations

Illustrations

Figure.....Page

Illustrations

Figure ..... **Page**

Tables

Tables

1.0 Introduction

NASA has invested billions of dollars in research and development of advanced materials in support of U.S. Space programs such as Apollo, Shuttle, National Aerospace Plane (NASP) and High Speed Civil Transport (HSCT). The advanced material technology resulting from this investment, in accordance with new government policy, is now being made available for commercial use, in such applications as turbines engines and the automotive industry. This is particularly true for advanced composite materials.

Cost constraints on design and fabrication prototypes using advanced composite materials are significantly intensified relative to those new monolithic materials. This can seriously impact selection of advanced composite materials for commercial use, especially in globally competitive markets.

Traditional methods rely on single value safety factors derived from data and experience, and do not address uncertainties. Probabilistic methods are needed to quantify uncertainties, sensitivities, reliability and risk, in order to minimize development time, and significantly lower cost. The use of the probabilistic method for analysis and design of systems to account for uncertainties in the characteristics of composite materials will substantially speed their acceptance as usable structural materials.

Promising advancements in the fields of numerical methods and super-computing have provided the means to implement development of analytical tools needed by industry to utilize MMC materials in their designs. One of these advancements is the application of non-deterministic analytical methods to evaluate numerous design parameters, such as strength, property degradation, and structural reliability, needed to design and manufacture safe and cost-effective MMC structures .

Another advancement is the technological headway made in computer hardware as a result of the heated battle to produce the first teraflop computer. This battle has emphasized parallel architectures and associated analytical methodologies. The speed and cost of the resulting parallel computers makes affordable the development and application of many computation intensive structural analysis methods previously rejected as too costly and time consuming.

Although advanced computer programs such as the High Temperature Composite Analyzer (HITCAN) have progressed to the point of addressing structural response with respect to specific material systems and can analyze structural response to determine loading environments, these lack the probabilistic capability needed to analyze complex composite structures. Accordingly, attention has been devoted to the implementation of finite element techniques for probabilistic solution methods, and new innovative approaches for probabilistic modeling have been developed.

High performance computers have been developed to provided a cost effective environment for rapidly executing computationally intense engineering software. Numerous innovative techniques have been developed which place large ensembles of high speed processors in parallel architectures producing massively parallel computer hardware. By the end of the decade, these developments promise to deliver a sustained performance of 1 trillion floating point operations per second (1 TFLOPS). However, the advantages of parallel processing require considerable software modification to fully realize the added speed of new multi-processor hardware.

The awareness of the need for non-deterministic methods is recognized by NASA's funding of research programs such as the Probabilistic Structural Analysis Methods for Select Space Propulsion System Components (PSAM), and specialized research codes such as NESSUS (Numerical Evaluation of

Stochastic Structures Under Stress), LDEXPT (a probabilistic load expert system software), PROMISS (Probabilistic Material Strength Simulator) and Integrated probabilistic Assessment of Composite Structures (IPACS). These capabilities have demonstrated that the development costs of composite structures can be reduced significantly from the formerly prohibitive levels.

In particular, what is needed now is a restructuring of the NASA funded programs to allow appropriate communication among processors and incorporation of decoupling techniques to enable solutions in a parallel environment. The integration of the advances in probabilistic structural analysis software, composite materials and parallel computing is a multifaceted task. The structuring of such an integrated capability will be facilitated by many aspects of parallelism inherent in composite programs like HITCAN.

The following report provides an insight to the future direction of industry with respect to developing an appropriate integrated package and establishing markets to support development of the proposed capability. In particular, this report details how Alpha Star (ASC), an engineering analysis software company, has addressed these issues by developing an advanced analysis package called *GENOA*. ASC has worked in conjunction with NASA to develop the probabilistic methods technology and algorithms, and compile a massive database of material properties with associated probabilities factors for metal matrix composites, polymer matrix composites, and ceramic matrix composites, to accomplish reliable simulation, testing and optimization of advanced materials.

1.1 BACKGROUND

Presently available sequential processor computers are not adequately equipped to perform computational tasks wherein large numbers of simulations must be made to evaluate responses resulting from perturbations of numerous uncertainties. Such evaluation involves a tremendous number of large arrays that must be temporarily stored for use in subsequent iterations. Achievement of this capability will require software development utilizing next-generation parallel processing hardware that is projected to have computational speed required to perform large computational simulations.

Developing codes for execution on parallel computers requires understanding of the type of multi-processor architecture. Also, consideration must be given to whether or not the parallel computer architecture is scaleable. The performance of a scaleable computer does not degrade as the number of processors is increased thereby ensuring that the gain in using a parallel computer outweighs any cost associated with transferring a sequentially developed code to a parallel environment. This is a primary reason for favoring selection of parallel computers with MIMD (multiple instruction, multiple data) hypercube architecture such as presently used by Ncube, IBM-SP2 and recently developed hybrid hypernode computers such as the CONVEX Scaleable Parallel Processor Exemplar-10000.

As far as the software control structure is concerned, the advantages of developing a code such as Alpha Star's GENOA on a local memory machine, such as the hypercube, are two-fold. A local memory machine may be thought of as a special case of a shared memory machine where each processor is restricted to access only the part of the memory assigned to it. Therefore, software development on local memory machines can easily be ported to shared memory machines. The converse is not the case. Moreover, the development of software on local memory machines alleviates many of the problems (including loss of efficiency) associated with memory contention on shared memory machines.

Further evidence supporting selection of the MIMD hypercube architecture is presented in research performed by John Gustafson and co-workers [1.1]. They noted that it can be much easier to achieve a high degree of parallelism than might be entered from *Amdahl's Law* [1.2]. Their results showed that the inherent serial component ,s, of scientific programs can be made quite small for practical problems and that when problem size is scaled in proportion to the number of processors s can effectively decrease removing the (Amdahl's Law) barrier to speedup as the number of processors is increased. This is evidenced by the following.

If P is the number of processors, s is the fraction of time spent (by a serial processor) on serial parts of the program, and p is the fraction of time spent (by a serial processor) on parts of the program that can be done in parallel, then Amdahl's law gives:

$$\text{Speedup} = \frac{(s+p)}{\left(s+\frac{p}{P}\right)} = \frac{1}{\left(s+\frac{p}{P}\right)} \qquad (1)$$

For a machine like the NCubed/10, with P = 1024, speedup is a steep function of s near s = 0 (as shown in Figure 1-1).

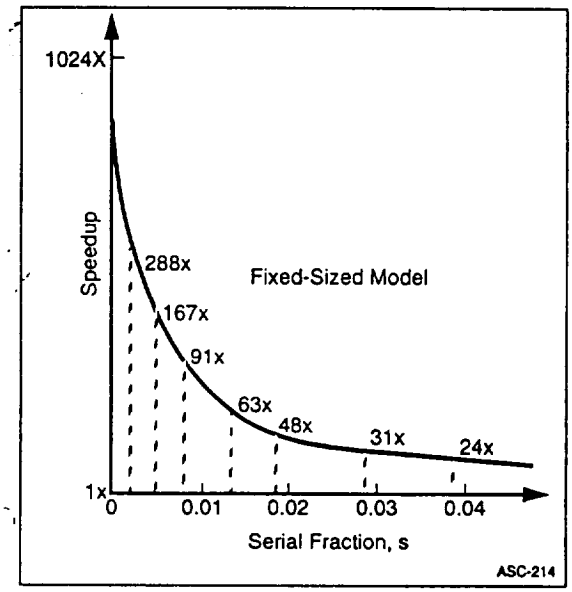


Figure 1-1. Speedup Given by Amdahl's Law.

Amdahl's law is based on the implicit assumption that p is independent of P and that a problem of fixed size is simply run on various numbers of processors. In practice, this is not the case. A scientific computing problem can be scaled with the number of available processors (as done in GENOA). As a first approximation, Gustafson found that the parallel part of a program scales with the problem size. Computing time associated with program loading, serial bottlenecks, and I/O that make up the s component of the application do not scale with problem size. When the number of processors is doubled, it doubles the number of spatial variables in a physical simulation. Therefore, the amount of work that can be done in parallel varies linearly with the number of processors.

Gustafson demonstrated this by considering the inverse of Amdahl's law: rather than asking how fast a given serial program can be run on a parallel processor, he asked how long a given parallel program would have taken to run on a serial processor. This reasoning gives an alternative to Amdahl's law:

$$\text{Scaled Speedup} = \frac{(s' + p'P)}{(s' + p')} = P + (1 - P)s' \qquad (2)$$

where s' and p' represent the serial and parallel time spent on the parallel system, s' + p' = 1. Therefore, a single processor requires s' + p'P to perform the task. This relationship contrasts that of (1) and is shown to be a line of moderate slope in Figure 1-2.

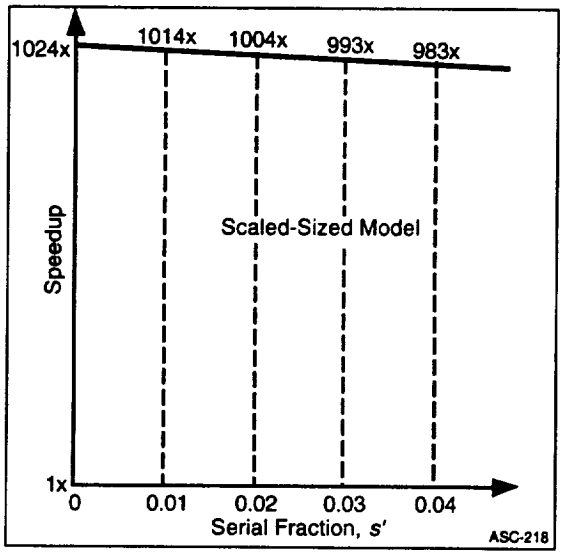


Figure 1-2. Speedup Given by Problem Scaling

1.1.1 Hardware for Parallel Processing

ENIAC [1.3], the very first general-purpose computer, employed parallel processing in the form of multiple arithmetic units. Today sequential computers employ parallel processing in the form of coprocessors that off-load from the central processing unit (CPU) tasks such as servicing input and output (I/O) devices. Within the CPU itself, instruction execution is partitioned into multiple stages allowing concurrent use of multiple functional units [1. 5]. These stages constitute a pipeline each stage of which can be occupied by a different instruction.

Pipelining has been universally utilized since being introduced in 1961 in the IBM STRETCH [1.6]. The pipeline stages of the Stanford MIPS processor consist of: instruction fetch, register fetch, execution, memory access, and register write-back [1.7]. A glance at the MIPS pipeline shows that most of the cycles expended by an instruction are used to decode it, fetch its operands, and store the results. These are the overheads required to perform a specific operation on a general-purpose computer. If the same instruction is to be repeated on a set of N contiguous values, this overhead can be amortized over the N operations by providing a vector instruction that specifies multiple operands. Vector processing yields a tremendous improvement in performance and as a result has been incorporated in all supercomputer starting with Control Data Corporation's STAR-100 [1.8].

The preceding techniques are features used to maximize the performance by improving throughput sequential machines used for scientific and engineering applications. However, since no sequential machine can run faster than the fastest available logic (the cycle time of the Cray 2 is 4.lns), achievement of higher performance, requires the use of multiprocessor machines. In the last twenty years, there have been an increasing number of multiprocessor machines available. At the present time, almost all vendors of minicomputers, mainframes, and supercomputers offer four processor versions of their systems [1.9, 1.10].

The first issue in design of a multiprocessor machine is the distribution of control. In von Neumann computers, execution is controlled by sequencing through a single instruction stream (SI). Single instruction stream, single data stream (SIMD) multiprocessors retain this configuration. Concurrency is exploited by providing multiple data streams and, of course, multiple functional units.

Early multiprocessors such as SOLOMAN [1.11] and ILLIAC-IV [1.12] tended to be SIMD machines as the need for only one CPU reduced hardware costs. These machines offer tremendous speedups when applied to, nested FORTRAN DO loops that do not contain insurmountable, inter-loop data dependencies [1.13]. As integrated circuit densities have increased, so have the number of processors incorporated in SIMD machines. ILLIAC-IV (introduced in 1972) had 64 processors, STARAN (also 1972) had 256 [1.14], the MPP (1983) has 16,384 [1.15], and the Connection Machine (CM, 1986) has 65536 [1.16]. Zakharov [1.17] provides an excellent survey of early parallel processors and more recent SIMD machines.

Often, potentially concurrent portions of applications programs are not incorporated in nested loops and thus cannot utilize SIMD machines. The scalar control processor becomes a bottleneck and disappointing speedups are attained [1.18]. This result, together with the fact that VLSI makes multiple control streams affordable, has led to the introduction of numerous multiple instruction stream, multiple data stream (MIMD) machines [1.19]. These machines offer greater flexibility in that different processors can execute unrelated code segments concurrently.

The biggest issue in MIMD architectures is the distribution of the memory. Karp [1.20] classifies systems as being shared memory, hybrids or, message passing,. Shared memory multiprocessors provide each processor with uniform access to a global address space. Access time to any datum is independent of the processor making the request and processes can migrate among the processors without any degradation in performance. Hybrids provide a shared address space, but physically distribute the memory banks. As a result, access to local memory banks is faster than access to remote ones. Message passing machines provide each processor with a unique memory space. Figure 1-3 depicts each of these organizations[1.21].

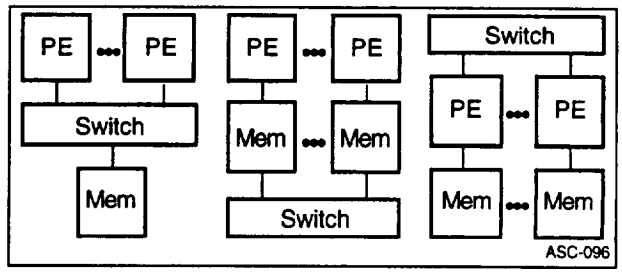


Figure 1-3. Three Multiprocessor Organizations: Shared Memory (a), Hybrid (b), and Message Passing (c)

The cross-bar interconnection network is depicted in Figure 1-4a. It is ideal in that, short of allowing multiple processors (PE) simultaneous access to the same memory bank(MEM), all possible permutations of memory requests can be supported. One of the earliest shared-memory multiprocessors, the C.mmp [1.22] employed a cross-bar switch.

Currently, the Alliant FX/8 [1.23] uses a time multiplexed 4 X 4 cross-bar to interconnect eight vector processors to the four banks of a shared cache. The disadvantage of the cross-bar is that it scales as P 2, where P is the number of processors. Therefore, they are generally considered too expensive for large scale multiprocessors.

A less costly alternative is the omega or butterfly switch, shown in Figure 1-4b. A P by P network is composed of logr,P layers of P /r switches. Each of these switches is an r by r cross-bar. All requests for access to the memory can be resolved as long as there are no conflicts accessing a single bank or a communication link in the network. Because of the latency, or the time required to transit the switch, these machines are usually organized as hybrids. Examples of machines that use these, or similar networks

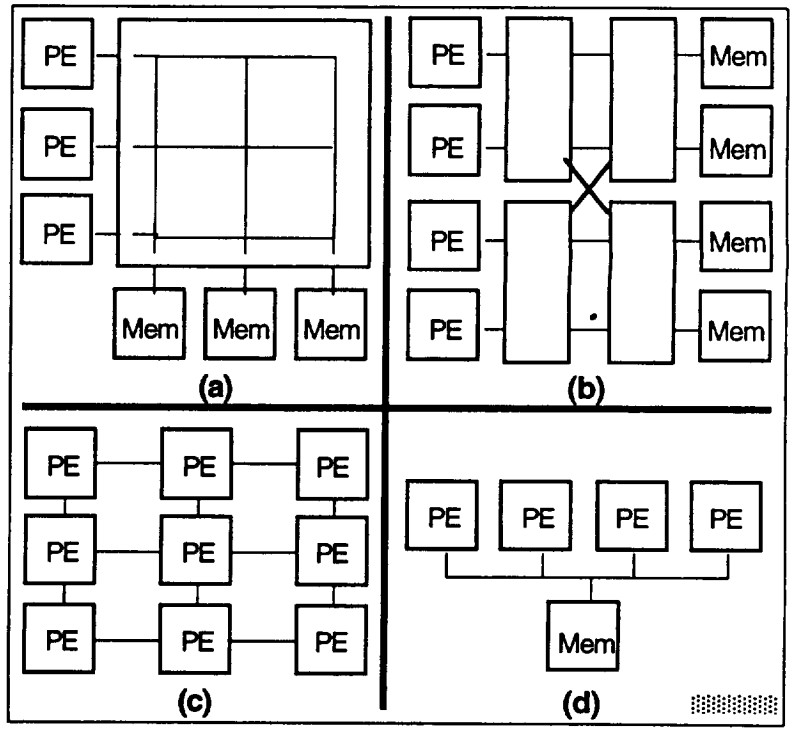


Figure 1-4. Four Interconnection Networks: the Crossbar (a), the Butterfly (b), the 2-D Mesh (c), and the Shared Bus (d)

abound. The Denelcor HEP was a shared-memory MIMD machine [1.24]. Hybrid MIMD machines include the BBN Butterfly [1.25], Burroughs' proposed Flow Model Processor (FMP) [1.18], the NYU Ultracomputer [1.26], and IBM's Research Parallel Processor Prototype (RP3) [1.27]. These machines are designed to have as many as 512 processors.

While omega networks are less expensive than cross-bars, they still scale as p logr P. In order to further reduce the cost of the interconnection network, D-dimensional meshes are used. Mesh-connected multiprocessors are almost invariably distributed memory, message-passing machines. A 2-D mesh or ring of processors, such as the Waterloop/64 represents the least expensive extreme. Most research projects have focused upon 2-D and 3-D meshes which correspond to the 2- or 3-dimensional Cartesian coordinates of a physical problem. The PACS [1.28] is an example of a MIMD, 2-D mesh. Finally, there are the hypercubes in which each of 2^D processors is placed at the corners of a D-dimensional binary cube. The Cosmic Cube [1.29] was the first hypercube implemented. Their are now four commercial examples of MIMD hypercubes [1.28].

The simplest interconnection network is the shared bus. The single bus and main memory depicted in Figure 1-4d can support as many as thirty microprocessors if each has a local cache to buffer its memory references [1.29]. The only problem is preventing the caches from containing different values of a shared datum. This cache coherency issue is overcome by having each cache monitor the memory bus for writes to locations it stores. The new values can either be input to the cache or the appropriate location in the cache invalidated. MIMD shared bus research projects include Cm* [1.32], ZMOB [1.33], the Multi-Maren machine [1.34], and Pringle [1.35]. Examples of commercial machines are provided by Encore, Elxsi, and Sequent [1.19].

1.1.2 Parallel Sparse Solvers

The vast majority of programs written to perform numerical simulations were designed to execute on uniprocessors. These programs must be recoded to incorporate new, concurrent algorithms for use in parallel processing machines.

Fortunately, the throughput of these codes tends to be dominated by the assembly and solution of large sparse systems of equations. Therefore, parallel sparse solvers and their associated assembly routines are the key to porting numerical simulators to multiprocessors. The need for efficient parallel sparse solvers has long been recognized. An excellent review of the research performed through 1984 is provided by Ortega [1.36].

The repeated assembly and solution of large systems of equations is often the computational bottleneck of numerical simulation program. The storage and computational requirements of these problems are outpacing the growth of capabilities of traditional sequential machines. The situation can be rectified by the use of multiprocessor machines. provided that concurrency is identified in the solution algorithms. This requires consideration of how concurrency can be exploited when a system of equations is many times larger that the number of processors.

There are two straightforward ways to distribute assembly of the Jacobian across a multiprocessor. The first is to allocate a subset of the vertices of the simulation grid to each processor. Processors then assemble the rows, or columns, corresponding to their own block of vertices. The second is to allocate faces of the grid to each processor and allow them to separately compute the contribution made by each face to the equations at its vertices. If one assumes that addition is associative (which is not correct for finite precision arithmetic) then the resulting matrix will be the same, regardless of how its assembly is distributed. In either case, the processors will have to share information regarding the values at vertices adjacent to their respective subsets. One way to minimize this sharing of data is to allocate contiguous blocks of vertices or faces to each processor such that the perimeter of each block is minimized.

Not only must the assembly of the Jacobian be distributed, so should the underlying physical problem and the matrix itself. If either is stored in a global memory, processors can be blocked while reading or writing the values. Writing, sometimes called stamping, the coefficients of the matrix is the biggest problem. Processors must not only overcome contention for the memory, they must synchronize in order to prevent one processor from overwriting the contribution of another. This problem can be circumvented by the use synchronization primitives (similar to those developed for uniprocessors [1.37]) at the expense of a possible run-time delay incurred in the execution of such primitives [1.38].

Parallel efficiency can be constrained by a poor distribution of the work to the processors when using Finite difference (FD) or Finite Element (FE) techniques. The work required to compute each coefficient of the Jacobian can be determined from the structure of the grid and the discretization scheme and therefore, a static allocation of the grid to the processors is usually performed. Unfortunately, it is often difficult to balance the load exactly. Allocating an even number of vortices or faces of the grid to every processor may conflict with the requirements that each processor's local block of vortices retain spatial locality in the grid and that the premiter of the block be minimized. Furthermore, an even distribution of the grid, one that would maximize the efficiency of the assembly phase, can be suboptimal for the subsequent solution phase.

An optimal way of distribution of tasks, while minimizing message passing among processors can be effectively performed if each processor is responsible for the assembly and solving of the matrices assigned by a dynamic load balancing system such as developed for GENOA. This is accomplished by an alternate distribution of the sparse matrix using a frontal or out-of-core matrix factorization technique [1.39].

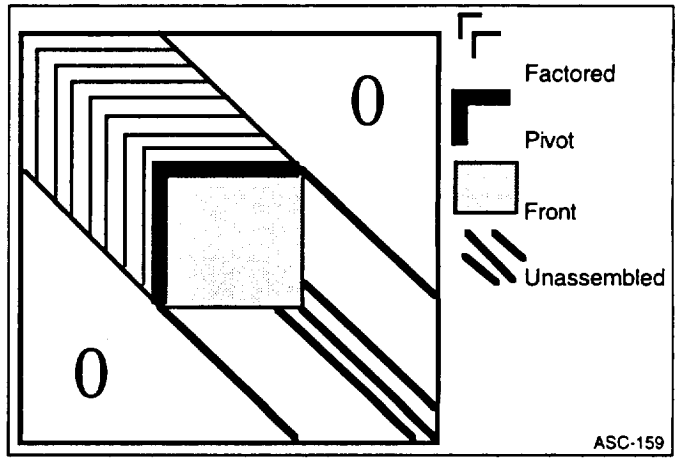


Figure 1-5. A Representative of the Frontal Decomposition of a Banded Matrix.

The frontal method [1.40] was introduced as a means of solving finite element systems that are too large to reside in the main memory of a computer. Figure 1-5 contains a representation of the factorization of a band matrix by the frontal method. The submatrix representing a physically adjacent set of elements is assembled and factored. Updates generated for locations in the matrix beyond those already assembled are maintained in a separate submatrix called the front. After elements are decomposed, the factors are placed in secondary storage. New elements are then assembled with updates from the values stored in the front. The procedure continues until the entire matrix has been factored.

The frontal method in [1.40] treats the symmetric positive-definite case. An assembly order is chosen a priori, and each variable is eliminated as soon as it is fully summed. Hood [1.41] extended this idea to unsymmetric matrices with a symmetry sparsity pattern. The frontal method has proved to be very effective for solving large-scale problems on small computers [1.42]. Duff [1.43, 1.44] has provided an implementation that treats unsymmetric patterns. Pivots are chosen on numerical grounds from the submatrix of fully summed rows and columns. Some pivots are now off the main diagonal, and all the fully summed variables are not necessarily eliminated at once. In both cases all operations are performed in a frontal matrix that corresponds to the submatrix of rows and columns belonging to variables that have appeared in one or more of the fronts so far assembled, but have not yet been eliminated. After each elimination the pivot row is stored ready for use in back substitution.

Multifrontal solution algorithm can be implemented on various computers with different architectures. Calalo et. al. [1.46] implemented a parallel version of the algorithm on NCUBE2 hypercube. They showed additional speed up by presenting several numerical examples.

Duff and Reid[1.40] have observed that the frontal method can be applied concurrently to the leaves of the elimination tree. Each processor assembles the submatrix corresponding to the row and column of a leaf of the elimination tree along with the resulting front. A processor can independently perform the Divide operation on its column and update its front. Multiple processors' fronts can overlap, effectively decoupling the multiplication and addition operations in the Update steps. While this prevents chaining of the arithmetic units of the processor, it still provides for long vectored multiplications. Furthermore, partial updates of the same location can be accumulated concurrently. An implementation has been proposed for the Denelcor HEP [1.41]. Processors synchronize and resolve data dependencies by communicating through the shared memory. For distributed-memory systems, this algorithm will suffer the same bottleneck as Geist's. To alleviate this, Duff has suggested that an increase in the granularity of the problem is warranted. This could be accomplished by assigning branches of the elimination tree to individual processors.

1.1.3 PROBABILISTIC SIMULATION

The current approach to aircraft design is deterministic and is based on an array of worst condition material properties, operating stresses, manufacturing defects, temperature and moisture of the operating environment, and service induced damage. Load conditions are increased by as much as 50 percent to provide a safety factor based on experience with metallic structures. Despite the best endeavors of designers, manufacturers, and users, engineers know that two identical-looking articles are never the same and fail at different lives. The deterministic approach can not accurately assess this variability in component failure.

The shortcomings of deterministic approach can be overcome by employing a probabilistic design (PD) approach. The foundation of PD involves basing design criteria objectives on quantified reliability targets. The benefits of PD are two-fold: the quantification of the structural reliability; and the ability to manage the structural risk through the identification of important design drivers. As a consequence, both weight savings and major reductions in life-cycle cost can be realized.

PD is not a replacement of well-founded deterministic methods, but rather an extension of them. The basis of probabilistics is the expression of the statistical range of a variable in the form of a probability distribution. This can be accomplished utilizing simulation methods such as Monte Carlo or Importance Sampling, or algebraic methods such as First- or Second-order Reliability, Mean-value Methods, or Response Surface Methods.

Economic benefits of using PD for structural analysis and design will result from incorporating efficient verification methods into structural analysis to; 1) reduce weight, 2) reduce operations and servicing costs, 3) reduce failure rates of structures, and 4) develop the capability to prepare predictable maintenance/overhaul schedules,

Probabilistic methods (PM), allow results of perturbations of a range of parameter values to be studied. PM provides a measure of parameter sensitivity and gives a realistic indication of component life by taking into account significant parameters; e.g. low cycle fatigue, flaw-induced cracking, yield and ultimate strengths, creep strength, operating environment, material flaws, and service damage.

The increasing focus on composite materials to provide higher-performance makes it important to give consideration to life prediction of composite materials. Experience gained from applying probabilistic methods to monolithic materials will provide an invaluable foundation for this, recognizing that the inherent weight and temperature advantages of composites entail a wider variation in material properties than is the case for monolithic materials. Uncertainties in variable values for fiber and

void volume ratios, ply misalignment, ply thickness, and the basic material properties of fiber and matrix are make composite properties more sensitive to uncertainties than is the case for the equivalent monolithic metallic materials. The use of a single value to represent each of these variables fails to adequately take into account these variable uncertainties. Development of a probabilistic approach for composites is, therefore essential. Application of the probabilistic design to composite material is illustrated in Figure 1-6.

Probabilistic methods (PM) enable incorporation of uncertainties and random variability, and consistent inclusion in computational models. PM technology determines the robustness of a design and rigorously analyzes the sensitivity of the failure risk to uncertainties (randomness and modeling uncertainties) in design parameters.

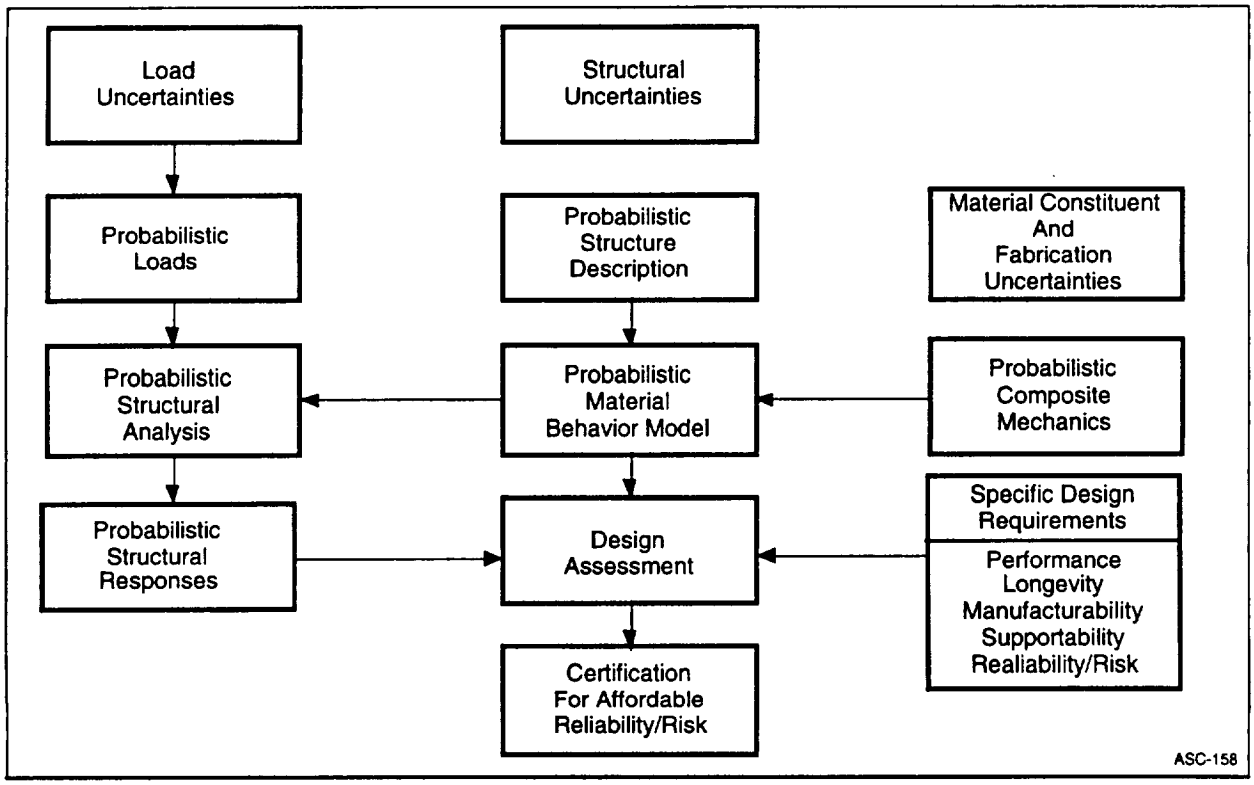


Figure 1-6. Block Diagram Approach For Applying Probabilistics to Composite Materials; The Diagram is Also Relevant to Monolithics.

1.2 GENOA SOFTWARE OVERVIEW

The inherent complexity of simulation of composite structure requires hierarchical multiple levels of interactive analysis utilizing time consuming convergence criteria that make design and analysis computing costly. Alpha Star Research and NASA Lewis Research Center under SBIR phase II contract has developed a computational tool GENOA, which meets industries future demand for expedience and reduced cost in the design and analysis of high temperature composite structures. The power of parallel processing and dynamic load balancing optimization used in GENOA has made the complex simulation of structure, material and processing of high temperature composite structure affordable. This unique software is dedicated to high speed analysis of next generation aerospace systems. GENOA is commercially viable software package that provides computational integration and parallel computing synchronization for the probabilistic mathematics, and structural/material mechanics used in the design and analysis of composite structures. GENOA handles the inherent complexity of high temperature composite structures by utilizing massive parallel processing and dynamic load balancing optimization techniques to speed the simulation processing time of a diverse field of specialized analysis techniques and mathematical models.

GENOA's highly modular architecture makes it fast, accurate, versatile and user friendly for use in a wide variety of applications. It provides rapid solution turn-around because of the parallel processing environment. Analytical performance is enhanced with the ability to dynamically size adjacent problem domains to minimizes processor wait time. Reduction of CPU time and memory limitations is accomplished by introducing an effective optimized parallelization algorithm employing machine independent Mult Instruction Multi Data (MIMD), Single Instruction Multi Data (SIMD), and Open Software Foundation (OSF) types of computer architecture. Hierarchical stochastic simulation is used to accommodate the numerous levels of uncertainty present in environmentally dependent material properties, enabling the user to quickly identify the most probable point of design criticality. These simulations are accompanied by an easy to use and highly visual graphical user interface (GUI)

In conjunction with the software technology and algorithms development, Alpha Star, is involved in the preparation of a data bank of composite material properties required for the simulation effort. This involvement consists of assisting NASA/LeRC in it's task of developing and compiling a massive database of materials DATABANK for metal matrix composites polymer matrix composites, and ceramic matrix composites. The use of this database in simulations allows test reduction and reliable optimization of advanced composite materials.

The unique capabilities of GENOA that make it superior to competing products are the following:

GENOA is the first completely commercially available integrated hierarchical Parallel Material structural and processing analysis software package. The value of integration cannot be over emphasized since most finite element analysis programs available in the market today are stand alone products which entail difficult data transference from a CAD/CAM environment

GENOA provides for an easy, seamless transition of data from SDRC/I-DEAS, PDA/PATRAN/ NASTRAN , programs used to produce or analyze a design (Figure 1-7).

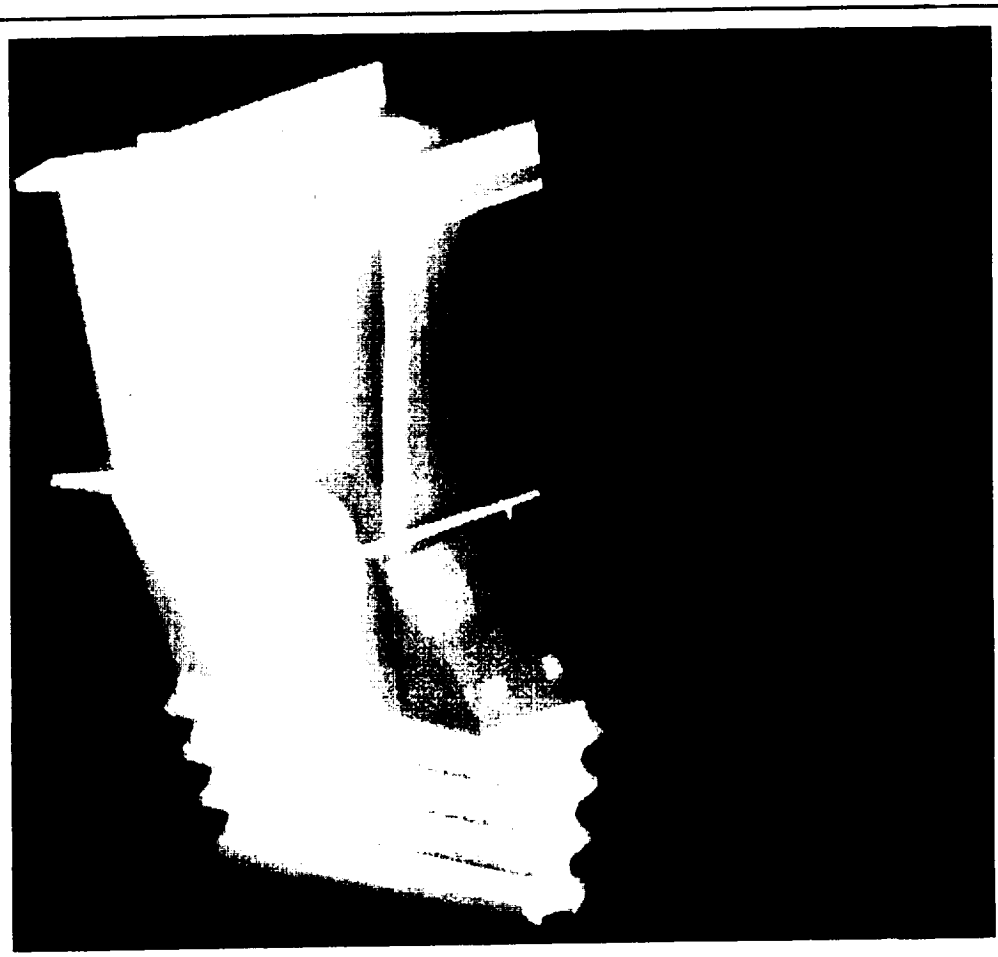


Figure 1-7. Space Shuttle Main Engine (SSME) Turbo Blade CAD Model Was Easily Transferred to GENOA for High Speed Computational Structural Analysis

The concurrent computing environment of GENOA is a superior multiprocessing resource for the design and analysis of the composite structure placing unequaled performance and analytical capability directly in the hands of engineers. It has removed memory limitations, greatly increased processing speed, and significantly facilitated data transference, and visualization of composite structures compared to existing programs.

1.3 PRODUCT STRATEGY

Combining development from government sponsored research with the commercialization of technology, Alpha Star has taken advantage of a rare opportunity and created an unchallenged, state-of-the-art software product under the trade name of *GENOA*. The *GENOA* software package represents the embodiment of knowledge gained during contracts with NASA and other government agencies, and in October of 1994, ASC began to position itself to commercialize the *GENOA* technology.

New composite materials are creating new markets in Industry (aerospace, automotive, semiconductor chip, etc.). Traditional techniques for design modeling and testing are too cumbersome, costly and time consuming for these new materials. The *GENOA* family of products meets the challenge of the new materials by employing the latest innovative computational integration and synchronization capabilities of probabilistic mathematics, structural/material mechanics, and parallel computing to the design and analysis of high temperature composite structures.

GENOA's development pushes the state-of-the-art in four key areas: 1) composite analysis modeling, 2) probabilistic simulation, 3) finite element analysis, and 4) computer processing speed.

- *GENOA* is a truly revolutionary software package for the engineer who designs and manufactures advanced aerospace and automotive products. Dedicated to the high speed analysis of next generation materials, the *GENOA* software package employs leading edge technology from the fields of composites, structures and parallel computing sciences to deliver unequaled performance and analytical capability in three general categories:

- Consolidation of Design and Manufacturing of Composite Process for Metal Matrix Composites (MMC), Ceramic Matrix Composites (CMC), and Polymer Matrix Composite (PMC)

- Simulation of Failure/Life Prediction for MMC, CMC, and PMC. It is the only software to successfully incorporate Finite Element Analysis and Failure Probability Analysis.

- Real Time Parallelization

GENOA is a modular, heterogeneous software system that can be easily ported to any hardware platform using a UNIX operating system. Several stand alone modules are offered as a package or selectively purchased according to the users specific needs. The Graphic User Interface (GUI) and Executive Controller System (ECS) together make up the main driver for one or all of the modules and are a required part of GENOA. The GUI provides the pre and post processor visualization capability while the menu driven ECS connects all the modules. The following is a brief description of GENOA's drivers and modules.

GENOA Driver

Graphic User Interface (GUI) - A visualization system based on ICONS. The GUI is written in C language and employs a standard graphical library such as X11 Motif and/or Phigs.

Description: GUI (Graphic User Interface): an enhanced color graphic user interface with menu driven and on-line help options spanning from input deck preparation, postprocessing plots, and contour plots of microstresses to failure probability distribution function plots.

Functionality: The main functions of GUI are to provide visualization to view results, to import CAD models from SDRC and PATRAN, and to export data to other software systems such as NASTRAN.

Portability: SUN, Silicon Graphics, Hewlett Packard, and IBM RISK workstation.

Executive Controller System (ECS) - A menu driven panel system. The ECS is written in FORTRAN 77 language

Description:

Functionality: The ECS has two main functions: 1) provide communication between modules and 2) check the validity of the models in the data base. Communication consists of as automatically defining and processing the input to and from each module, checking the validity of the data and model, and processing the output for the GUI and other modules. The ECS also assists the user in creating and preparing models for each module.

Portability: SUN, Silicon Graphics, Hewlett Packard, and IBM RISK workstation.

RIP and PEEL (optional) - A static load balancing system based on Recursive Inertia Partitioning and interactive user input. RIP and PEEL is written in C language and employs a standard graphical library such as X11 Motif and/or Phigs.

Description:

Functionality: Partitions the model into discritized static load balance. It creates a mathematical, and/or natural subdivisions between the divisions and super elements.

Portability: SUN, Silicon Graphics, Hewlett Packard, and IBM RISK workstation.

MAESTRO: A dynamic load balancing system with a real time parallelization algorithm. MAESTRO is written in FORTRAN 77 language and provides the power of parallel processing to the GENOA user.

Description: Dynamic Load Balancing Computing Environment: a multi-factor optimizer providing a dynamic platform that continuously tests facility resources, distributes available computer resources, monitors machine utilization and analysis progress, controls environmental conditions in real time, and redistributes operational tasks in real time.

The concurrent computing environment is achieved by four key elements: (1) estimating and optimizing the requested memory and time interface connection among processors, (2) automatically analyzing and reconfiguring algorithms for the distributed and massively parallel architecture, (3) monitoring by generating memory and time allocation tables, and (4) managing data storage, work load balance, inter processor communication, buffering, and message exchange among the active and non-active processors.

Multiple Design Optimization (MDO) offers an unprecedented framework for creating a unique parallelized and optimized system for multiple instruction multiple data (MIMD) or single instruction multiple data (SIMD) computer architecture. A constrained optimization problem is formulated for which minimization of CPU time subject to the parallelism constraints (e.g. available RAM, model size, global calculation time) is to be satisfied. The power of MDO becomes readily apparent in a universal concurrent computing environment able to provide the needed accuracy and design speed.

Functionality: Minimization of CPU and wall clock time based on the Optimization of computer/processors resources (memory management, disk space, communication rate, and real and integer operation). MAESTRO complements Parallel Virtual Machine (PVM) by directing it's activity.

Portability: IBM-SP2, CONVEX, nCUBE, Hewlett Packard cluster workstations, SUN, Silicon Graphics, Hewlett Packard, and IBM RISK workstation.

Parallel Finite Element Module (PFEM) - A linear and nonlinear, static and dynamic finite element based solver for thermo-mechanical problems. PFEM implements the material (composite and metallic) data bank library based on NASA and vendors test data. PFEM is written in FORTRAN 77 language.

Description: *GENOA*/FEM (Finite Element Method): a highly versatile, state-of-the-art finite element code dedicated to fast and efficient large-scale computing. High speed linear and nonlinear analysis of static and dynamic problems and sensitivity analysis of large structures make the *GENOA*/FEM a leading composite analyzer. Its economic application is accomplished by combining the latest technology, such as the mixed-iterative method which is known for accuracy in stress/strain solutions and displacement results.

Functionality: Composite orthotropic element capability.

Portability: IBM-SP2, CONVEX, nCUBE, Hewlett Packard cluster workstations, SUN, Silicon Graphics, Hewlett Packard, and IBM RISK workstation.

Material Constituent Analysis (MCA) - Determines the reliability of a material system to construct a structure. MCA supports the constituent analysis of a laminate and substrate composite system.

Description: Resident Data Bank: the most complete and current resident data bank available containing the test data of constituent material properties. The data bank encompasses the thermomechanical properties (e.g. elastic modules, strength, Poisson's ratio, coefficient of thermal expansion, heat capacity, etc.) of several most commonly used PMC systems, including T-300, AS graphite, S-Glass, IIMS fibers, and epoxy-type resins matrix with different levels of modules and strength.

Functionality: This module performs the simulation of composite laminate fracture toughness and degradation in laminate structural integrity in terms of strain energy release rate, displacement, loss in stiffness, loss in vibrating frequencies, and loss in buckling resistance. Various types of delamination and boundary conditions can be simulated. Three composite material systems are supported.

Portability: IBM-SP2, CONVEX, Hewlett Packard cluster workstations, Silicon Graphics, Hewlett Packard, and IBM RISK workstation.

Ceramic Module- Calculates the lamina and laminate material property degradation due to environmental, manufacturing, and service thermo-mechanical loading conditions. It is written in FORTRAN 77 language and implements the ceramic data bank library based on NASA and vendors test data. Ceramic module calculates material properties and performs the probability distribution, cumulative distribution functions, and probability of failure.

Portability: IBM-SP2, CONVEX, Hewlett Packard cluster workstations, Silicon Graphics, Hewlett Packard, and IBM RISK workstation.

Polymer Module - Calculates the lamina and laminate material property degradation due to environmental, manufacturing, and service thermo-mechanical loading conditions. It is written in FORTRAN 77 language and implements the polymer data bank library based on NASA and vendors test data. Polymer module calculates material properties and performs the probability distribution, cumulative distribution functions, and probability of failure.

Metal Matrix Module - Calculates the lamina and laminate material property degradation due to environmental, manufacturing, and service thermo-mechanical loading conditions. It is written in FORTRAN 77 language and implements the ceramic data bank library based on NASA, and vendors test data. Metal Matrix module calculates material properties and performs the probability distribution, cumulative distribution functions, and probability of failure.

System Reliability Analysis (SRA) - Performs confidence level analysis for a system involving material, structure, and the manufacturing process. SRA calculates the probability distribution, cumulative distribution functions, and probability of failure for a component probability. SRA is written in FORTRAN 77 language.

Description: provides for the probabilistic treatment of the MFIM to account for the statistical nature of the diverse effects. It randomizes these diverse effects based on the user specified distribution type, using Monte Carlo simulation. The probabilistic treatment of the MFIM yields a cumulative probability distribution function (CDF) from which a probabilistic evaluation of material property degradation can be obtained. An interface is provided for the users to select the sampling size, the critical region where the information of failure probability is desired, and the distribution type of each variable in the MFIM. The distribution types include: 1) normal distribution, 2) logarithm normal distribution, and 3) Weibull distribution

Functionality: Probabilistic component failure analysis.

Portability: IBM-SP2, CONVEX, Hewlett Packard cluster workstations, Silicon Graphics, Hewlett Packard, and IBM RISK workstation

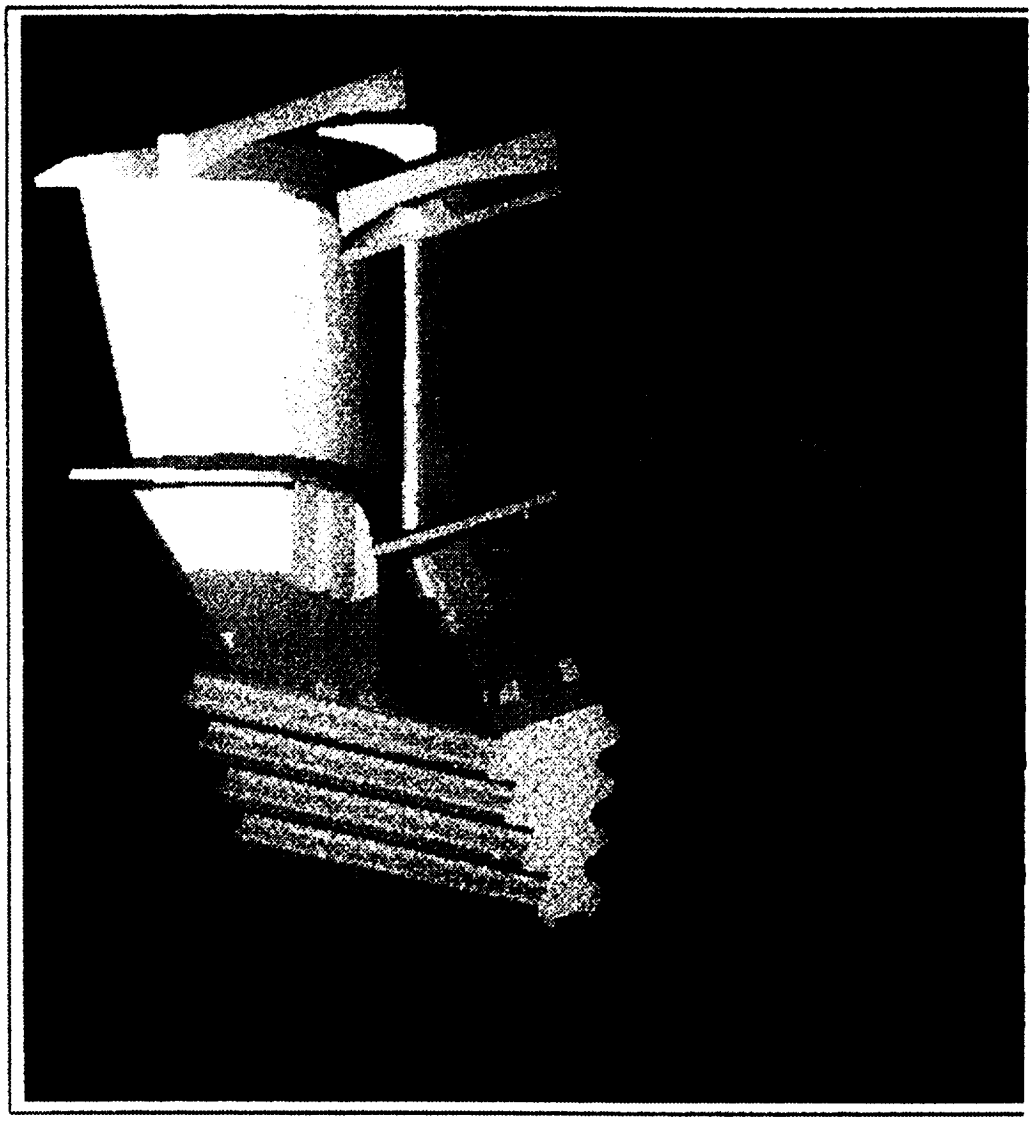
MFIM (Multi-Factor Interaction Model): an industry accepted model which simulates the nonlinear degradation of constituent properties due to a number of diverse effects and their mutual interactions. The MFIM model is in product form of these effects, including static stress, temperature, thermal and mechanical fatigue, acoustic fatigue, creep, chemical reaction, thermal shock, mechanical impact, and time. The user can initiate either a single factor analysis (e.g., mechanical fatigue analysis) or a multi-factor analysis to account for the mutual interactions. The global stiffness matrix is then updated from the resulting constituent material property degradation for the next iteration of finite element analysis.

GENOA incorporates translators both into and out of the programs so it is transparent to widely used software programs such as SDRC's I-Deas and MacNeal Schwindler's Patran. Future product development will include translators for ProEngineer and CATIA.

GENOA

A look into the *FUTURE* with High Speed Computational Structural Analysis

GENOA



Alpha STAR Corporation
Simulation Technology & Advanced Resea

GENOA

GENOA is a truly revolutionary software package for the engineer who designs and manufactures advanced aerospace and automotive products.

Dedicated to the high speed analysis of next generation materials, the **GENOA** software package employs leading edge technology from the fields of composites, structures and parallel computing sciences to deliver unequaled performance and analytical capability directly into the hands of todays engineers.

Development of the **GENOA** capability is presently underway at the Alpha STAR Corporation in cooperation with the NASA Lewis Research Center.

Alpha STAR recognizes that the implementation of new materials is closely coupled with the advancement of analytical tools enabling intelligent selection of fabrication parameters and the immediate realization of benefits for final component design. As a result, the **GENOA** package is structured to embody an array of specialized modeling techniques which provide the insight and analytical power necessary to meet the ever increasing demand for a leading edge understanding of structural response.

GENOA features a modular architecture making it fast, accurate, user friendly and highly versatile to adapt to needs of your company.

GENOA features include:

Parallel Processing – in either distributed or multi–processing systems. **GENOA** provides rapid solution turn–around through the added performance of the parallel processing environment. The analytical performance is further enhanced with the ability to dynamicly size adjacent problem domains which minimizes processor wait time.

Stochastic Simulation – to accommodate the numerous levels of uncertainty inherent to all aspects of structural response analysis, enabling the user to quickly identify the most probable points of design criticality.

Hierarchical Simulation – delivering iteratively computed, environmentally dependent material properties.

Nonlinear Finite Element Solver – assuring accurate and expedient convergence while implementing the appropriate constitutive equations.

Graphical User Interface – providing the seemless transition from problem description, solver implementation and post–processing visualization. Catering to SDRC/I–DEAS and PDA/Patran.

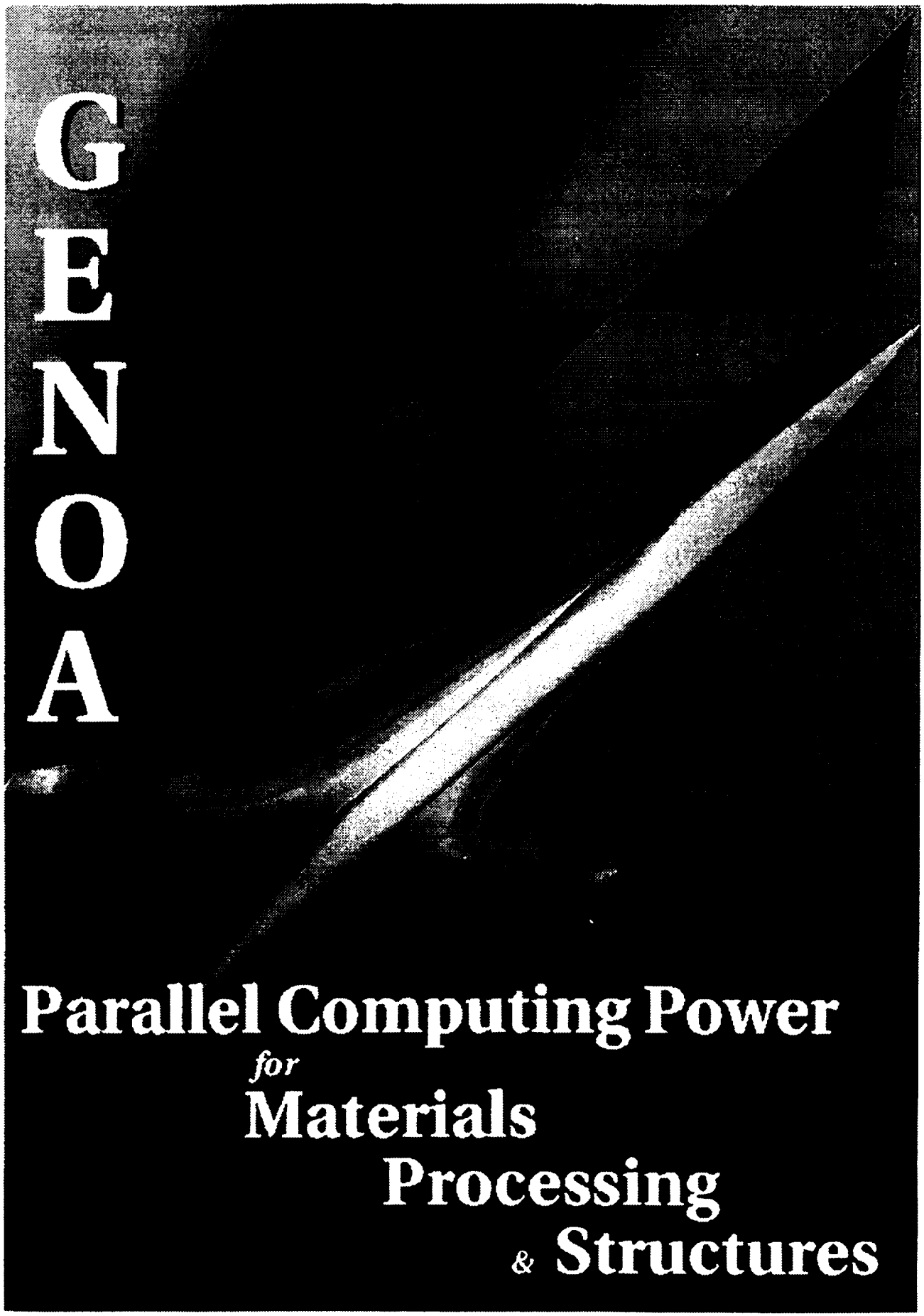
Object Oriented Sub–programming – guaranteeing the appropriate utilization of parallel utilities without requiring user knowledge of higher level programming routines.

Hardware Support – designed for UNIX platforms such as: HP700 series in a distributed architecture, nCUBE and the CONVEX Meta Series.

For Additional Information contact:

Alpha STAR Corporation

5200 West Century Boulevard, Suit 340
Los Angeles, California 90045
(310) 417–8547 Fax (310) 417–8546
E–mail acs@metcom.com


GENOA
Parallel Computing Power
for
Materials
Processing
& Structures

GENOA/MAESTRO (Pre Calclulated & Real time parallelization Algorithm) is a state of the art algorithm that when combined with multilevel design optimiztion (MDO) methodology, provides speed and efficiency not previously obtained in the design and analysis of large scale, high temperature, composite structure systems.
Simulations utilizing next generation parallel computing architecture require a balance between the demands of the structural model and the available computing resources. Multidisciplinary optimization techniques combine with MESTRO to bring together a diverse field of specialized analysis techniques and mathematical models that minimize total CPU time. Our technique is designed to exploit the availability of multiple processors while performing computationally intense probabilistic analysis involving both structurally based finite element analysis and advanced composite micro mechanics.

MULTILEVEL DESIGN OPTIMIZATION

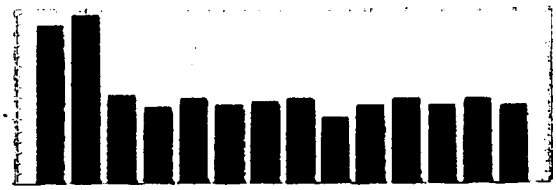
GENOA/PRPA's unprecedented framework for creating a unique parallelized and optimized system for multiple instruction multiple data (MIMD) or single instruction multiple data (SIMD) computer architecture lies with the MDO technique. A constrained optimization problem is formulated for which minimization of CPU time subject to the parallelism constraints (e.g., available RAM, model size, global calculation time) is to be satisfied. The power of MDO becomes readilly apparent in a universal concurrent computing environment able to provide the needed accuracy and design speed.

DYNAMIC COMPUTING ENVIRONMENT

GENOA/PRPA and multi-factor optimization is a dynamic platform that continuosly test facility resources. distributes available computer resources, monitors machine utilization and analysis progress, controls enviromental conditions in real time, and redistributes operational tasks in real time.

REMOVING RESOURCE LIMITATIONS

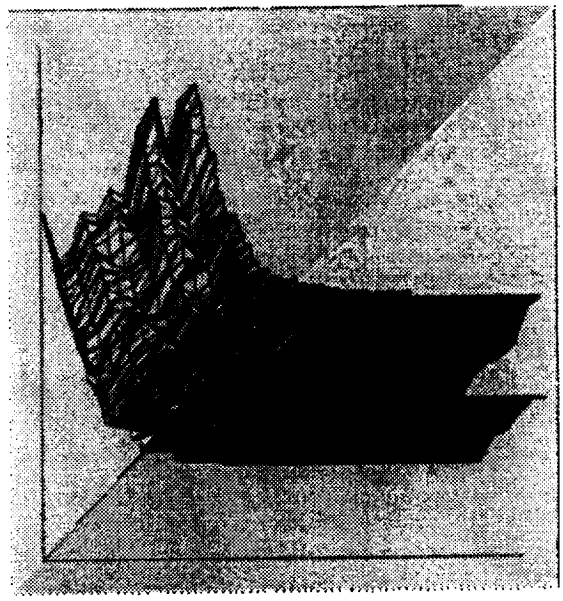
GENOA/MAESTRO development has been directed toward reducing limitations imposed by restricted computer memory and large scale structural models. The PRPA tools take advantage of multiple processor programming and available Parallel Virtual Machine (PVM) access (such as nCUBE2, CONVEX C3800, IBM SP2, and integrated HP 700 series workstations). The first step iin the MAESTRO optimization process is the selection of a set of processor resources (real and
virtual) for the inherent connection to the searator tree operations. The last step is to control and manage the best combination of available computer resources such as machine type, network level, number of processors, available memory, and inter processor communication message maps to exploit the optimal combination.



Processors Activity is Monitored Interactively

DEMONSTRATED SUCCESS

The success of GENOA/MAESTRO has been demonstrated on a hypothetical Large Scale High Speed Civil Transport (HSCT), and Space Shuttle Main Engine (SSME) Blade finite element models. It is observed that implementation of MAESTRO result in further substantial performance gain with increased problem size.



Convergence of MAESTRO is Monitored in real time

For additional information, please contact:

Alpha STAR Corporation
5200 W. Century Boulevard, #340
Los Angeles, CA 90045
Ph: (310) 417-8547
Fax: (310) 417-8546
email: asc@netcom.com

Alpha STAR Corporation
Simulation Technology & Advanced Research

GENOA/METCAN (Metal Matrix Composite Analyzer) is a highly advanced simulation code which performs computational simulation of nonlinear high temperature behavior of intermetallic matrix composites. The cutting–edge predictive capability of GENOA/METCAN consists of several modules encompassing constituent material nonlinear behavior, composite mechanics, and finite element analysis. GENOA/METCAN is a unique, state–of–the–art, user friendly, command and menu driven package with interactive color graphics of instantaneous material properties, stresses, strains, and regional failure probabilities. GENOA/METCAN provides scientists and engineers a never before available desk–top laboratory, enabling not only the life prediction of HT–MMCs, but also the tailoring of HT–MMCs performance by optimization of processing conditions.

MATERIAL PROPERTY DEGRADATION

GENOA/METCAN utilizes the industry accepted multi–factor interaction model (MFIM) which simulates the nonlinear degradation of constituent properties due to a number of diverse effects and their mutual interactions. The MFIM model is in product form of these effects, including: static stress, temperature, thermal and mechanical, fatigue, acoustic fatigue,creep, oxidation,thermal shock, mechanical impact and time. The user can initiate either a single factor analysis (e.g., mechanical fatigue analysis) or a multi–factor analysis to account for the mutual interactions. The global stiffness matrix is then updated from the resulting constituent material property degradation for the next iteration of finite element analysis.

PROBABILISTIC FAILURE ANALYSIS

GENOA/METCAN provides for the probabilistic treatment of the MFIM to account for the statistical nature of the diverse effects. It randomizes these diverse effects based on the user specified distribution type, using Monte Carlo simulation. The probabilistic treatment of the MFIM yields a cumulative probability distribution function (CDF) from which a probabilistic evaluation of material property degradation can be obtained. GENOA/METCAN efficiently provides the users an interface to select the sampling size, the critical region where the information of failure probability is desired, and the distribution type of each variables in the MFIM. The distribution types include:

- normal distribution
- logarithm normal distribution
- Weibull distribution

The user can invoke the probabilistic simulation for each constituent in some critical regions when the stress value is close to the failure criteria. The resulting CDF provides a probabilistic evaluation of the constituent failure.

METAL MATRIX LAMINATE TAILORING(MMLT)

The MMLT code is coupled with GENOA/METCAN for a never before offered concurrent tailoring of the constituent material characteristics and the fabrication process to achieve a priori specified HT–MMC behavior such as maximum fatigue life and minimum residual stress upon cool–down. This unique and long desired feature enables the engineers to quantify the strong coupling between the fabrication process conditions and the nonlinear thermomechanical response of MMC during the subsequent service, as well as significantly reduce the manufacturing and testing costs.

RESIDENT TEST DATABANK

GENOA/METCAN has the most complete and current resident databank available containing the test data of constituent material (e.g. fiber, matrix and interphase) properties. The databank encompasses the thermomechanical properties (e.g. elastic modules, strength, Poisson's ratio, coefficient of thermal expansion, heat capacity, etc.) of several most commonly used MMC systems, including P100/Cu, SiC/Ti–15–3, SiC/Ti–6–4, and SiC/Ti–24Al–11Nb.

GRAPHIC USER INTERFACE

GENOA/METCAN has an enhanced color graphic user interface with menu driven and on–line help options spanning from input deck preparation, postprocessing plots, and contour plots of microstresses to failure probability distribution function plots.

HARDWARE/SOFTWARE SUPPORT

GENOA/METCAN has been developed on a HP–UNIX platform. It can be installed on any workstation or PC system supported by UNIX. X–Windows and Motif with only slight modifications.

For additional information, please contact:

Alpha STAR Corporation
5200 W. Century Boulevard, #340
Los Angeles, CA 90045
Ph: (310) 417–8547
Fax: (310) 417–8546
email: asc@netcom.com

Alpha STAR Corporation
Simulation Technology & Advanced Research

GENOA/ICAN (Polymer Matrix Composite Analyzer) is a highly advanced simulation code which performs computational simulation of thermomechanical behavior and material property degradation of structural polymer composites in both component and constituent levels. The cutting–edge predictive capability of GENOA/ICAN consists of several modules encompassing constituent material nonlinear behavior, composite mechanics, and finite element analysis. GENOA/ICAN is a unique, state–of–the–art, user friendly, command and menu driven package with interactive color graphics of instantaneous material properties, stresses, strains, and regional failure probabilities. GENOA/ICAN provides scientists and engineers a never before available desk–top laboratory, enabling the stress analysis and life prediction of structural polymer matrix composites.

MATERIAL DEGRADATION

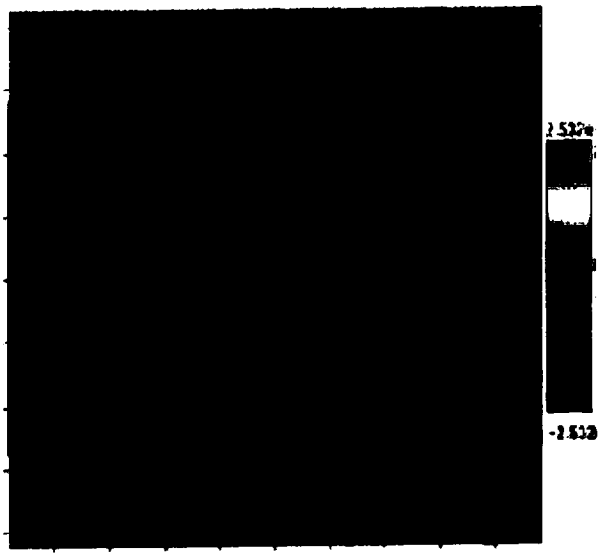
GENOA/ICAN utilizes the industry accepted multi–factor interaction model (MFIM) which simulates the nonlinear degradation of constituent properties due to a number of diverse effects and their mutual interactions. The MFIM model is in product form of these effects, including static stress, temperature, thermal and mechanical fatigue, acoustic fatigue, creep, chemical reaction, thermal shock, mechanical impact, and time. The user can initiate either a single factor analysis (e.g., mechanical fatigue analysis) or a multi–factor analysis to account for the mutual interactions. The global stiffness matrix is then updated from the resulting constituent material property degradation for the next iteration of finite element analysis.

ACOUSTIC FATIGUE ANALYSIS

The acoustic fatigue analysis simulates the situation in which a vibrating component generates acoustic noise, causing the dynamic response and degradation of material properties of an adjacent component. This feature of GENOA/ICAN enables engineers to calculate the acoustic noise level emanating from a vibrating structural component, degradation in material properties of multilayered composite laminates, dynamic response of acoustically excited composite structure, degradation in the first ply failure strength due to acoustic loading, and acoustic fatigue resistance of the excited structure.

GENERAL DELAMINATION ANALYSIS

GENOA/ICAN can perform the simulation of composite laminate fracture toughness and degradation in laminate structural integrity in terms of strain energy release rate, displacement, loss in stiffness, loss in vibrating frequencies, and loss in buckling resistance. Various types of delamination and boundary conditions can be simulated.



Postprocessing of microstresses due to failure Probability distribution

PROBABILISTIC FAILURE ANALYSIS

GENOA/ICAN provides for the probabilistic treatment of the MFIM to account for the statistical nature of the diverse effects. It randomizes these diverse effects based on the user specified distribution type, using Monte Carlo simulation. The probabilistic treatment of the MFIM yields a cumulative probability distribution function (CDF) from which a probabilistic evaluation of material property degradation can be obtained. An interface is provided for the users to select the sampling size, the critical region where the information of failure probability is desired, and the distribution type of each variables in the MFIM. The distribution types include: normal, logarithm normal, Weibull.
GENOA/ICAN color graphics displays contour plots of the instantaneous microstresses. The user can invoke the probabilistic simulation for each constituent in some critical regions when the stress

Alpha STAR Corporation
5200 W. Century Boulevard, #340
Los Angeles, CA 90045
Ph: (310) 417–8547
Fax: (310) 417–8546
email: asc@netcom.com

Alpha STAR Corporation
Simulation Technology & Advanced Research

GENOA/PFEM (Parallel Finite Element Method) is a highly versatile, state-of-the-art finite element code dedicated to fast and efficient large-scale computing. High speed linear and nonlinear analysis of static and dynamic problems and sensitivity analysis of large structures make the GENOA/PFEM a leading composite analyzer. Its economic application is accomplished by combining the latest technology such as the Multi Frontal, and mixed-iterative method which is known for memory reduction, and accuracy in stress/strain solutions and displacement results.

GENOA/PFEM DRIVERS
- Transient dynamics by mode superposition
- Transient dynamics by direct integration
- Static analysis in one or more increments
- Harmonic and random excitation
- Dynamics eigenvalue problems
- Buckling eigenvalue problems

SOLUTION ALGORITHMS
- Incremental Frontal-iterative for static problems
- Direct integration for transient dynamics problems using generalized Newmark solution algorithm to march in time
- Subspace iteration procedure for linearized buckling and dynamic eigenvalue problems
- Jacobi iteration for reduced eigenvalue problems
- Standard methods of linear dynamics for transient dynamics, and both harmonic and random excitation problems.

CONSTITUTIVE MODELS
The conventional von Mises plasticity model is employed by default unless the user exercised the programs versatility and selects one of the following available models:
- Standard linear elasticity model for experimental purposes
- Simplified plasticity (secant elasticity) model using the material tangent with Newton-Raphson type iteration
- Classical von Mises J2-flow plasticity model using the radial return algorithm to treat the associated flow rule
- Walker's nonlinear viscoplastic model using an initial stress iteration

FINITE ELEMENT LIBRARY
The GENOA/FEM modeling elements include industry standard continuum type elements (plain strain, plain stress, axisymmetric, three-dimensional) based on both the use of independent strain interpolations and the use of selective reduced integration techniques. GENOA/FEM also offers extended capability through two degenerate continuum elements which account for transverse shear deformations. These are beam and shell elements that are derived from Timoshenko beam equations and Reissner-Mindlin theory, respectively.

MIXED-ITERATIVE METHODS
The GENOA/FEM is based on a progressive mixed-iterative finite element formulation derived from a three-field Hu-Washizu variational principle. Stress/strain equations at selected sampling points, which need not necessarily correspond to the element integration points, can be added at the users discretion to the familiar displacement equations without significant increase in problems size.

The choice of a three-field mixed-iteration formulation yields accurate values of displacements, strains, and stresses at all nodes of the finite element mesh. This increased accuracy improves the set of stress/strain interpolations, eliminates inter-element stress discontinuities, and decreases the number of assembled equations to displacements.

PERTURBATION ANALYSIS
The sensitivity analysis of the structural response to small perturbations of the random input variables is provided by a fast and accurate perturbation analysis algorithm. The algorithm is residual-driven and will never converge to a wrong solution. Besides the perturbation expansion, a subspace iteration solution is also available for special cases.

INTERFACE
Many shape and geometry parameters, material properties, temperature and pressure data are specified directly at the mesh nodes when nodal data input and output are used. This permits the use of a very simple interface to existing statistical properties databases and grid-based laboratory data acquisition techniques. The combination of nodal data approach and mixed-iterative finite element formulation is highly desirable feature from and interface design standpoint.

For additional information, please contact:

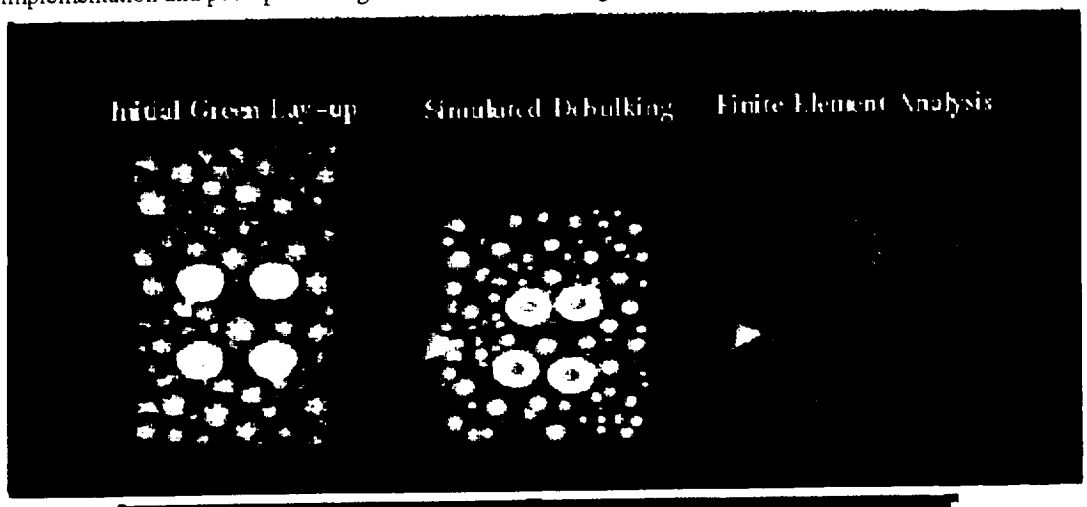
Alpha STAR Corporation
5200 W. Century Boulevard, #340
Los Angeles, CA 90045
Ph: (310) 417-8547
Fax: (310) 417-8546
email: asc@netcom.com

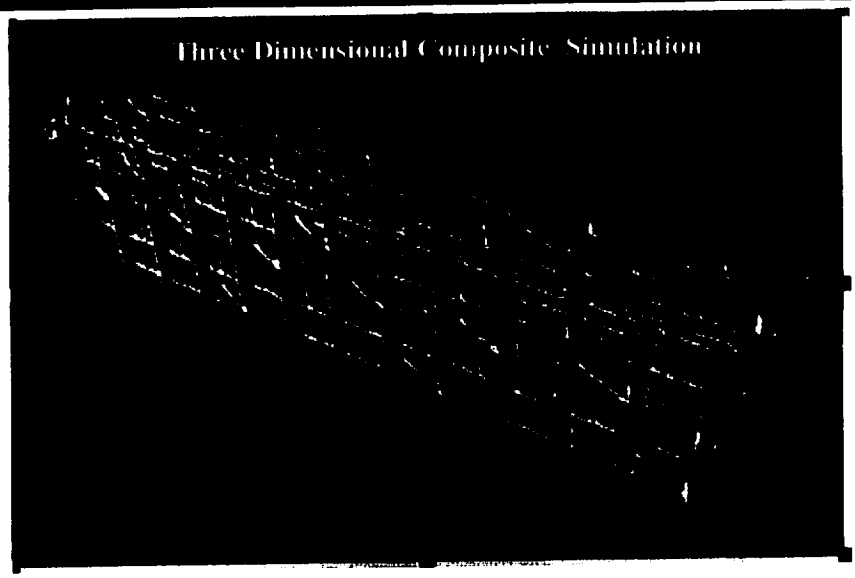
Alpha STAR Corporation
Simulation Technology & Advanced Research

GENOA's Graphical User Interface

GENOA, a commercially viable software package under development by Alpha STAR Corporation and NASA Lewis Research Center, combines the latest innovative computational integration and synchronization capabilities of probabilistic mathematics, structural/material mechanics, and parallel computing to the design and analysis of high temperature composite structures.

The graphical user interface provides the seemless transition from problem description, solver implementation and post–processing visualization. Catering to SDRC/I–DEAS and PDA/Patran.





Alpha STAR Corporation
Simulation Technology & Advanced Research

1.5 REFERENCES

1.1. Gustafson J. L., Montry G. R. and Benner R.E., " Development of Parallel Methods for a 1024-Processor Hypercube, "SIAM Journal on Scientific and Statistical Computing, Vol. 99, No. 4, pp. 609-638, 1988

1.2. Amdhal, G., Validity of the single processor approach to archiving large-scale computer capabilities, AFIPS Conf. Proc., pp. 438-485, 1967

1.3. H. H. Goldstine and A. Goldstine, "The Electronic Numerical Integrator and Computer (ENIAC)," MTAC, pp. 97-110, 1947

1.4. R. Rice and W. R. Smith, "SYMBOL-A Major Departure from Classic Software Dominated von Neumann Computing Systems," Proceedings of the 1971 Spring Joint Computer Conference, Mantvale, NJ, pp. 575-587, 1971

1.5. P. Kogge, "The Architecture of Pipelined Computers," McGraw-Hill, New York, NY, 1981

1.6. E. Bloch, "The Engineering Design of the STRETCH Computer," Proceedings of the Eastern Joint Computer Conference, pp. 48-59, 1959

1.7. J. Hennessy, "VLSI Processor Architecture," IEEE Transactions on Computers, Vol. C-33, No. 12, pp.1221-1246, 1984

1.8. Control Data STAR-100 Computer-Hardware Reference Manual, Control Data Corp., 1974

1.9. P.H. Enslow, "Multiprocessor Organization - A Survey," in Tutorial on Parallel Processing, ed.D.A. Padua, pp. 5-6, IEEE Computer Society, Aug. 1981

1.10. M. J. Flynn, "Very High Speed Computing Systems," IEEE Proceedings, Vol. 54, No. 12, pp. 97-103, Dec. 1966

1.11. D. L. Zlotnick, "The SOLOMAN Computer," AFIPS Conference Proceedings FJCC, Vol. 22, pp. 97-107, 1962

1.12. D. J. Kuck, "ILLIAC-IV Software and Applications Programming," IEEE Transactions on computers, Vol. C-17, pp. 758 - 770, Aug. 1960

1.13. D. J. Kuck, "The Structure of Computers and Computations," John Wiley and Sons, New York, NY, 1978

1.14. K. E. Batcher, "STARAN Parallel Processor System Hardware," Proceedings of the National Computer Conference, pp. 405 - 410, 1974

1.15. K. E. Batcher, "MPP - A Massively Parallel Processor," Proceedings of the 1979 International Conference on Parallel Processing, p. 249, 1979

1.16. W. D. Hillis, "The Connection Machine, MIT Press," Cambridge, MA, 1985

1.17. V, Zakharov, "Parallelism and. Array Processing," IEEE Transactions on Computers, Vol.C-33, no. 1, pp. 45 - 78, Jan. 1984

1.18. S. Lundstrom, "Applications Considerations in the System Design of Highly Concurrent Multiprocessors," IEEE Transactions on Computers, Vol. C-36, No. 11, pp. 1292 - 1310, Nov. 1987

1.19. J. J. Dongarra and I. S. Duff, "Advanced Architecture Computers, Mathematics and Computer Science Division," Argonne National Laboratory, Argonne, IL, Aug. 1985

1.20. A. H. Karp, "Programming for Parallelism," IEEE Computer, Vol. 20, No. 5, pp. 43 -57, May 1987

1.21. C. Wu and T. Feng, "Tutorial: Interconnection Networks for Parallel and Distributed Processing," IEEE Computer Society, 1984

1.22. W. A. Wulf and C. G. Bell, "C.mmp - A Multi-Mini-Processor," AFIPS Conference Proceedings, Vol. 41, part II, FJCC, pp. 765 - 777, 1972

1.23. R. Perron and C. Mundie, "The Architecture of the Alliant FX/8 Computer," Compcon Digest, pp. 390 - 393, 1986

1.24. B. Smith, " A Pipelined, Shared Resource MIMD Computer", Proceedings of the International Conference on Parallel Processing, pp. 6 - 8, 1978

1.25. "Butterfly Parallel Processor Overview," BBN Laboratories Inc., 1985

1.26. A. Gottlieb, "An Overview of the NYU Ultracomputer Project," Ultracomputer Note #100, Courant Institute of Mathematical Sciences, New York University, New York, NY, July, 1986

1.27. G. F. Pfister, "The IBM Research Parallel Processor Prototype (RP3): Introduction and Architecture," Proceedings of the International Conference on Parallel Processing, pp. 764 - 771, Aug., 1985

1.28. T. Hoshino, "PACS: A Parallel Microprocessor Array for Scientific Calculations," ACM Transactions on Computer Systems, Vol. 1, No. 3, pp. 195 - 221, 1983

1.29. C. L. Seitz, "The Cosmic Cube," Communications of the ACM, Vol. 666, No. 666, p.666, Jan. 1985

1.30. P. Wiley, "A Parallel Architecture Comes of Age at Last," IEEE Spectrum, Vol. 24, No. 6, pp. 46 - 50, June 1987

1.31. "Balance 8000/21000 Guide to Parallel Programming: Users Guide," Sequent Computer Systems, Inc., Portland, OR, July, 1985

1.32. R. J. Swan, S. H. Fuller, and D. P. Siewiorek, "Cm* - A Modular, Multiminicomputer," Proceedings of the National Computer Conference, Vol. 46, pp. 637 - 644, 1977

1.33. C. Rieger, "ZMOB: Hardware from a User's Viewpoint," Proceedings of the Conference on Pattern Recognition and Image Processing, pp. 309 - 408, 1981

1.34. P. Moller-Nielsen and J. Staunstrup, "Experiments With a Multiprocessor," Technical Report DAIMI PB - 185, Computer Science Department, Aarhus University, Nov. 1984

1.35. A. Kapauan and K. Wang, J. Cuny, and L. Snyder, "The Pringle: An Experimental System for Parallel Algorithm and Software Testing," Proceedings of the International Conference on Parallel Processing, pp. 1 - 6, 1984

1.36. J. M. Ortega and R. G. Voigt, "Solution of Partial Differential Equations on Vector and Parallel Computers," SIAM Review, Vol. 27, No. 2, pp. 149 - 240, June 1985

1.37. J. Peterson and A. Silberschatz, "Operating System Concepts," Addison-Wesley, Menlo Park, CA, 1983.

1.38. P. Sadayappan and V. Visvanathan, "Circuit Simulation on a Multiprocessor," Proceedings of the Custom Integrated Circuits Conference, pp. 124 - 128, 1987

1.39. I. S. Duff and J. K. Reid, "The Multifrontal Solution of Indefinite Sparse Symmetric Linear Equations," ACM Transactions on Mathematical Software, Vol. 9, No. 3, pp. 302 - 325, Sept. 1983

1.40. B. M. Irons, "A Frontal Solution Program for Finite Element Analysis," International Journal of Numerical Methods in Engineering, Vol. 2, pp. 5 - 32, 1970

1.41. B. Smith, "A Pipelined, Shared Resource MIMD Computer," Proceedings of the International Conference on Parallel Processing, pp. 6 - 8, 1978.

1.42. I. Ergatoudis, B. M. Irons, and O. C. Zienkiewicz, "Three Dimensional Analysis Of Arch Dams And Their Foundations," Symposium on Arch Dams, Instn, civ Engng, London, 1968

1.43. I. S. Duff, "Design Features Of A Code For Solving Sparse Unsymmetric Linear Systems out-of-core," SIAM J. Sci. Stat. Comput., Vol. 4, 1983.

1.44. I. S. Duff, "MA32-A Package For Solving Sparse Unsymmetric Systems Using The Frontal Methods," Harwell Rep. AERE R. 10079, HMSO, London, 1981

1.46. R. Calalo, B. Nour-Omid, and M. Wright, "An Implementation Of The Multifrontal Solution Algorithm on MIMD Computers," AIAA, pp. 1-9, 1992

2.0 Objectives And Scope

Technological developments of the late 1980s and early 1990s have set an ideal stage for the timely introduction and acceptance of an innovative concept for performing probabilistic structural analysis within a parallel processing environment. New materials and structural concepts are being introduced at an unprecedented rate to meet the demands for sustained performance at high temperatures utilizing low-cost manufacturing. As a result, traditional analysis methods are increasingly unable to deliver the analytical insight needed to fulfill the diverse requirements of those demands.

A unique opportunity presently exists for converging three varied fields of study (probabilistic structural analysis, materials science and parallel processing technologies) to combine their latest innovative capabilities into a powerful integrated software package. In an effort to take advantage of this opportunity, Alpha STAR has developed a system that integrates existing specialty codes and methodologies to perform probabilistic simulation of high temperature composite structural response, as shown in Figure 2-1. This system capitalizes on the computational speed of parallel computing hardware and is intended to meet the demands for an analytical engineering tool for design of advanced aerospace structures.

The integrated analysis tool was demonstrated by a detailed verification and benchmark studies of an array of applications. This portion of the study was divided into four categories, each of which require the application of specialized techniques and methodologies. These categories are:

1. Structural Analysis Methods utilizing Laminate Theory and Micromechanics for MMCs
2. Probabilistic Analysis for Structural Reliability and Materials Strength Simulation
3. Domain Decomposition and Parallel Transformation Methods.
4. Dynamic load balancing and optimization to minimize the Computer Processing Unit (CPU) calculations

A separate section of this report (Sections 4.0 through 7.0) is devoted to each of these categories to review the various codes and methodologies viewed as the best suited for application in an integrated package.

Beginning with Section 3.0, a brief overview of the GENOA, integration and commercial packaging of the GENOA finite element-based structural analysis capabilities of the High Temperature Composite Analyzer (HITCAN) are presented. This section will detail some of the unique features of the GENOA (Figure 2-1) code and the innovative techniques employed for the analysis of structures and components made from metal matrix composites.

Section 4.0 depicts the graphical user interface and the executive controller the latter being a high level system which allows the user to interface within the GENOA technical modules to setup data sets, define and allocate filenames, access the GENOA data base and execute jobs. Figure 2-1 illustrates how the executor interacts with all the GENOA modules.

Section 5.0 introduces the partitioning finite element mesh for concurrent computing.

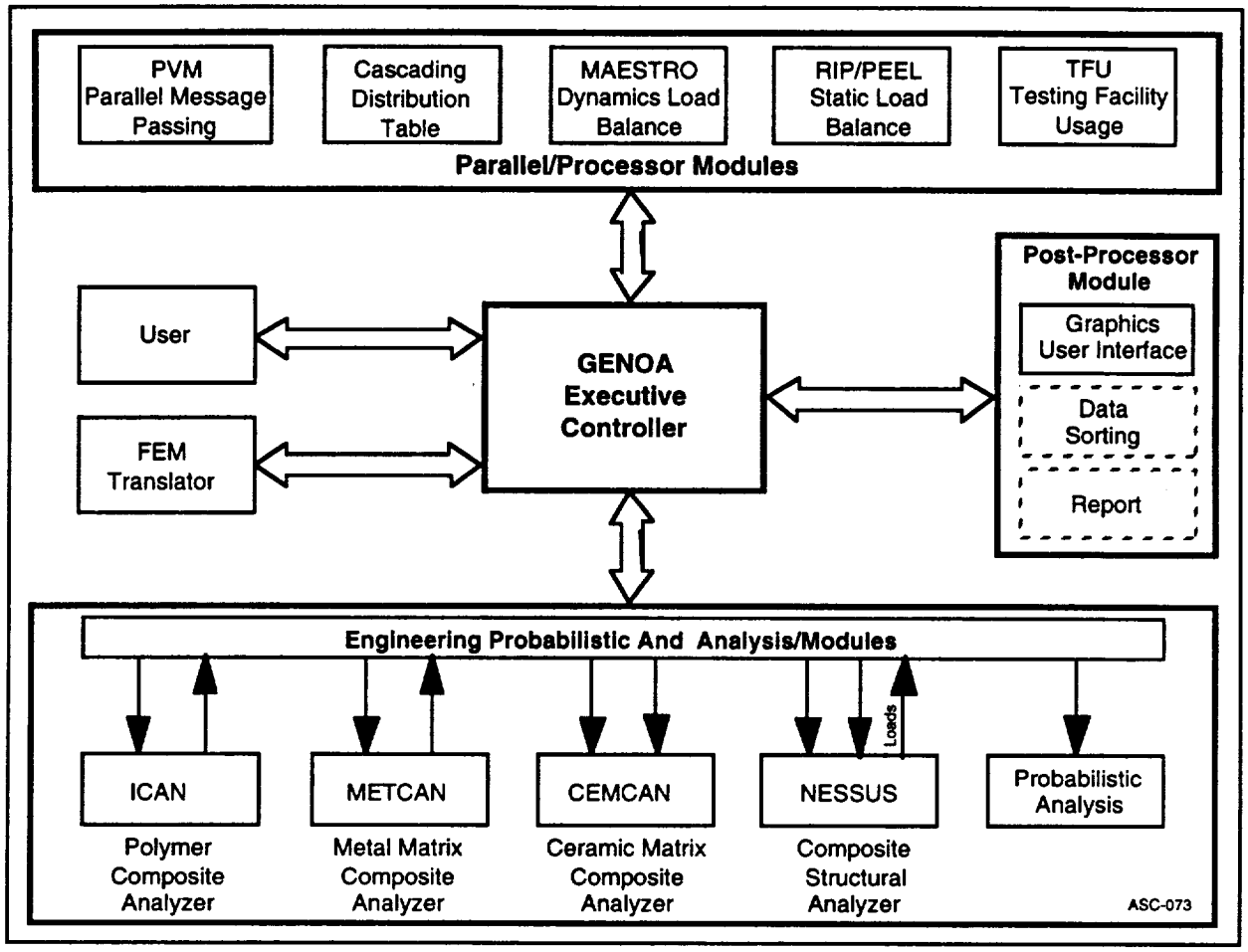


Figure 2-1. Architecture of GENOA Parallel Software System

This section also details the methodology and application of domain decomposition techniques for decoupling sequential finite element matrices for implementation on parallel hardware. A brief description of the RIP and PEEL algorithms is presented .

Section 6.0 introduces the parallel structural analysis methods. It also outlines some innovative concepts for utilizing a hybrid parallelization method which implements the RIP and PEEL codes. This section provides information that is key to the development of GENOA. Following the review of the candidate codes and methodologies for incorporation into the integrated package, it outlines the unique conceptual approach for adapting each component to a concurrent computing environment. The manner in which structural and material response information is processed and passed through an iterative multilevel decomposition procedure to model and ascertain realizations about the various inherent uncertainties is described. An approach for optimizing the use of available processors by means of "Cascading Processor Assignment" specifically for the purpose of multilevel probabilistic simulation is introduced.

As a result of the recent emphasis being placed on the composite stochastics of uncertainty which accompany structural design, the software program, Probabilistic Lifetime Strength of Aerospace Materials via Computational Simulation (PROMISS), and the constituent composite analyzers (METCAN, CEMCAN, ICAN) were incorporated in GENOA as described in Section 7.0.

In Section 8.0, results from several problems are presented to demonstrate integration of High Temperature Composite Analyzer (HITCAN) with PROMISS, RIP and PEEL, and AMF.

Section 9.0 provides the principle investigator's conclusions and recommendations for the continued development of an integrated capability to provide probabilistic composite structural response utilizing massively parallel computer hardware.

In Appendix A, a market analysis is performed to project future demand for the development of a probabilistic composite structural analysis program. Considerable latitude is taken in making assumptions regarding the connectivity and interdependence of the various industry IR&D efforts, government funded programs, initiatives and commercial buying trends. The principle goal of this marketing analysis is to demonstrate that nation wide momentum is gathering and being directed toward research spending in three technological areas: supercomputing, engineering application software development and utilization of new materials.

3.0 *GENOA: An Integrated Modularized Software Package*

This section will present the details of developing an integrated modularized commercial software package called GENOA that has specialized capabilities for analysis of composite structure. Associated innovative concepts for adapting the various analytical techniques to a concurrent multi-processor environment are also presented. These concepts were implemented to optimize the capabilities of both the software and hardware to perform multi-level probabilistic structural analysis for high temperature composites.

Development of this commercial composite software package involved the integration of an ensemble of highly specialized analysis codes and employed a clearly stated and sharply focused modular approach. This modular approach provided the advantages in computer memory management, and modification of the various analytical components. Figure 3-1 broadly illustrates the approach for the integration of existing analytical codes and the iterative procedure for their application. This approach combines the incremental iterative procedures presently employed by NASA-Lewis codes such as HITCAN [3.1] and IPACS [3.2, 3.3, 3.4] with a hybrid method of domain decomposition and an innovative concept for load balancing "cascading processor" assignment.

As an aid in describing GENOA in detail an inventory is presented of the candidate analytical building blocks proposed for constructing a "commercially viable" integrated software package. Table 3-1 provides a listing of the available software selected to fulfill the analytical requirements as well as those needed to package and manage the commercial software. Each of the sources listed are presently portable to a vast array of UNIX-based hardware. The source code is also available in case modifications are necessary. In one or two cases, special arrangements may have to be made for acquisition of these available codes.

Integration of the hierarchical analytical components in an iterative procedure (from the micro level to the macro level) for a heterogeneous and modular software system easily portable to any hardware platform was accomplished by the utilization of a Graphic User Interface (GUI) and an Executive Controller System (ECS). These make up the main drivers for the modules and are a required part of GENOA. The GUI provides the pre- and post-processor visualization capability while the menu driven ECS connects all the modules.

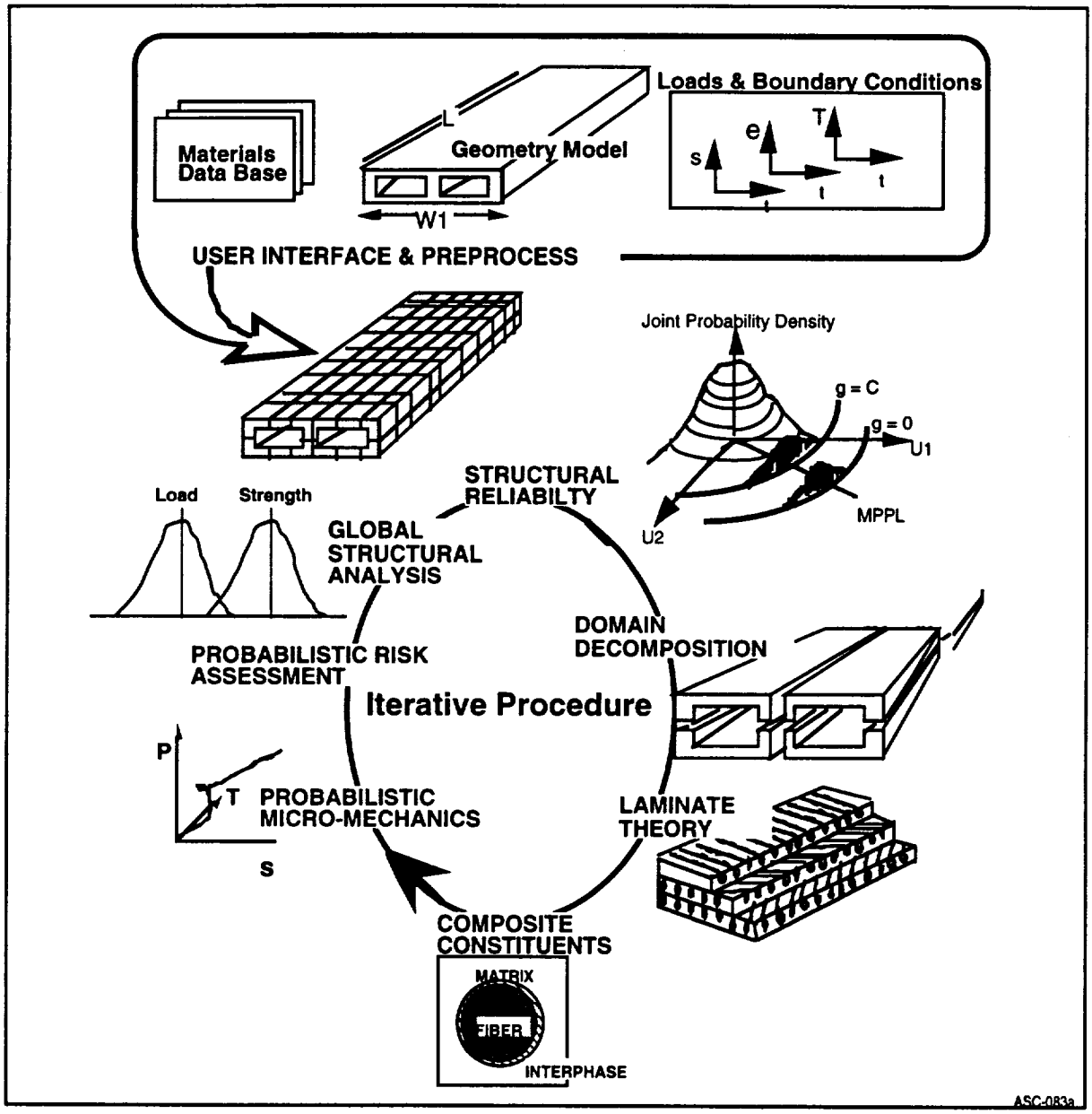


Figure 3-1. GENOA Parallelization Of Probabilistic Structural Analysis For Metal Matrix, Polymer Matrix, And Ceramic Matrix Composites. GENOA is Exploiting Hierarchical Multi-Level Parallism (Macro and Micro Scale)

Table 3-1 . Analytical, Data and Window Management Sources For Developing A Commercially Viable Software Package.

Required Analytical Analysis	Selected Sources
Finite Element Structural Analysis	HITCAN, NESSUS
Micromechanic Analysis	METCAN, CEMCAN, ICAN
Structural Reliability Analysis	FPI
Probabilistic Numerical Simulation for Micromechanics Analysis	PROMISS< PROMISC, IMSL
Domain Decomposition and Mesh Partitioning	RIP and PEEL
Portable User Interface and Windows Manager	X11R-5.0, Executive Control
Geometry Preprocessor	PATRAN
Post-processing and Data Visualization	PHIGS 2D, 3D, X11/Motif
Static and Dynamic Load Balancing	MAESTRO
Translator	NASTRAN, SDRC, COMET

ASC-081

3.1 MODULE 1: EXECUTIVE CONTROLLER SYSTEM (ECS)

The GENOA executive controller system, hereafter referred to as the executor, is basically a high level system which allows the user to interface with the GENOA technical modules to set up datasets, define and allocate filenames, and execute jobs. The executor utilizes a user-friendly screen menu system The ECS has two main functions: provide communication between modules and check the validity of the models in the database. Communication between modules automatically defines and processes the input for each module from other modules, checks the validity of the data and model, and processes the output for the GUI and other modules. The ECS also assists the user in creating and preparing models for each module.

The seven basic functions which constitute the executor are depicted in Figure 3-2 and the functionalities are briefly described in Table 3-2.

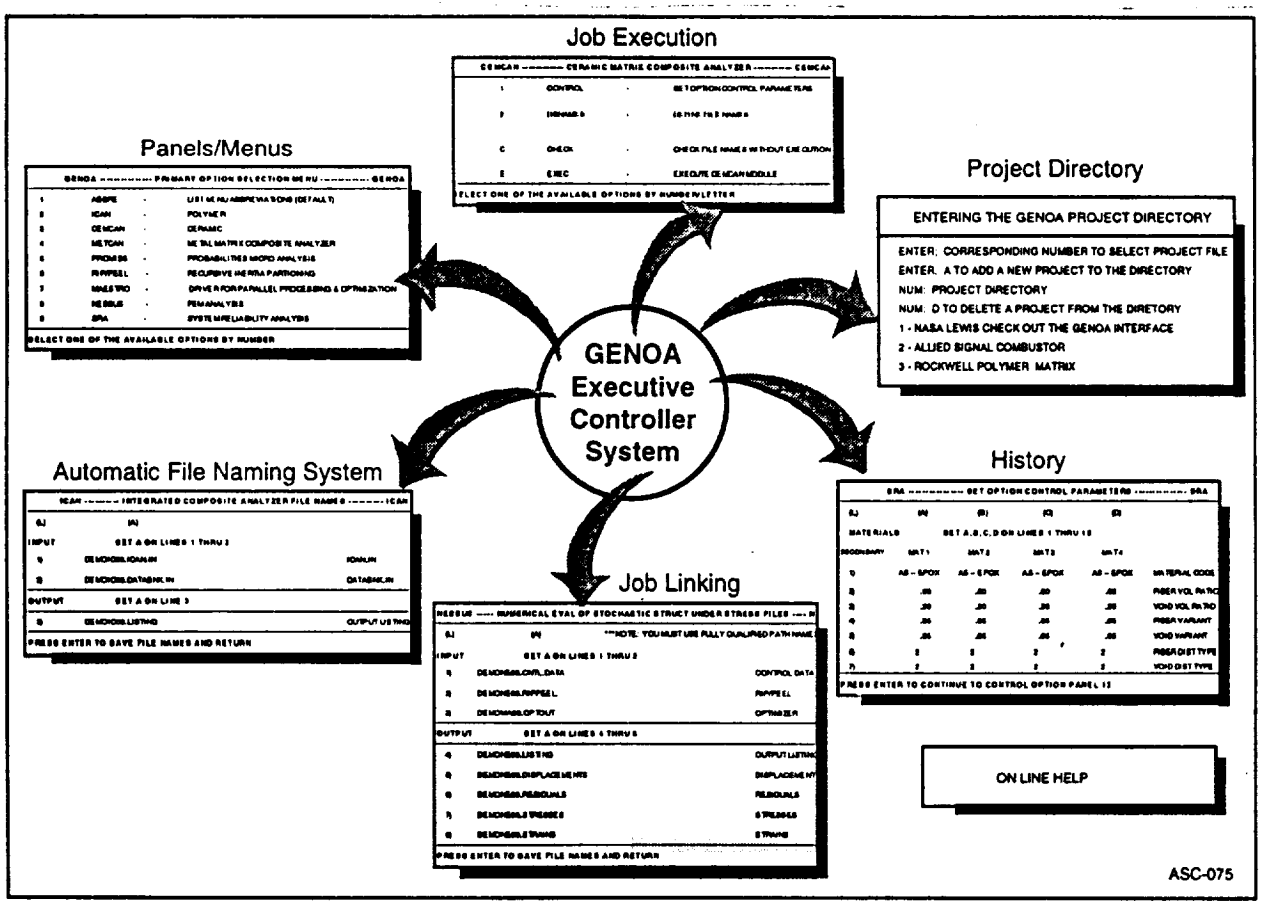


Figure 3-2. GENOA Executive Controller Functions

Table 3-2. GENOA Executive Controller Functions

Executive Controller	Functions
Panels and Menus	User to select from options or to enter data onto screen
Automatic File Naming System	User to select from system default filenames, inserting his own filenames, or using combination of both
Job Execution	User to select Batch, or Interactive mode of job execution
Job Linking	Link several specific series of jobs together
History	User to save his terminal log for future reference
Project Directory	Information created by other users and determine the status of a particular project
Help Menu	On line Manual and help function

Panel and Menus: The words "panel" and "menu" refer to any screen which requires the user to select from options or to enter data. Panel and menu will allow the user to access various GENOA technical modules, by using shortcuts to quickly move from one panel to another, as well as to access the help menus.

Automatic File Naming System: An automatic file naming subsystem has been incorporated into all of the GENOA technical modules. The user has the options of using the system default filenames, inserting his own filename, or using a combination of both. File sequencing is a method whereby the user can slightly change the names of all output datasets to distinguish these from previous runs of the same job without overwriting the old datasets. Use of the file sequencing method requires that filenames adhere to a specified naming scheme.

Job Execution : The executor automatically sets up the system calls and allocates the correct units to appropriate filenames . There are two modes of execution: batch, and interactive. Batch mode allows the user to submit a job to execute on the batch system while continuing working to set up datasets, check files, etc. In the interactive mode, an executing job locks the user out of the system and terminal until interactive execution has been completed. Execution in the interactive mode is recommended only for short duration jobs.

Project Directory: The project directory provides the user with the capability to work on multiple projects, jobs, or tasks and control the data that has been entered for each project. The user may easily switch back and forth between projects, add new projects or delete old ones.

Job Linking: Job linking is a method whereby the user can use special executor commands to tailor a file to link several jobs together in a job submittal instead of running jobs separately. Job linking can be done in either batch or interactive modes but is primarily designed for medium or long duration jobs (overnight runs). Job linking is accomplished by following appropriate linking rules and syntax.

History: History is an executor and UNIX operating system function which allows the user to save all terminal output on a disk file. This feature allows the user to have a record of his session terminal log for future reference.

Help: Help is an interactive user manual helping a user to understand inputs descriptions for a specific simulation

3.2 MODULE 2: USER INTERFACE AND COMMERCIAL PACKAGING OF GENOA

In order to provide the GENOA solvers, geometry modeler and data I/O managers with state-of-the-art commercially viable capabilities, Alpha Star has conducted a task to aid in industry acceptance of GENOA. This task objective was focused on the development of industry standard, fully portable, graphical user interfaces for GENOA's internal code. The user interface for the GENOA integrated software package was developed using Motif, PHIGS and X Windows, Using these tools together, allowed creation of the desired multiple windowed graphical user interface (GUI), employing PHIGS or X11 to produce the three and two dimensional graphics or X to visualize the material analysis data and X Windows and Motif tool-kits to create the other user interface (UI) components. These tools are industry standards, so the user interface is portable to any system that supports the X protocol. Section 4.0 will further explain the tools employed to develop GENOA and the benefits from their use. An example of the X/Motif GUI is shown in Figure 3-3.

Development of the graphical user interface of GENOA with the development tool-kits from Motif/X and PHIGS, and incorporation of the user interface development independent of the data analysis, has made GENOA an application that is easy to use, understandable, portable, configurable and, most of all, a commercially viable material analysis software product.

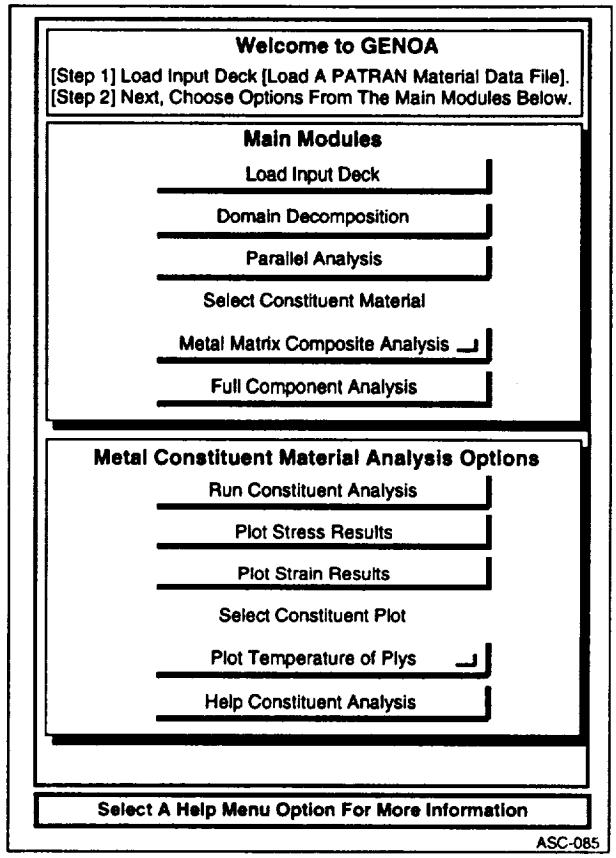


Figure 3-3. Examples Of X/Motif Style GUI Of The Interactive Packaging Of The GENOA Solver, Modeller And Data Manager.

3.3 DOMAIN DECOMPOSITION PARALLELIZATION ROUTINES

The goal of a partitioning algorithm is to take a finite element model composed of nodes and elements and break it into many pieces. The partitioning algorithm further partitions each substructure assigned to a single processor to determine the order of eliminating unknowns within the substructure. The AMF reduces the global memory and the global finite element equations to equations with unknowns only on the boundaries between substructures. Once AMF solves for the unknowns on these interfaces, each processor can work independently to find the solution within its substructure.

The domain decomposition is performed by the RIP and PEEL algorithms [3.5] as shown in Figure 3-4. These algorithms exploit the discretization of the FEM and perform balanced partitioning and natural subdivision of the grid topology to minimize the computational bottleneck. The PEEL algorithm performs natural substructuring of the model based on geometric topology. It identifies the exterior boundary nodes, finds all elements attached to the exterior nodes and peels them off to produce super elements. The RIP algorithm is applied to further subdivide the model. It first identifies the smallest moment of inertia in the long direction, minimizes the boundary region by rotation, and finally balances processor computational loading. Figure 3-4 illustrates the partitioning of a 2D finite element model. At the first level, RIP partitions the model into two separate structures by cutting it at modes 37 through 45. At the second level, RIP partitions each of the previously partitioned structure into a new pair of substructures by cutting the model further at nodes (5, 14, 23, 32) and (50, 59, 68, 77). In Figure 3-4 the partitioning is continued to four levels and a total of 16 substructures (super elements) are created. At each level the list of cut nodes (separator nodes) are listed in a separator tree. The interior nodes of final super elements are listed in a new level added to the last level. The separator tree contains all of the information on the list and order of nodes to be eliminated from the stiffness equations.

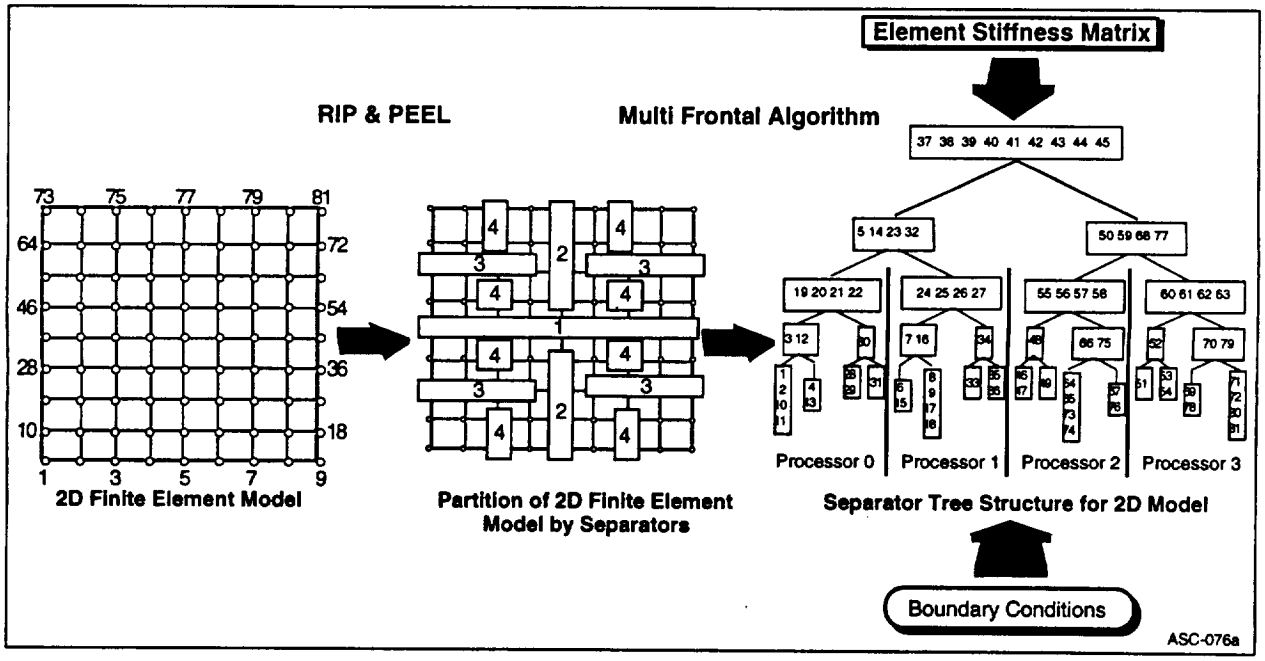


Figure 3-4. Schematical Representation of GENOA/Parallelized FEM Procedure

3.3.1 Module 3: Recursive Inertial Partitioning (RIP) and PEEL

This technique is designed to reduce the communication overhead in sparse matrix decomposition and thus permit efficient runs on message-passing multiprocessors. The number of potential messages is minimized by assigning pivots to processors on the basis of spatial locality in the dissected problems.

This is accomplished using a nested dissection of the underlying problem grid. At each stage of the dissection process, a separator is found that divides the adjacent graph into two disjointed blocks. This procedure is recursively applied to each block until a block is isolated for every processor. The resulting blocks of equations are linearly independent of one another. Nested dissection allows exploitation of large grain parallelism in the sparse matrix decomposition by allocating branches of the elimination tree to multiple problems.

The GENOA parallelization module is developed through the integration of the two unique algorithms known as RIP and PEEL as shown in Table 3-3.

Table 3-3. Domain Decomposition Enable us to Perform Finite Element Mesh Partitioning in Mathematical and Natural Subdomain

PEEL ALGORITHM
Performs substructuring of the model based on geometric topology
Identifies exterior boundary nodes
Finds all attached elements to the exterior nodes and peels them off to produce super elements
RIP ALGORITM
Finds the least moment of inertia in the long direction
Minimizes the boundary region by rotation, and finally balances processors computational loadings
Continues Partitioning until model domain is subdivided into the desired number of sub-domains

The recursive inertial partitioning (RIP) algorithm is a utility routine that addresses the problem of domain decomposition. It has automated this process, essentially making the problem of partitioning transparent to the software engineer. The algorithm works for 2-D and 3-D finite element meshes, partitioning them into subdomains with an equal number of elements with minimal boundary nodes.

The RIP algorithm may be used to break up a finite element problem into p (a power of 2) subdomains; recursively bisecting a structure. The partition allows the element-level computations to be performed concurrently by all processors. It also generates a list of interface nodes for each processor allowing boundary information to be passed between processors during the solution phase.

The second algorithm used in the partitioning utility (PEEL) is designed to find subdomains with a minimal number of nodes on the interfaces of the resulting substructures. It first identifies all peripheral nodes in a finite element mesh. All the peripheral nodes are then removed from the mesh by peeling all the elements that are attached to them. This peeling process is repeated until the mesh reduces to a set of disjointed subdomains. A set of separator nodes can then be obtained by reconstructing the mesh. These separators partition the mesh into a number of subdomains. The partitioning process is then applied to each of the subdomains until the mesh cannot be reduced further. At this point, each subdomain is regular enough so that RIP algorithm can be applied to them to produce balanced domains for parallel computation.

3.3.1.1 The Cascading Processor Micromechanics Assignment

Alpha Star's approach to calculating material properties is to incorporate the macro level parallelism with material-dependent micro scale parallelism. This coarse grain parallelism takes advantage of scalability and exploits the inherent parallelism for the integrated software package.

This inherent parallelism results from the RIP and PEEL algorithms which perform domain substructuring to optimize processor loading and create the binary tree operation of elements assigned to multi-processors. The key to realizing this capability hinges on the development of a processor assignment routine to mitigate the computationally intense repetition associated with probabilistic structural mechanics and probabilistic micromechanics. This routine called "Cascading Processor Assignment," assigns available processors to the cascading levels of parallelism.

Optimal balancing computational processor loads was the primary motivation in determining processor assignment. Unlike the methods presented by Sues, the initial assignment of processors is determined by the granularity of the structural finite element model rather than the number of simulation histories and recursive subroutines. First partitioning the structural model such that each super element has been assigned a processor guarantees that the initiation of the problem solution is performed in a discretely optimal manner.

When dealing specifically with the FE-based probabilistic structural analysis portion of the problem solution, there are several initiation steps to be considered in a processor assignment routine; subdivision of simulation tasks, grouping of additional processors, memory allocation and information communication amongst processors assigned to individual sub-domains. Upon initiation of an analysis, numerous simulations are necessary for each independent variable to be included in the structural response analysis. Having subdivided the available processors based on the structural model, it is then necessary to subdivide the simulation tasks and, in contrast to other approaches for a limited number of processors, assign a task or set of tasks to a previously assigned processors or groups of processors.

Assuming that each element of the structural model has been assigned to processors for simulation tasks, element structural and materials analysis is performed by two approaches: 1) independently calculate the laminate properties for each node and store the data in external storage devices to be used by the superelements as properties of processors, and 2) (TFU) distribute the laminate properties calculations as part of the super-element assembly process. This is done by assigning nodes to processors for super element binary tree operation. This approach takes advantage of optimization to minimize the calculation required for the laminate property. For example, the laminate analysis of METCAN represents a known computational load for a single processor which would reduce the idol time between iterations and convergence evaluations. This procedure is best illustrated in Figures 3-5.

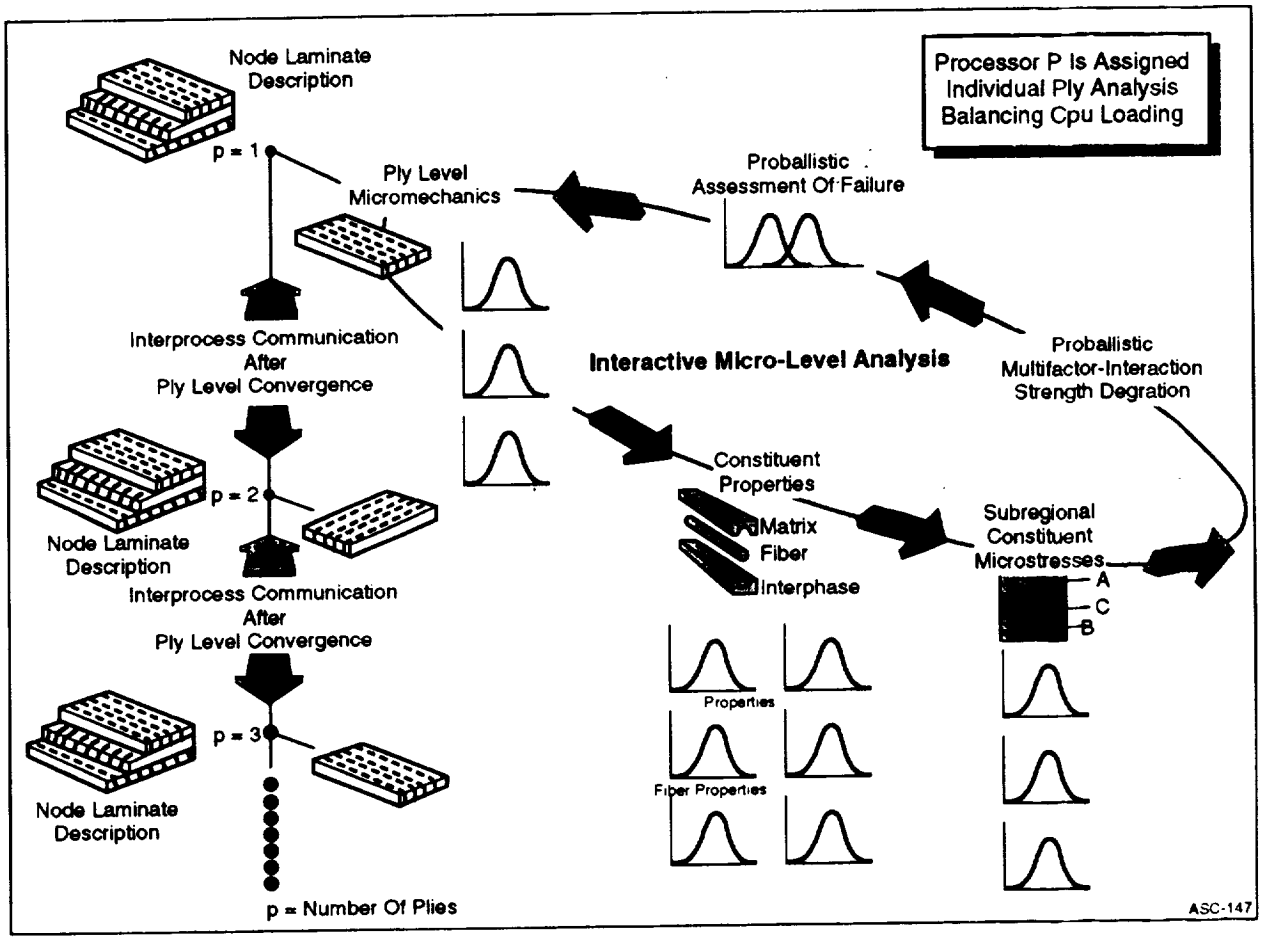


Figure 3-5. Illustration of a Cascading Processor Assignment Routine for PSM

3.3.2 Module 4: Dynamic Loads Balancing MAESTRO

A dynamic load balancing system with a real time parallelization algorithm. MAESTRO provides the power of parallel processing to the GENOA user. MAESTRO functionality is minimization of CPU and wall clock time based on the Optimization of computer/processors resources (memory management, disk space, communication rate, and real and integer operation). MAESTRO complements the Parallel Virtual Machine (PVM) by directing it's activity. Figure 3-6 shows the functionality of the MAESTRO for minimization of CPU and wall clock time, and removal of resource limitations. Hierarchical optimization is achieved by (1) testing computer facility usage, multifactor optimization , and (3) usage of domain decomposition technologies.

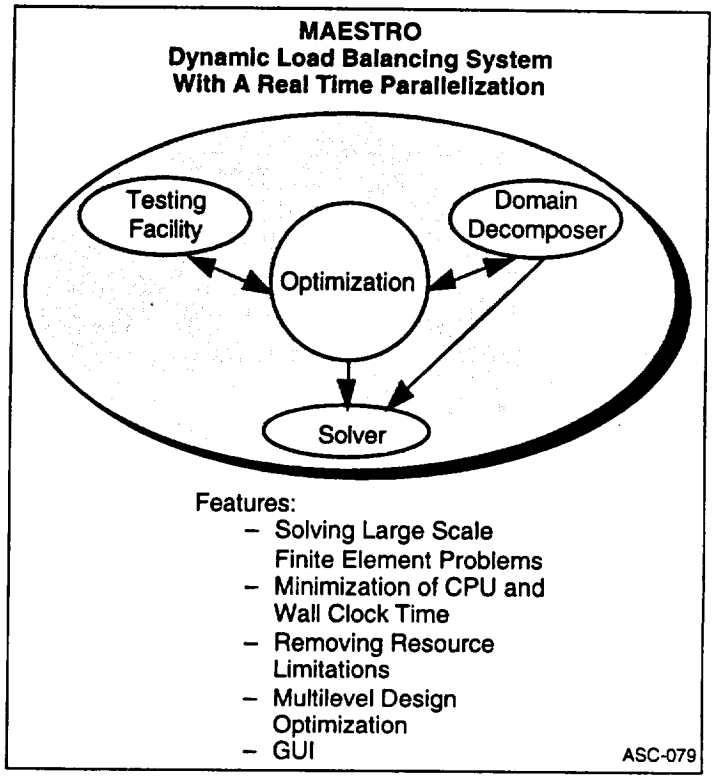


Figure 3-6. MAESTRO Achieves Dynamic Load Balancing Utilizing Testing Facility Usage, Optimization, and Domain Decomposition Technologies

MAESTRO implements Multidisciplinary Design Optimization technique for large scale computing problems. The problem is formulated as a constrained optimization for which minimization of CPU time subject to the parallelism constraints is to be obtained. The dependent and independent design parameters are defined as (1) number of demanded processors, (2) requested processor memory, (3) inter processor communication time, and (4) processor sleeping time.

Concurrent computing environment is based on the effective resources of multiprocessing for the:

1. Estimation and optimization of the requested memory and time interface connection among processors.
2. Automatical analysis and reconfiguration of algorithms for the distributed and massively parallel architecture with irregular binary tree connectivity and hierarchical operation.
3. Execution and generation of building Memory and Time Allocation Table (MAT, TAT) for the process simulation.
4. Implementation of partition of the entire domain into sub domains and prepare the corresponding tree with the desired hierarchical levels.
5. Management of data storage, work load balance, inter processor communications, buffer, and message exchange among the active and non-active processors. Figure 3-7 illustrates the dynamic loading balancing and real time parralization and algorithm. As shown on the upper portion of Figure 3-7 the model is partitioned into static load balance system. The lower chart on Figure 3-7 illustrates processor CPU minimization. As shown the best combination of processors and model divisions can be achieved interactively by the integer optimization algorithm.

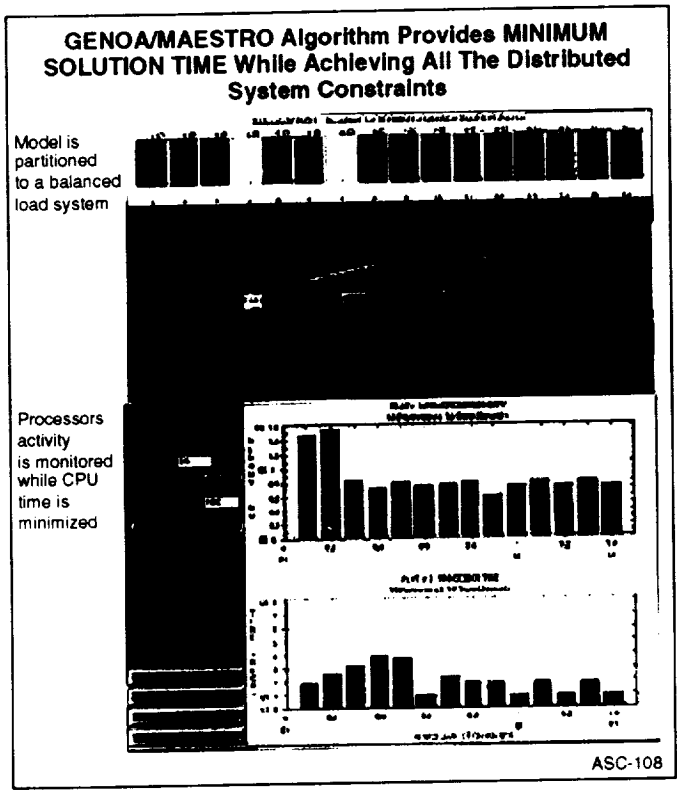


Figure 3-7. GENOA/MAESTRO Algorithm Provides Minimum Solution Time While Achieving all the Distributed System Constraints

3.3.2.1 Testing Facility Usage (TFU)

The development of the TFU for the MAESTRO optimization module has resulted in accurate precalculated task distribution and real running time estimation. TFU complements the PVM message passing system by determining load size and dynamic availability of processors. This system is composed of two parts. The first part measures the interprocessor communication time between selected processors, and the second part measures the CPU time of certain arithmetic operations on selected processors. These techniques are mainly used in distributed workstation computing with unequal processor speed. In comparison, PVM message passing only spawns using round-robin scheduling for equally sized tasks. TFU supports heterogeneity at the application, machine, and network levels. In other words, TFU instantaneously supports information of available processors preparing the spawning of allocated tasks on the desired processors.

TFU generates the required information in the optimization process, with the selection of a set of processor resources (real and virtual). The inherent connection to the separator tree operations is control and management of the best combination of heterogeneous available resources such as machine type, network level, number of processors and available memory to exploit the desired architecture suited for the operations on the separator tree

Communication : This segment measures the time to transfer a certain amount of bytes between two selected (peer to peer) real or virtual processors. The PVM libraries are used to initiate these processes between the communication partners. The UNIX system calls are used to retrieve the elapsed time of transferred packets between sending and receiving peers. The packets are sent a few times and averaged to increase the precision.

CPU Testing : This segment measures the CPU time and elapsed time usage of a selected CPU for a certain amount of arithmetic operation. The PVM libraries are used to spawn the process on the selected processor. The UNIX system calls are used to retrieve the CPU time and elapsed time at different points of this testing. This module supports the testing of the operations (+ , *) for the data types *integer* and *float*. In addition, the testing for (< , >) is supported. The availability of a processor is expressed by the ratio of wall clock to CPU time. This function can assist the MAESTRO dynamic load balancing system to redistribute the tasks among the processors.

3.3.2.2 Message Passing Paradigm

The message passing paradigm is the conventional PVM (Parallel Virtual Machine), software that enables a collection of heterogeneous computers to be used as a coherent and flexible concurrent computational resources. The PVM toolkit, developed by Oak Ridge National Laboratory (ORNL), is one of the more popular portable toolkits The PVM library is a proven solution for portable parallel

programming and provides the function, control, and parallel efficiency needed to effectively implement real time parallelization. The selection of PVM was based on portability, and standardization all classes of parallel architecture. PVM supports a wide range of facilities including the ability to configure the set of participating hosts dynamically, to debug selected component instances, to position specific processes, and to execute multiple process that make up an application using several different control structures. The XPVM front-end is designed to enable convenient access to the PVM facilities using a graphical interface, and is described in this section. Figure 3-8 shows a sample of the XPVM session using three real and five virtual workstation processors as shown. The XPVM interface permits the specification of an object module and the number of instances that are to be initiated enabling the individual initiation of separate programs, thereby avoiding the need for a user written driver program.

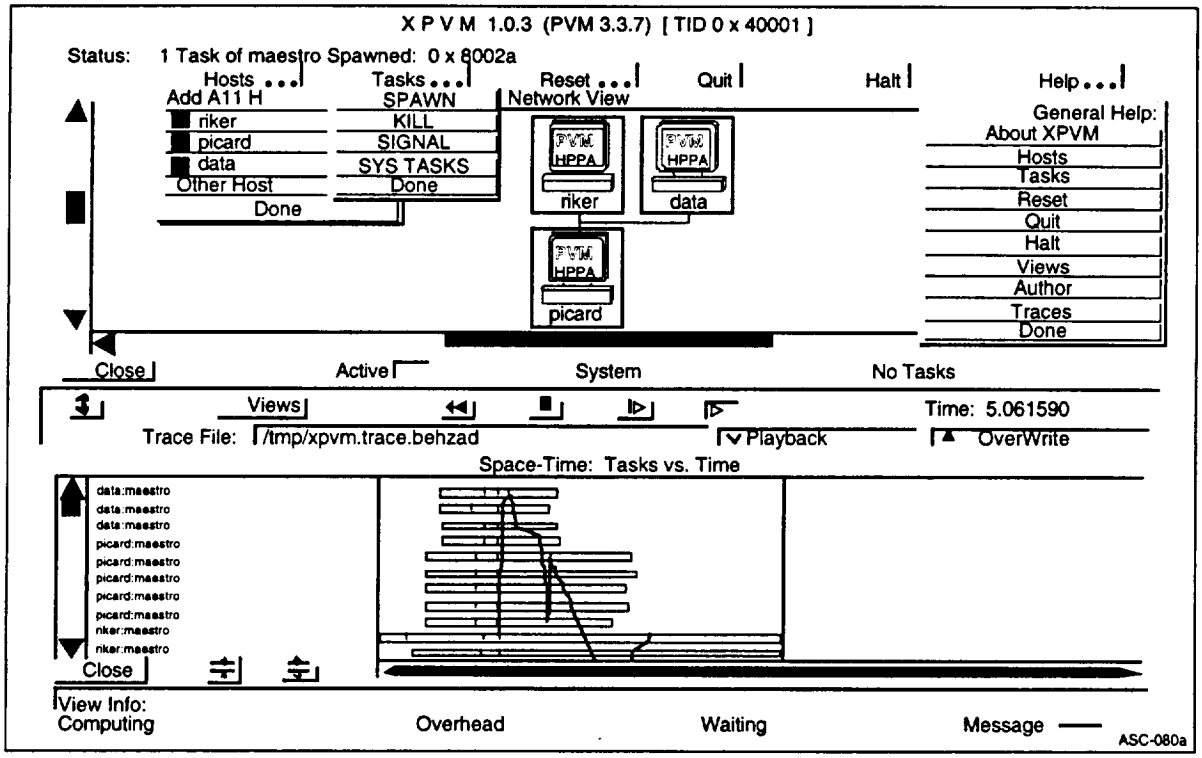


Figure 3-8. Sample of XPVM Session of Three Real and Five Virtual Processors.

3.3.2.3 Task Allocation Table (TAT)

TAT responsibility is the management of the instantaneous task loading assignment of operations. For multifrontal algorithm and micromechanics cascading assignments, TAT distributes processors for super-element operations such as climb, and descent. As shown in Table 3-4, TAT assigns to each processor the following assignments: 1) PARASS which partially assembles any group of elements 2) COND which applies external loads and boundary conditions, and performs the condensation process, 3) ASSEMBLE which assembles a pair of super elements and updates the super element-to-global mapping vector, 4) READ, 5) WRITE 6) SOLVE, 7) TAKE, and 8) (.....) (processor in sleeping mode).

3.3.2.4 Multi Factor Optimization

For large scale computing problems, the Multilevel Design Optimization (MDO) technique is implemented. The parallelization of the solution is effectively optimized to reduce CPU time and memory limitations and optimize the TAT of Table 3-4.

Multi Objective optimization of inter-processors' task distribution is based on the numerical generation and analysis of the functional time-memory to minimize the goal time function. Based on the available memory size, estimation procedures calculate the memory needed to perform the necessary operation, and minimization of the processor sleeping time by performing synchronization (eliminate symbol ..... in Table 3-4).

Table 3-4. The Processor Task Loading Assignment Generated by TAT of 8 Super-Elements Between 8 Processors.

Proc No. 1	Proc. No. 2	Proc No. 3	Proc No. 4	Proc No. 5	Proc. No. 6	Proc No. 7	Proc No. 8
			Beginning Of The Climb				
MP&Pr 8	MP&Pr 9	MP&Pr 10	MP&Pr 11	MP&Pr 12	MP&Pr 13	MP&Pr 14	MP&Pr 15
Condn 8	Condn 9	Condn 10	Condn 11	Condn 12	Condn 13	Condn 14	Condn 15
Store 8	Store 9	Store 10	Store 11	Store 12	Store 13	Store 14	Store 15
Tk M 8	.....	.....	.....	.....	.....	.....	.....
.....	.....	Tk M 10	.....	.....	.....	.....	.....
.....	.....	.....	.....	Tk M 12	.....	.....	.....
.....	.....	.....	.....	.....	.....	Tk M 14	.....
RD 2 S 9	WT 1 S 9	.....	.....	.....	.....	.....	.....
.....	.....	RD 4 S11	WT 3 S11	.....	.....	.....	.....
.....	.....	.....	.....	RD 6 S13	WT 5 S13	.....	.....
.....	.....	.....	.....	.....	.....	RD 8 S15	WT 7 S15
Assem 4	.....	Assem 5	.....	Assem 6	.....	Assem 7	.....
Condn 4	.....	Condn 5	.....	Condn 6	.....	Condn 7	.....
Store 4	.....	Store 5	.....	Store 6	.....	Store 7	.....
Tk M 4	.....	.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	Tk M 6	.....	.....	.....
.....	.....	WT 1 S 5	.....	.....	.....	.....	.....
.....	.....	.....	.....	RD 7 S 7	.....	WT 5 S 7	.....
Assem 2	.....	.....	.....	Assem 3	.....	.....	.....
Condn 2	.....	.....	.....	Condn 3	.....	.....	.....
Store 2	.....	.....	.....	Store 3	.....	.....	.....
Tk M 2	.....	.....	.....	.....	.....	.....	.....
RD 5 S 3	.....	.....	.....	WT 1 S 3	.....	.....	.....
Assem 1	.....	.....	.....	.....	.....	.....	.....
Condn 1	.....	.....	.....	.....	.....	.....	.....
Store 1	.....	.....	.....	.....	.....	.....	.....
			END OF THE CLIMBING				
Solv TOP	.....	.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	RD 1 S 1	.....	.....	.....
Tk Y 1	.....	.....	.....	Tk Y 1	.....	.....	.....
Solvx 2	.....	.....	.....	Solvx 3	.....	.....	.....
				Level 2			
StorY 2	.....	.....	.....	StorY 3	.....	.....	.....
WT 3 S 2	.....	RD 1 S 2	.....	WT 7 S 3	.....	RD 5 S 3	.....
Tk Y 2	.....	Tk Y 2	.....	Tk Y 3	.....	Tk Y 3	.....
.....	.....	Solvx 5	.....	Solvx 6	.....	Solvx 7	.....
StorY 4	.....	StorY 5	.....	StorY 6	.....	StorY 7	.....
				Level 3			
WT 2 S 4	RD 1 S 4	WT 4 S 5	RD 3 S 5	WT 6 S 6	RD 5 S 6	WT 8 S 7	RD 7 S
Tk Y 4	Tk Y 4	Tk Y 5	Tk Y 5	Tk Y 6	Tk Y 6	Tk Y 7	Tk Y 7
Solvx 8	Solvx 9	Solvx 10	Solvx 11	Solvx 12	Solvx 13	Solvx 14	Solvx 15
				Level 4			

ASC-102

Memory constraint is satisfied by initial replacements of tasks between the processors. The number of design variables for global optimization is based on the following formulation:

$$N_{dv} = 2 * N_{sg} + 2 \qquad (1)$$

where N_{dv} is the effective number of design variables, and N_{sg} is the number of separators after domain decomposition.

The convergence of the time memory functional is based on ; 1) the reduction of the number of the inter processors communications, and 2) reduction of messages sizes by simple replacement of tasks between processors.

Figure 3-9 demonstrates the stabilization of the convergence of the functional time memory function from different initial arbitrary positions. As shown, the stabilization of the optimization convergence for Inter Processor Communication (IPC) will result in reduction of total CPU time. The three-dimensional plot describes the surface of the time-memory function presented as the maximum value of processor loading time (Y axis), and local iteration number (X axis), versus the number of initial arbitrary points of local minimum.

Figure 3-10 demonstrates the NASA/LeRC SSME turbo blade model, CPU, and wall clock time minimization between 16 processors and 16 super elements before and after optimiztion.

3.3.3 Module 5: Alpha Star Multifrontal (AMF) Algorithm

A distributed multifrontal (DMF) sparse matrix decomposition algorithm for a parallel processing environment uses a nested dissection ordering and multifrontal distribution of the matrix to minimize inter-processor data dependencies and overcome the communication bottleneck. The number of potential messages is minimized by assigning pivots to processors on the basis of spatial locality in the dissected problem. This is accomplished by using a nested dissection [3.2] of the underlying problem grid.

The finite element solve operations in the binary tree are performed by an AMF algorithm. In this algorithm, the global system of equations are broken down to smaller systems of equations resulting in reduced memory for stiffness matrices .

A distributed multifrontal (DMF) algorithm for sparse Gaussian elimination on parallel computers was presented by Nour-omid, et. al. This method uses the nested dissection reordering heuristic to extract separator from the graph of the matrix, thereby partitioning the matrix into disjoint blocks that can be allocated to the processors. Symbolic decomposition of the result is shown to be completely independent. The number of messages exchanged during the sparse matrix factorization is limited by

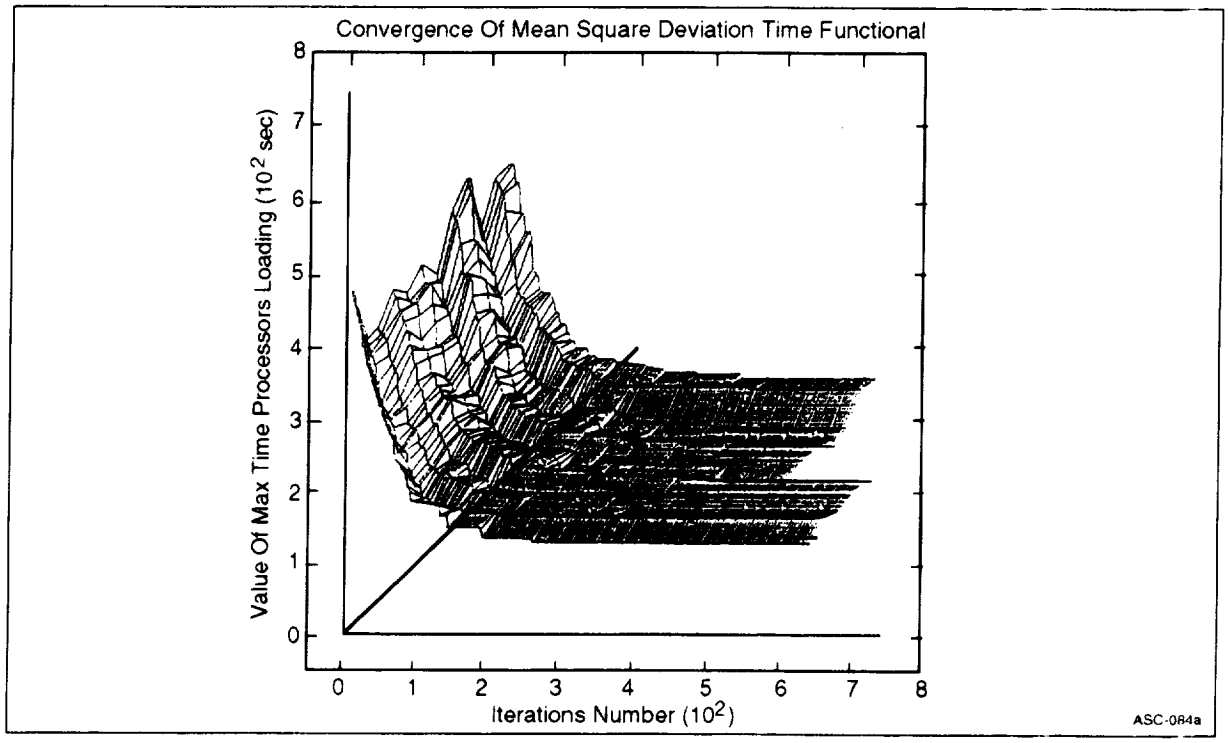


Figure 3-9. Stabilization Of The GENOA/RPA Optimization Process

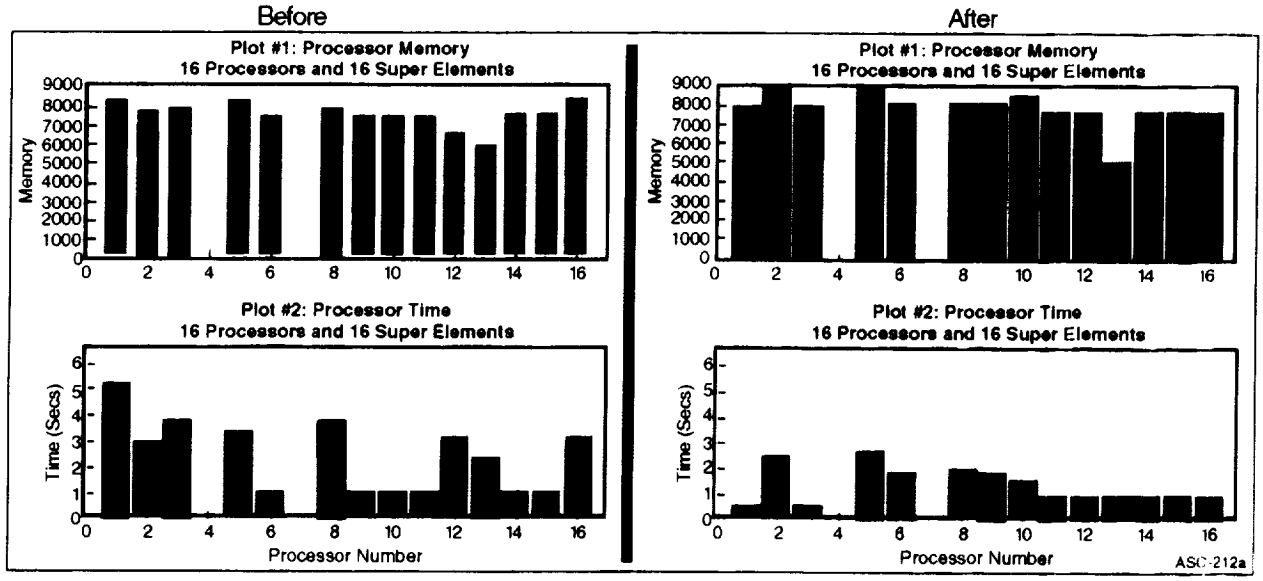


Figure 3-10. X11/Motif Animation Of SSME Turbo Blade Model CPU And Wall Clock Tim Minimization Before And After Optimization Between 16 Processors And 16 Super Elements

the function of the length of the separators. The DMF sparse solver achieves parallel efficiencies of over 70 percent. AMF was developed to remove the limitations of DMF. AMF can operate on all types of elements, mixture of elements, and even unsymmetric binary trees. Both DMF and AMF are based on multifrontal algorithms consisting of groups of operations at different levels of the of the binary tree as shown in Table 3-5.

Table 3-5. The Alpha Star Multifrontal (AMF) Algorithm Reduces Time and Memory Required in the Finite Element Analysis

	I. CHILD LEVEL
STEP 1:	Elements are grouped into several super elements according to the domain decomposition performed by RIP & PEEL
STEP 2:	Assembly of the elements of each super element using local stiffeness [K] matrix
STEP 3:	Condensation of the stiffness matrix associated with each super element
	II. CLIMBING LEVELS
STEP 4:	New super elements are created ; Assemble every pair of condensed super elements according to binary tree
STEP 5:	Combining a pair of super elements into a new super element
STEP 6:	Condensation of the new super element at each level of binary tree; repeat step 3 to decompose, and then step 4 to assemble at next level of binary tree
	Repeat steps 3, and 4 till top level is reached
	III. TOP LEVEL
STEP 7:	Only one new super element is created
	All of the nodes at this level are interior and fully summed
STEP 8:	Solving for the unknowns at these nodes
STEP 9:	Sending the solutions to every processor
	IV. DESCENDING LEVELS
STEP 10:	The binary tree is descended in the reverse order of climbing
STEP 11:	Solving for the interior nodes
	Use the solution in STEP 8 and back substitution to solve for the eliminated unknowns at the level below top level of binary tree
STEP 12	Use the solution of higher levels and back substitution to solve for eliminated unknowns at lower levels binary tree
STEP 13	Sending the solutions to the processors acting on the corresponding children

3.4 STRUCTURAL ANALYSIS METHODS FOR METAL MATRIX COMPOSITES

HITCAN is a general purpose code for predicting the global structural and local stress-strain responses of angleplied metal matrix composite structures. It is a compilation of two principle codes known as MHOST/NESSUS, and METCAN that perform predictive analysis at both the constituent (fiber,matrix and interphase region of MMC) and structural levels. Key to the unique capabilities of the HITCAN code is the computational procedures for utilizing a multifactor-interaction material behavior model within an incremental iterative nonlinear analysis.

The MHOST program, developed by MARC Analysis Research Corporation [3.6] for NASA Lewis Research Center, is a 3-D inelastic finite element code. Although the MHOST program was designed to perform nonlinear analysis of turbine engine hot section components, it has also shown great potential for performing large-scale structural analysis as an integral part of the IPACS program. The METCAN program embodies a unique set of micromechanics equations and constitutive relationships for analysis of angleplied metal matrix composites. This program has been implemented in numerous studies as a stand-alone analysis tool demonstrating an array of capabilities from materials process modeling to high temperature creep analysis.

The development of several programs through the combined efforts of Dr. Chamis and co-workers, marks the bridging of the gap between purely global structural response and the highly localized methods of modeling materials behavior with micromechanics. Specific to the analysis of metal matrix composite structures, Chamis and co-workers have successfully developed and demonstrated the capabilities of the finite element-based computer program HITCAN (High Temperature Composite Analyzer). As in Like many of the integrated analysis tools being developed under the direction of Dr. Chamis (ICAN, PICAN and IPACS), an incremental iterative analysis procedure facilitates the bridging of finite element and micromechanics methods. Figure 3-11 illustrates this procedure as it is applied to HITCAN. HITCAN represents the synergistic integration of various specialized methods produced at NASA-Lewis over the last two decades.

In order to further implement and enhance the capabilities of the HITCAN program, Alpha Star has used the program's unique computational structure as a backbone for GENOA development. In this regard the HITCAN code has two important aspects: 1) it is self-contained (independent of commercial codes offering versatility for porting to a varied array of computing architectures, and 2) it is well documented facilitating adaption and integration of additional capabilities.

3.4.1 Modularization Of HITCAN: A Source For Baseline Analytical Capabilities

In concurrence with establishing a relationship between the local (micromechanics) and global models (FEM), and performing iterations between them, a modular HITCAN program is developed as the Baseline Analytical driver of the GENOA to achieve convergence with standard iterative solution method by performing joblink between the GENOA modules such as METCAN, NESSUS, and CEMCAN. HITCAN input and output parameter panel is illustrated in Table 3-6.

HITCAN is modularized to perform iterative local, and global modeling while achieving computational efficencyby using dynamic allocation of memory. The modularization is directed to: 1) maximize the size of the iterative problem that can be solved, 2) easily utilize and adopt other material simulators such as CEMCAN, ICAN. etc, 3) implement mixed material models within the same FE model, 4) integrate the NESSUS parallel FEM solver into HITCAN to substitute for MHOST and 5), to ease setting up the control information to perform a HITCAN simulation.

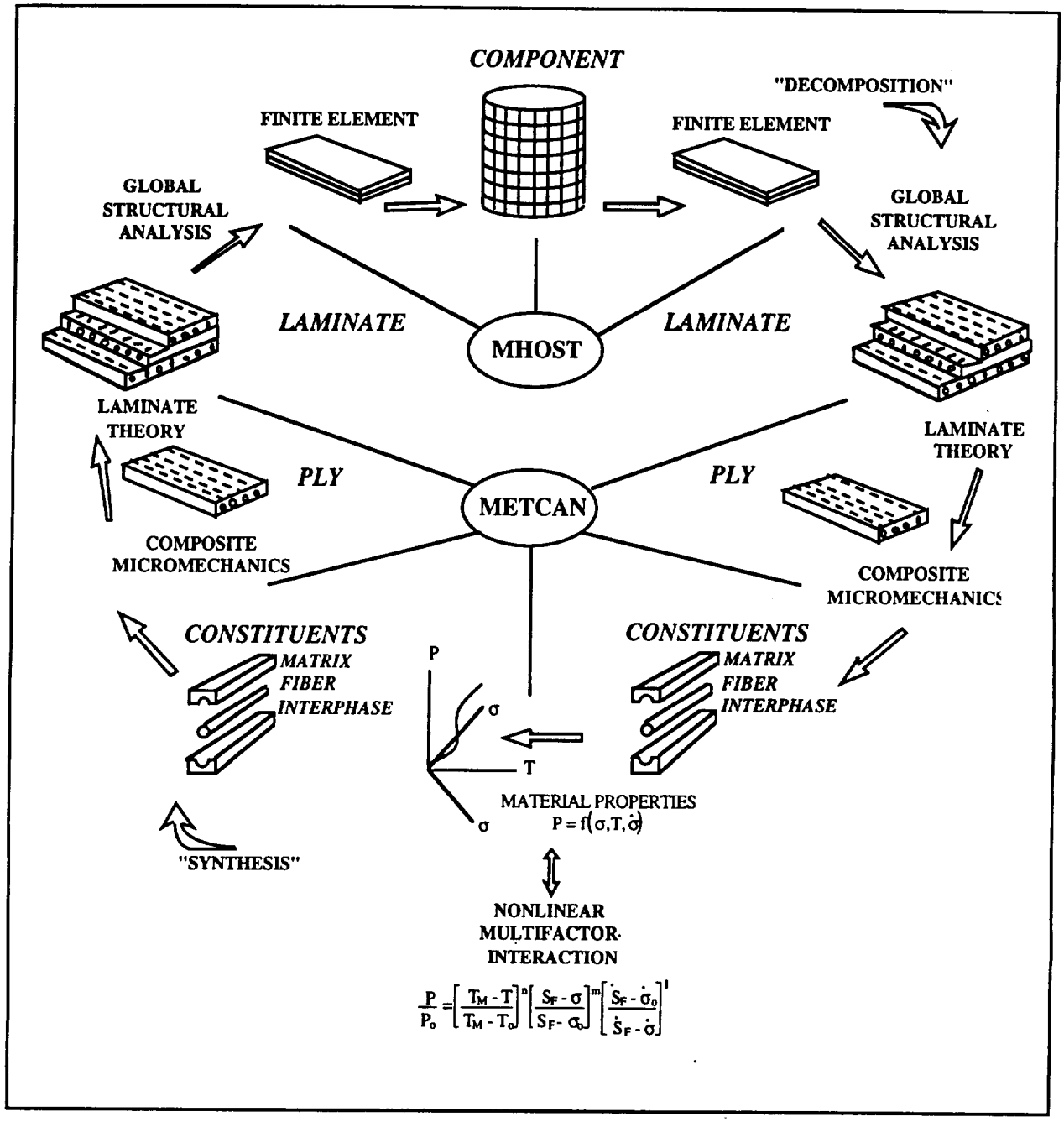


Figure 3-11. Illustration of the Computational Iterative Procedure Utilized by HITCAN to Perform Composite Micromechanics Within a Finite Element-Based Analysis.

In order to perform a modular integration with the HITCAN program, a three stage approach is taken . The first stage copies the deck initialization data from the NESSUS panel which performs parallel FEM analysis. The second stage copies the initialization deck from the METCAN panel and initiates the actual parallel analysis using METCAN. The third stage sets up the HITCAN control

Table 3-6. High Temperature Composite Analyzer

HITCAN---High Temperature Composite Analyzer ---HITCAN

This code performs integration and iteration between finite element and material constituents

1	METCNTL	– Set METCAN Option Control Parameters
2	NESCNTL	– Set NESSUS Option Control Parameters
3	HITCNTL	– Set HITCAN Option Control Parameters
4	DSNAMES	– Define METCAN File Names
5	DSNAMES	– Define NESSUS File Names
6	DSNAMES	– Define HITCAN File Names
C	CHECK	– Check File Names Without Execution
E	EXEC	– Executive HITCAN Module

Select One Of The Available Options By Number/Letter

ASC-203

file with the necessary information to begin the itterative analysis The steps taken in this modularization are as follow:

STAGE 1 NESSUS Initialization: by the Executive controller

1. Start the Peel algorithm and partition the model into as many substructures as there are available processors.
2. Read the computer facility information from MAESTRO, e.g. the number of processors, rate of communication , and Partition data from RIP and PEEL algorithms
3. Read the FE model from PATRAN or GENOA FEM preprocess
4. Setup loads, boundary conditions and temperatures as physical parameters.
5. Create an initial NESSUS deck and Send the initializing parameters to HICAN control deck.

STAGE 2 METCAN Initialization:

1. Make a system call to METCAN input deck (automatically performed by the Executive controller).
2. Execute a system call to initialize the METCAN (the number of mesh nodes, the number of parallel processors, and the distribution of METCAN runs among the processors).
3. Map the METCAN data for use in Stage 3.

STAGE 3 HITCAN Initialization

1. Initiate the HITCAN program (using input from STAGES 1, and 2) for assembly of an iterative job.
2. Perform parallel METCAN analysis and generate laminate data for NESUSS FEM.
3. Use parallel processing to obtain a NESSUS FEM solution for the stress and strains.
4. BASED ON HITCAN executive controll data continue NESSUS/METCAN iteration, and check for convergence.
5. Return to HITCAN to perform response computations and post processing.

3.4.2 NESSUS: 3-D Inelastic Finite Element Analysis

The NESSUS program implements an innovative mixed finite element procedure which is intended to improve accuracy and diversity in modeling capability [3.7]. The parallel version of NESSUS is integrated with AMF (version-6 that allocates processor memories and disk space . This allows the finite element user to run large models beyond the limitation of local processors to make static analysis

of three types of inelastic constitutive models: 1) simplified plasticity, 2) conventional plasticity, and 3) an advanced viscoplasticity.

The AMF allows the user to simulate combination of elements within the same finite element analysis. This enhancement allows the use of beam, and shell elements as the mixed element capability.

Dynamic Allocation Memory: NESSUS program originally written in FORTRAN uses the static allocation of memory that is hard coded before compilion time. This means code recompiling is necessary for different sizes of input models. Instead GENOA uses dynamic memory allocation supported in " C" language and passes the allocated memory to the other parts of the code written in FORTRAN. The optimization module of MAESTRO dynamically calculates the amount of memory needed based on the specifications of the model, number of processors, and their capabilities. If there is not enough memory for some processor, the whole process can be stopped and run again for a new configuration.

The iterative finite element solution procedure of HITCAN-NESSUS is illustrated by the flow diagram shown in Figure 3-12. Matrices [K] and [F] are the conventional finite element stiffness matrix and load vector, respectively. Other conventional finite element matrices appearing in this flow diagram are [B], [D], and [Q], the strain-displacement matrix, the stress-strain matrix and the diagonalized inner product of shape functions respectively. All quantities involved in the computation are evaluated as nodal vectors. Values for stresses, strains and displacements are obtained at nodes by the nodal strain recovery calculation and the residual load correction shown in the flow diagram.

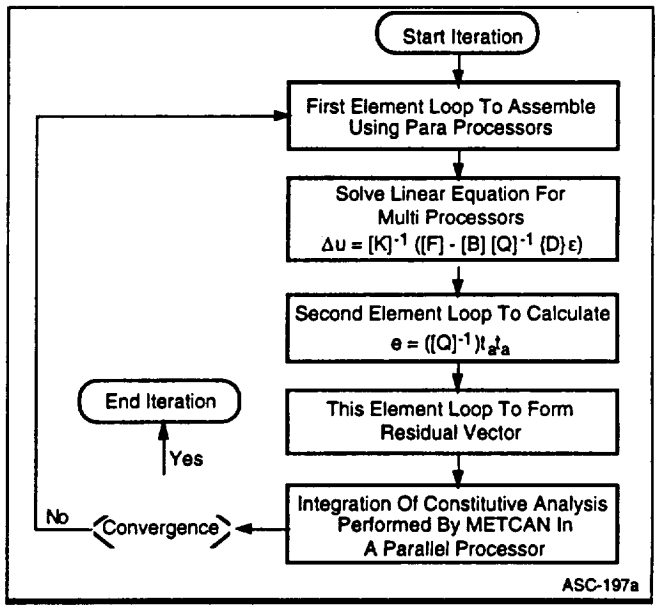


Figure 3-12. Iterative Method Utilized By HITCAN To Perform Nonlinear Structural Analysis.

Iterative solutions are utilized by NESSUS even when solving for a purely elastic response. An iterative solution of elastic response, compared to the standard displacement method, produces improved stress results.. In path-dependent inelastic calculations, such improved response increases the accuracy of analyzing the behavior of a composite structure at high temperatures.

3.4.3 METCAN: Bridging The Micro-to-Macro Gap

At the heart of HITCAN's ability to perform high temperature composite structural analysis is the use of innovative computational methods embodied in the METCAN code [3.8, 3.9]. The formulation and development of METCAN spans many years, going back to the Itegrated Composite Analyzer (ICAN) [3.10]. The ICAN program gave rise to the multilevel iterative procedures for decomposing the coupled response of angleplied layers of conventional composite constituents to provide a constitutive-based predictive model for a composite laminate. The extension of these procedures to the nonlinear behavior of MMCs was initiated by Hopkins and Chamis [3.11]. Their efforts derived a unique set of micromechanics equations which would provide basic mechanics of materials formulation for MMC constituents while retaining the essential square array unit cell model common to the ICAN methodology (Figure 3-13).

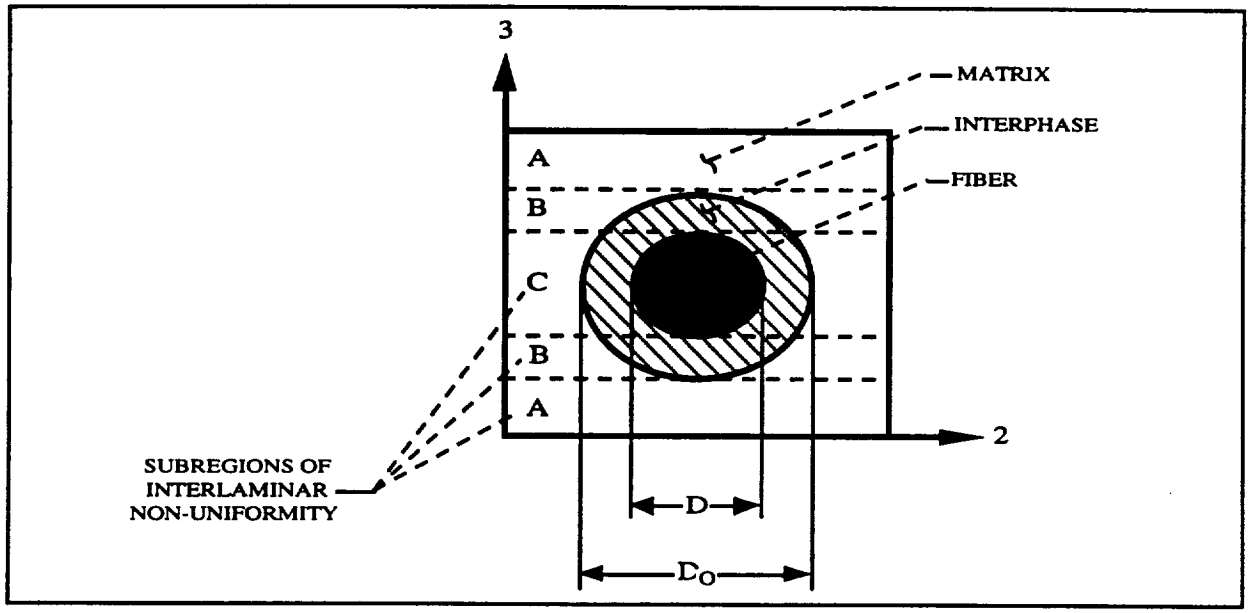


Figure 3-13. Schematic Diagram Illustrating The Square Array Unit Cell Utilized By METCAN As A Micromechanics Model

The thermal and mechanical response of a component or structure fabricated from continuous fiber reinforced metal matrix composites is influenced and ultimately limited by the constituent materials. For an angleplied laminate, the behavior of the constituents becomes very complex and highly coupled. The approach taken in the development of ICAN (and later METCAN), utilizes three levels of analysis to decouple and subdivide the material system into individual components for comprehensive evaluation. As illustrated in Figure 3-11, the analysis of the composite is performed by: 1) loading of the individual plies at the macro level by virtue of the load type and laminate description; 2) decoupling of the ply response through the use of the laminate theory; and 3) modeling of the micromechanical ply properties by subdividing the square unit cell model into constituent material subregions. Once the composite system is completely decomposed and the respective thermal and mechanical loads have been distributed as stresses via micromechanical relations, the nonlinear response of the constituent properties and strengths is determined through the use of a multifactor-interaction equation. The basic form of this equation is given as:

$$\frac{P}{P_o} = \left[\frac{T_F - T}{T_F - T_o}\right]^n \left[\frac{S_F - \sigma}{S_F - \sigma_o}\right]^m \left[\frac{\dot{S}_F - \dot{\sigma}}{\dot{S}_F - \dot{\sigma}_o}\right]^l \left[\frac{\dot{T}_F - \dot{T}}{\dot{T}_F - \dot{T}_o}\right]^k \left[\frac{R_F - R}{R_F - R_o}\right]^p \left[\frac{N_{MF} - N_M}{N_{MF} - N_{Mo}}\right]^q \left[\frac{N_{TF} - N_T}{N_{TF} - N_{To}}\right]^r \left[\frac{t_F - t}{t_F - t_o}\right]^s \quad (1)$$

with the notations: P - property; T - temperature; S - Strength; R - metallugical reaction; N - number of cycles; t - time; overdot - rate; subscript o - reference; subscripts F - final; subscript M - mechanical; subscript T - thermal; and n,m,l,k,p,q,r,s the respective thermoviscoplastic exponents. Through the application of this relationship (1), METCAN is able to model the time-temperature-stress dependence of each constituent's thermal and mechanical properties at any point in the load history. This single constitutive relationship embodies the coupling of a diverse range of linear and nonlinear behavior. It also allows the user to implement expert opinion as to the degradation behavior of an individual effect

by means of thermoviscoplastic exponents. Figure 3-14 illustrates how these exponents are utilized to model the temperature dependence of the thermal and mechanical properties.

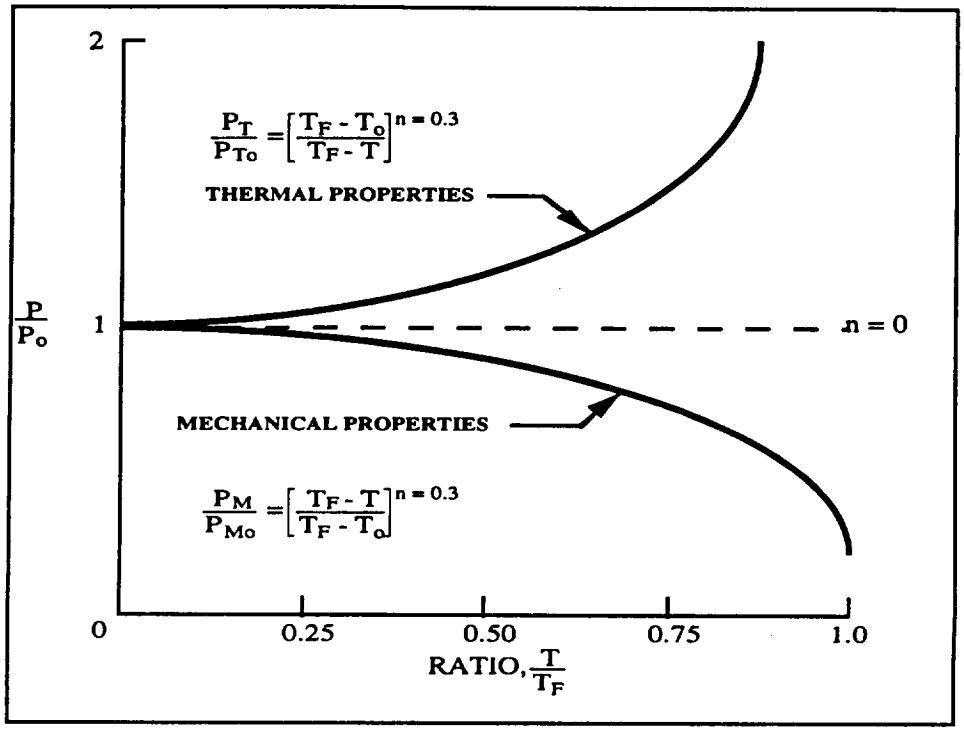


Figure 3-14. Illustration Of The Thermoviscoplastic Nonlinear Relationship Typical "Form" Behavior For A Given Exponent.

The micromechanical simulation of MMC analysis by METCAN provides the user with composite responses as a function of time as illustrated in Table 3-7 and shown in Figures 3-15 through 3-24. As shown a wide range of option is available for graphical output.

A graphics user interface using X11/Motif for GENOA/METCAN module allows the visualization of METCAN post processing files. GENOA/METCAN has the capability of plotting the ten postprocessing output files of METCAN from the executive control processed (Table 3-8) on a graphic user interface. These postprocessing files mainly contain the various mechanical and thermal properties of fiber, matrix and interface as a function of time.

The visualized graph of fiber/interface/matrix assembly superimposed with different contour lines, corresponding to different values of stress (Figures 3-25) or strain is processed by graphics output file from the METCAN output. This visualized unit cell consists of six sub regions (i.e., fiber, matrix A, B,C and interface B,C). Each sub region at each time is represented by a color corresponding to a stress or strain level.

Table 3-7. Visualization of Time Instantaneous MMCs Composite Constituent Properties

1-Plot Composite Temperature
Time instantaneous temperature of plies
2-Plot Fiber Properties of Plies
Time instantaneous fiber tensile stress in 11, 22, 33 Directions
Time instantaneous fibercompressive stress in 11, 22, 33 Directions
Time instantaneous fiber shear stress in 12, 23, 13 Directions
3-Plot Matrix Properties of Plies
Time instantaneous fiber tensile stress in 11, 22, 33 Directions
Time instantaneous fibercompressive stress in 11, 22, 33 Directions
Time instantaneous fiber shear stress in 12, 23, 13 Directions
4-Plot Inteface Properties of Plies
Time instantaneous Interface tensile stress in 11, 22, 33 Directions
Time instantaneous Interface compressive stress in 11, 22,33 Directions
Time instantaneous Interfaceshear stress in 23, 33, 12 Directions
5-Plot ply Properties of Plies
Time instantaneous ply tensile stress in 11, 22, 33 Directions
Time instantaneous ply compressive stress in 11, 22, 33 Directions
Time instantaneous ply shear stress in 12, 232, 13 Directions
6-Plot Fiber Stress and Strain of Plies
Time instantaneous fiber stress in 11, 22, 12, 23, 13, 33 Directions
Time instantaneous fiber strain in 11, 22, 12, 23, 13, 33 Directions
7-Plot Matrix Properties
Time instantaneous Matrix stress in 11A, 22A, 12A, 23A, 13A, 33A Directions
Time instantaneous Matrix stress in 12B, 22B, 23B, 33B, 11B Directions
Time instantaneous Matrix stress in 12C, 23C, 33C, 11C Directions
8-Plot Interface Stress and Strain Plies
Time instantaneous Interface tensile stress in 11, 22, 12, 23, 13, 33 Directions
Time instantaneous Interfacecompressive stress in 11, 22, 33 Directions
9-Plot Composite Stress and Strain Plies
Time instantaneous Composite Strain in XX, YY, and XY Directions
Time instantaneous Composite Stress XX, YY, XY, and ZZ Directions
Time instantaneous Ply strain in 11, 22, 12, 13, and 23, Directions
Time instantaneous Ply stress in 11, 22, 12, 13, and 23, Directions
10-Plot Composite Properties
Time instantaneous Composite Density Properties
Time instantaneous Composite Temperature

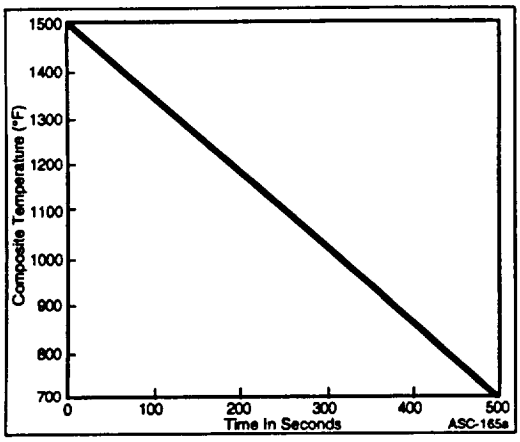


Figure 3-15. Composite Temperature Versus Time for Ply No. 1

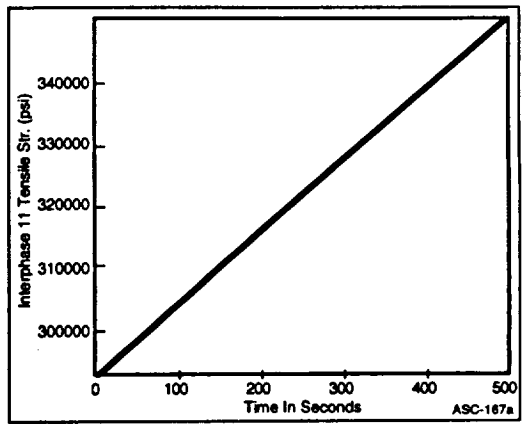


Figure 3-18. Interphase Tensile Strength (11) Versus for Ply No. 1

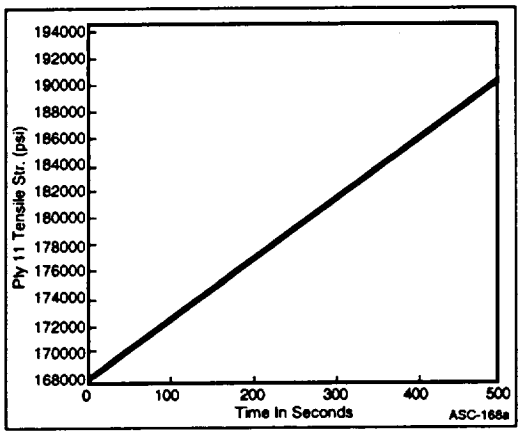


Figure 3-16. Fiber Tensile Strength (11) Versus Time for Ply No. 1

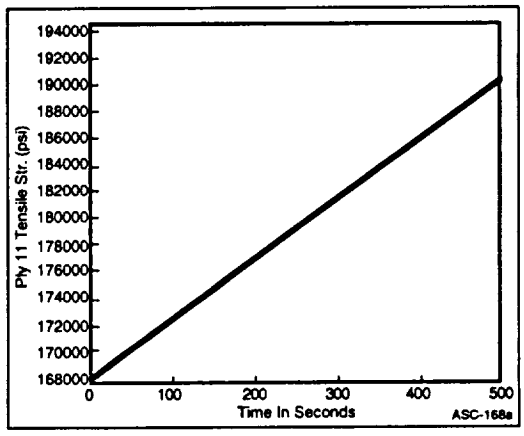


Figure 3-19. Tensile Strength (11) Versus Time for Ply No. 1

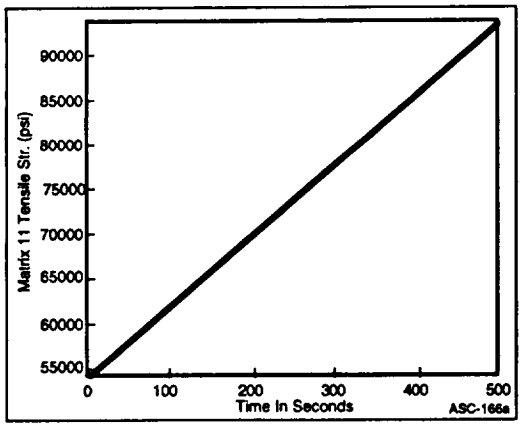


Figure 3-17. Matrix Tensile Strenth (11) Versus Time for Ply No. 1

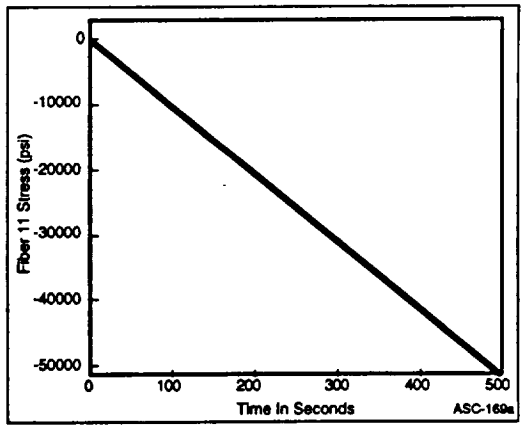


Figure 3-20. Fiber Stress (11) Versus Time for Ply No. 1

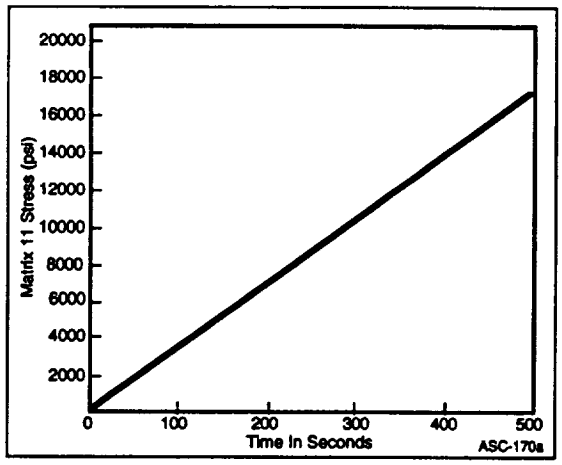


Figure 3-21. Matrix Stress (11) Versus Time for Ply No. 1

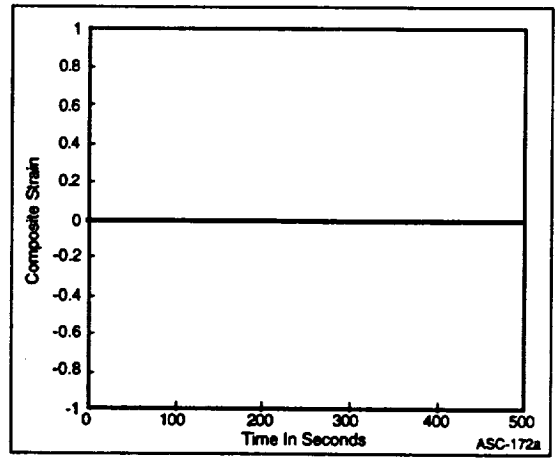


Figure 3-23. Composite Strain Versus Time for Ply No. 1

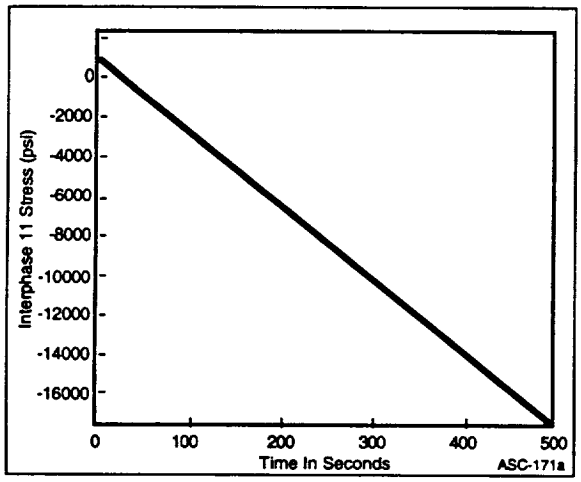


Figure 3-22. Interphase Stress (11) Versus Time for Ply No. 1

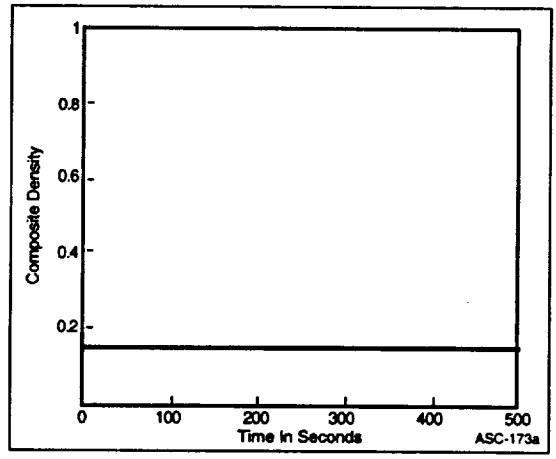


Figure 3-24. Composite Density Versus for Ply No. 1

3.4.4 CEMCAN Integration

The nonlinear version of Ceramic Matrix Composites Analyzer (CEMCAN) under development at NASA/LeRC was integrated as part of the GENOA system. The CEMCAN code incorporates various levels of composite mechanics models (substructuring) to allow a comprehensive point analysis of composite behavior [3.14]. The CEMCAN module input and output parameter panel is shown in Table 3-9. The CEMCAN control program is written to interact with the executor and build the control file for the CEMCAN module.

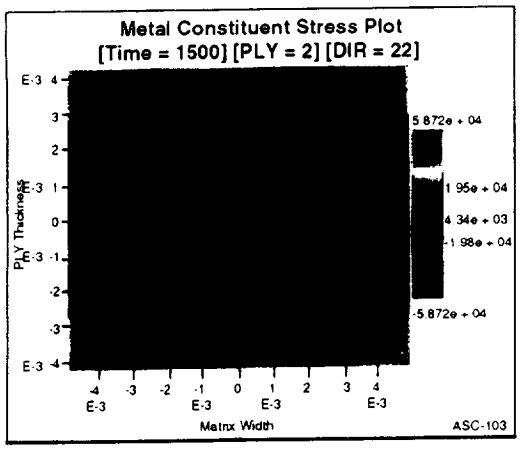


Figure 3-25. Metal Constituent Stress Plot

In the application of ceramic matrix composite used in CEMCAN computer code, the volume element or unit cell is assumed to be arranged in a regular patern. The unit cell consists of fiber, matrix and interphase. By applying the micromechanics equations which are based on mechanics of materials approach to the unit cell, the equivalent properties for the ply are obtained. The interphase is treated as a separate constituent with distinct properties. Thus, it can either represent a zone formed due to a chemical reaction between the fiber and the matrix or a separate layer or fiber coating provided intentionally to prevent such a reaction.

The CEMCAN application to ceramic matrix uses a volume element subdivided into several slices with micromechanics equations for each slice. These are subsequently integrated to obtain ply properties. Figures 3-26 and 3-27 demonstrates the graphical horizontal slicing, and vertical slicing of stress out put of CEMCAN in time domain. The user has the option to plot ply stress results in 11, 22, 33, 12, 13, and 23 directions.

Table 3-8. Metal Matrix Composite Analyzer File Names Input and Output Assembly by the Executive Controller

METCAN ---------- METAL MATRIX COMPOSITE ANALYZER FILE NAMES ---------- METCAN		
(L)	(A)	
INPUT	**SET A ON LINES 1 THRU 2**	
1)	DEMOME00.METCAN.IN	METCAN.IN
2)	DEMOME00.DATABNK.DB	DATABNK.IN
OUTPUT	**SET A ON LINES 3 THRU 15**	
3)	DEMOME00.LISTING	OUTPUT LISTING
4)	DEMOME00.MPOST01	01 PLY TEMP
5)	DEMOME00.MPOST02	02 FIBER PROP
6)	DEMOME00.MPOST03	03 MATRIX PROP
7)	DEMOME00.MPOST04	04 INTERF PROP
8)	DEMOME00.MPOST05	05 PLY PROP
9)	DEMOME00.MPOST06	06 FIBER STRESS
10)	DEMOME00.MPOST07	07 MATRIX STRESS
11)	DEMOME00.MPOST08	08 INTERF STRESS
12)	DEMOME00.MPOST09	09 LAMINA STRESS
13)	DEMOME00.MPOST10	10 LAMINA PROP
14)	DEMOME00.MPOST11	11 GRAPHICS
15)	DEMOME00.MPOST012	12 GRAPHICS
PRESS ENTER TO SAVE FILE NAMES AND RETURN		ASC-027

Table 3-9. Ceramic Matrix Composite Analyzer

CEMAN----CERAMIC MATRIX COMPOSITE ANALYZER----CEMAN

This code calculates fiber reinforced ceramic matrix composite property degration. CEMAN incorporates various levels of composite mechanics models to allow a comprehensive point analysis of composite behavior.

1 CONTROL - Set Option Control Parameters
2 DSNAMES - Define File Names

C CHECK - Check File Names Without Execution
E EXEC - Execute CEMAN Module

Select One Of The Available Options By Number/Letter

CEMAN--CERAMIC MATRIX COMPOSITE ANALYZER FILE NAMES--CEMAN

(L)	(A)	
INPUT	Set A On Lines 1 Thru 2	
1)	genoCE00.CEMAN.IN	CEMAN.IN
2)	new.dbk	DATABNK.IN
OUTPUT	Set A On Lines 3 Thru 18	
3)	genoCE02.LISTING	OUTPUT LISTING
4)	genoCE02.LSTRSTRN	LSTRSTRN
5)	genoCE02.TSTRSTRN	TSTRSTRN
6)	genoCE02.CPOST01	CPOST01
7)	genoCE02.CPOST02	CPOST02
8)	genoCE02.CPOST03	CPOST03
9)	genoCE02.CPOST04	CPOST04
10)	genoCE02.CPOST05	CPOST05
11)	genoCE02.CPOST06	CPOST06
12)	genoCE02.CPOST07	CPOST07
13)	genoCE02.CPOST08	CPOST08
14)	genoCE02.CPOST09	CPOST09
15)	genoCE02.CPOST010	CPOST010
16)	genoCE02.YSLICE	YSLICE
17)	genoCE02.ZSLICE	ZSLICE
18)	genoCE02.MICROSTS	MICROSTS

Press Enter To Save File Names And Return ASC-204

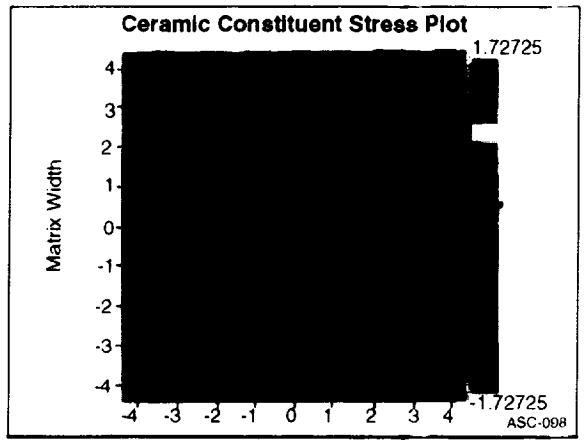


Figure 3-26. Ceramic Constituent Stress Plot - yy Slice

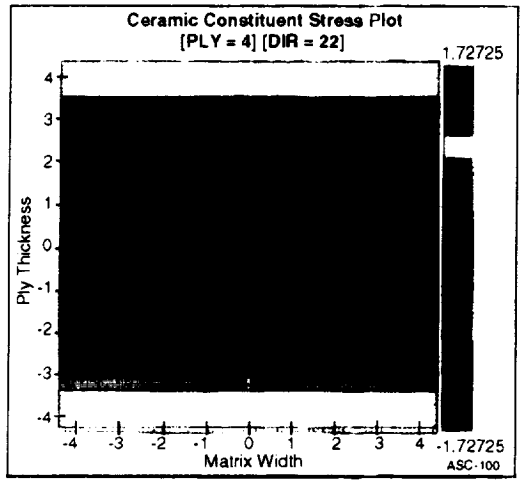


Figure 3-27. Ceramic Constituent Stress Plot - xx Slice

3.4.5 ICAN Integration

The Intgegrated Composite Analyzer (ICAN) computer code was integrated as part of the GENOA system. The ICAN control program is written to interact with the executor and build the control file for the ICAN module. The ICAN unit cell is depicted in Figure 3-28 demonstrates the graphical stress resultants. These can be ploted in the 11,22, 33, 12,, 13, and 23. directions .

GENOA has the capability to plot ICAN output information as shown in Figures 3-29 through 3-33.

3.4.6 Constitutive Relationship for Probabilistic Composite Micromechanics

Integrating Probabilistic Material Strength Simulator (PROMISS) in to GENOA gives GENOA the capability to perform probabilistic composite micromechanics analyses. This allows determination of strength degradation of composite materials due to the couple effects of various physical processes or

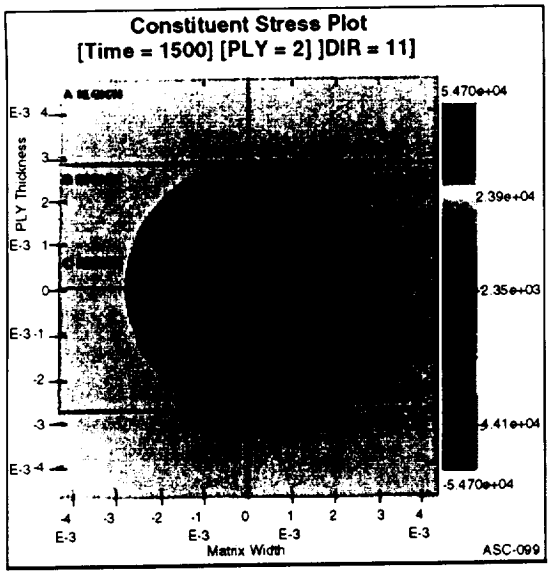


Figure 3-28. Constituent Stress Plot

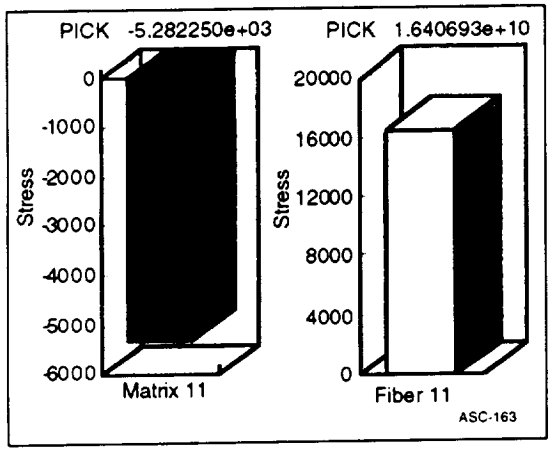


Figure 3-29. Stress Contribution for Fiber and Matrix (11 Directions)

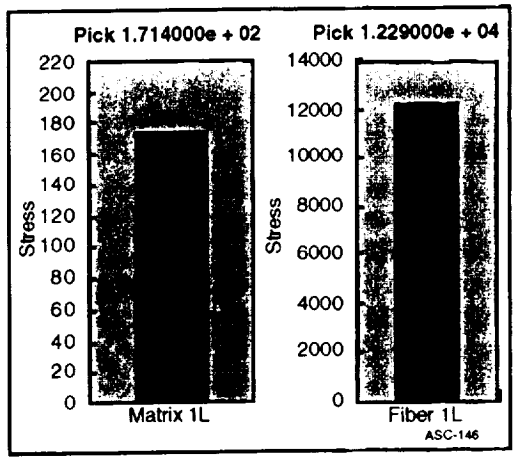


Figure 3-30. Stress Contribution in Longitudinal Direction for Matrix and Fiber

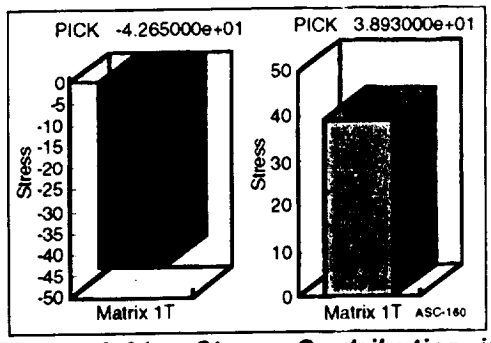


Figure 3-31. Stress Contribution in Transverse Direction for Matrix and Fiber

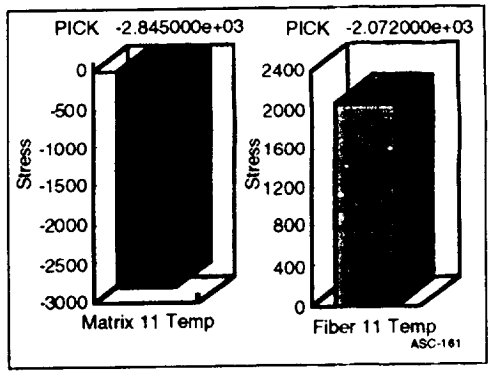


Figure 3-32. Thermal Contributionm to stress in 11 Directions for Matrix and Fiber

primitive variables. The unique capability of PROMISS evolves from the randomized treatment of the primitive variables incorporated in the constitutive equation. This equation is given as:

$$\frac{S}{S_o} = \prod_{i=1}^{n} \left[\frac{A_{iF} - A_i}{A_{iF} - A_{io}} \right]^{a_i}, \qquad (1)$$

where A_i, A_{iF} and A_{io} are the current, ultimate and reference values of a particular effect, a_i is the value of an empirical constant for the i^{th} primitive variable, n is the number of product terms of primitive variables in the model, and S and S_0 are the current and reference values of material strength. The program has considerable girth allowing deterministic (empirical via a co-developed program PROMISC), normal, lognormal, or Wiebull random modeling of the current, ultimate, reference values and empirical material constants for each primitive variable.

By incorporating PROMISS in GENOA users are given the capability of METCAN, CEMCAN, and ICAN to assess the failure probability of a specific region in the fiber/interface/matrix assembly, in the form of PDF and CDF plots of normalized strength degradation. Figure 3-34 shows a set up of the initial menu to execute the PROMISS code. As shown Normal, Lognormal, and Weibull distributions can be simulated for each specific region of the comosite constituents. The program also calculates for the user the suggested mean, and standard deviation, as well as the extracted information from a METCAN, ICAN, or CEMCAN simulation. Example realizations are shown in Figure 3-35.

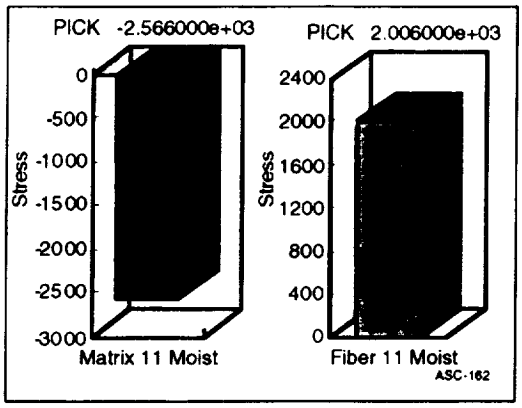


Figure 3-33. Contribution of Moisture to Stress in 11 Directions for Matrix and Fiber

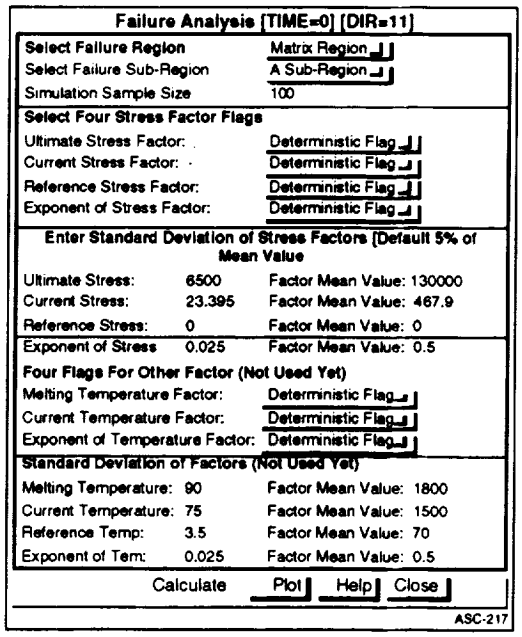
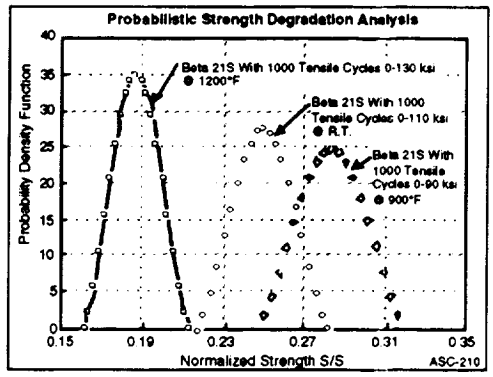


Figure 3-34. Set up of the Initial Menus to Execute the PROMISS Code



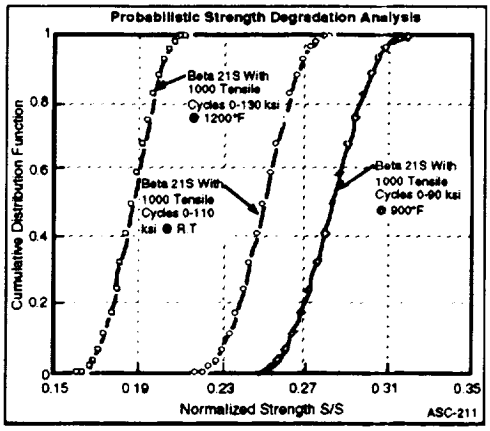


Figure 3-35. Example PROMISS PDF and CDF Output Showing the Calculated influence of uncertainties of Primitive Variables on the Tensile Strength Scatter for Beta 21S Foil

3.5 REFERENCES

3.1. C. C. Chamis, P.L.N. Murthy, and S. N. Singhal, "Computational Simulation of Hot Composites Structures," NASA TM-103681, 1991.

3.2. C. C. Chamis and M. C. Shiao, "IPACS - Integrated Probabilistic Assessment of Composite Structure: Code Development and Application," Third NASA Advanced Composite Technology Conference, Long Beach, CA, June 1992.

3.3. M. C. Shiao and C. C. Chamis, "Reliability of Composite Structures With Multiple Design Criteria," NASA TM-106716, 1994.

3.4. M. C. Shiao, S. N. Singhal, and C. C. Chamis, "Network Based Parallelism of Probabilistic Composite Structural Analysis," Proceedings of 36th AIAA/ASME/ASCE/AHS/ASC Structural Dynamics, and Materials Conference and AIAA/ASME Adaptive Structures Forum, pp. 889-899, 1995.

3.5. R. Calalo, B. Nour-Omid, and M. Wright, "An Implementation Of the Multifrontal Solution Algorithm on MIMD Computers," AIAA, pp. 1-9, 1992.

3.6. C. C. Chamis, Advances in 3-D Inelastic Analysis Methods for Hot Section Components," AIAA, pp. 802-811, 1987.

3.7. M. C. Shiao and C. C. Chamis, "A Methodology For Evaluating The Reliability and Risk Of Structures Under Complex Service Environments," NASA TM-103244, 1990.

3.8. D. A. Hopkins and P.L.N. Murthy, "METCAN - The Metal Matrix Composite Analyzer," Lewis Structure Technology, NASA CP-3003, Vol. 2, pp. 141-156, 1988.

3.9. H. J. Lee, P.L.N. Murthy, and C. C. Chamis, "METCAN Updates For High Temperature Composite Behavior: Simulation/Verification," NASA TM-103682, 1991.

3.10. P.L.N Murthy, C. A. Ginty, and J. G. Sanfeliz, "Second Generation Integrated Composite Analyzer (ICAN) Computer Code," NASA TP 3290, 1993.

3.11. D. A. Hopkins and C. C. Chamis, "A Unique Set of Micromechanical Equations For High Temperature Metal Matrix Composites," NASA TM-87154, 1985.

3.12. Ho-Jun Lee, "METCAN Simulation of Candidate Metal Matrix Composites for High Temperature Applications," NASA TM-103636, 1990.

3.13. P.L.N. Murthy, C. C. Chamis, and S. N. Singhal, "Hierarchical Nonlinear Behavior of Hot Composite Structures," NASA TM-106229, 1993.

3.14. S. K. Mital, P.L.N. Murthy, and C. C. Chamis, "Simulation of Ceramic Matrix Composites Behavior Including Progressive Fracture and Load Redistribution," Proceedings of 36th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and AIAA/ASME Adaptive Structures Forum, pp. 480-488, 1995.

4.0 Executive Controller And Graphical User Interface

4.1 AUTOMATIC FILE NAMING SYSTEM

The automatic file naming system is an executor scheme to assign system default filenames to all files when starting a project. This feature has been incorporated into all GENOA technical modules and provides options to use the system default filenames or to insert filenames by simply overwriting the default names.

Table 4-1. All Data Set Names Will Automatically Be Defined In Four Fields

Filename	Four Character Project Name
Filemode	Two Character module name that creates the file
Sequence Number	Two digits
Label	file descriptive name

Before describing how the automatic naming system works, the user needs to know a little about how UNIX filenames are defined. The word "filename" can have two meanings. Throughout most of this manual, "filename" will have the same meaning as "data set name." However the label can also refer to the first field of the data set name, as described below. In this manual, the word "filename" will refer to the complete data set name unless a specific reference is made to "filename" as the first field of the complete data set name. All filenames for UNIX must adhere to standard conversational operating system rules, of which the basic rules are as follows:

1. Where filename fields consist of from 1 to 4 alpha numeric characters, valid characters are A-Z and 0-9. The filemode, representing the actual load module descriptor which creates the file, can be two letters and chosen from the set A.....Z.

   Some examples of valid data set names are:

   ProME00.data

   testNE01.listing

   jobsCE11.control.data

2. The automatic file naming scheme is:

   (filename) (filemode) (sequence)

   a b c d ef qh

   where

   abcd Represents the name of the project on which you are working. This name was assigned to you when you started the project and is also the name that you typed on the first panel you encountered on initial entry into GENOA.

   ef Is the first and second letter of the name of the module being used (ME for METCAN, CE for CEMCAN, etc.)

   gh is a two-digit sequence number indicating how many cycles or versions of the file may exist. This number ranges from 00 to 99 inclusive.

As mentioned previously, the system default filenames can be overwritten with any name as long as they conform to standard UNIX field rules, or the default names can be modified by overwriting only that part of the default name which needs to be changed.

The system default filenames may appear in any executor panel containing fields for entering filenames. When first entering GENOA, all filenames are set to their default values. This is because the executor will only insert a default name if the field where the name to be entered is left blank.

4.2 PROJECT DIRECTORY

The project directory provides the user with the capability to work on multiple projects, jobs, or tasks and control the data menu that has been entered for each project. The user may easily switch back and forth between projects and can add new projects and delete old ones. The GENOA project directory is basically an information table containing information about certain files that exist for each project the user has worked on. A project catalog is provided to allow the user to select the project to work on. All project directory information is automatically created and saved in a file called GENOA.PROFILE in the current working directory from which GENOA was called. Therefore, it is important to establish a working directory within that will always be used to run GENOA. This does not preclude the use of input or output files from other directories or users.

4.2.1 Project Directory Usage

When the user first enters the GENOA executor system, a four-letter project name is requested with a short description of the project being worked on. This information needs to be entered only once - when the user creates the project. The four-letter project name is used by the executor to create and maintain the data for that project.

Example: If SSME is selected as the project name, then the project directory catalog entry is called SSME

The project directory can be accessed by one of two ways. One method is through the executor using Option 9 from the GENOA selection menu or by typing =PR from anywhere in GENOA. The other method, which is the easiest and most convenient way to view the directory, requires giving the user a list of projects previously created, once he is inside the directory. The user may choose any of the projects by typing in the corresponding project number and pressing the enter key. The user may also type A to add a new project or D to delete an existing project. Choosing A to add a project causes GENOA to prompt for a new four character project name and again for project description. This new project now becomes the working project when the user is returned to GENOA. Choosing D to delete a project causes GENOA to list the project numbers and names and ask for the project number to be deleted.

4.3 HISTORY

The history routine is an executor and UNIX function designed to save all terminal output on a permanent disk file. This feature allows keeping a permanent record of activities during a terminal login session for future reference or to aid in debugging problems.

Basically, the information stored on the history file is a listing of any commands have typed on the screen and the actions or messages resulting from those commands, as well as any messages sent to the system. With a history file, the latest session summary will be appended to the end of the existing history file; if the history file does not exist, a new history file is created. Once the history has been performed, it can be viewed, browsed, or edited just like any other file. Note that the history routine will provide only a summary of your current login session; it cannot "remember" anything from previous login sessions.

Panel Abbreviation and Shortcuts

The Panel Abbreviation and Shortcuts are designed to help the user to rapidly move from one panel to another. When learning to use GENOA , the user will probably move from one panel to the next sequentially by entering the appropriate response to each panel that appears on the screen. In this manner , he will be able to become familiar with the standard panel hierarchy. Once this is accomplished and he becomes more proficient with the system, unnecessary panels can be disgarded and he can go directly to the panel of interest; this eliminates responding to each individual panel. At this point, the user needs to know how to use the shortcuts and abbreviations in order to save time. As shown in Table 4-2 most of the primary option panels for each technical module have an abbreviation associated with them and the user can go directly to the panel by typing in the appropriate abbreviation from the panel currently being viewed, using the format

=M (where M represents the menu abbreviation)

Table 4-2. GENOA Menu Abbreviation List

CA	=	CADS	-	AUTOMATED GRID GENERATION
CE	=	CEMCAN	-	CERAMIC
IC	=	ICAN	-	POLYMER
M	=	ABBRE	-	LIST MENU ABBREVIATIONS (DEFAULT)
MA	=	MAESTRO	-	DRIVER FOR PARALLEL PROCESSING & OPTIMIZATION
ME	=	METCAN	-	METAL MATRIX COMPOSITE ANALYZER
NE	=	NESSUS	-	FEM ANALYSIS
PF	=	PFEM	-	PROBABILISTIC FEM ANALYSIS
PR	=	PROMISS	-	PROBABILIS MICRO ANALYSIS
RI	=	RIP/PEEL	-	RECURSIVE INERTIA PARTITIONING
SR	=	SRA	-	SYSTEM RELIABILITY ANALYSIS
NOTE: TYPE	=	M TO TRANSFER TO OPTION M, WHERE M IS THE MENU ABBREVIATION		
SELECT ONE OF THE AVAILABLE OPTIONS BY NUMBER				ASC-002

Examples:

1. If the user is currently in the METCAN panel (shown in Table 4-3) and he wants to switch to the METCAN CONTROL panel, type "1" and the panel shown in Table 4-4 will be obtained, or if he wants to define a data set name, type "2" and the panel shown in Table 4-5 will be obtained.

Table 4-3. Metal Matrix Composite Analyzer

METCAN ——— METAL MATRIX COMPOSITE ANALYZER ——— METCAN			
1	CONTROL	-	SET OPTION CONTROL PARAMETERS
2	DSNAMES	-	DEFINE FILE NAMES
C	CHECK	-	CHECK FILE NAMES WITHOUT EXECUTION
E	EXEC	-	EXECUTE METCAN MODULE
SELECT ONE OF THE AVAILABLE OPTIONS BY NUMBER/LETTER ASC-018			

Table 4-4. Set Option Control Parameters

METCAN ——— SET OPTION CONTROL PARAMETERS ——— METCAN				
(L)	(A)	(B)	(C)	(D)
MATERIAL	**SET A, B, C, D ON LINES 1 THRU 4**			
	MAT NUMBER	FIBER VOL RATIO	VOID VOL RATIO	FIBER/MATRIX
1)	1	.30	0.0	FI11MA11
2)	2	.30	0.0	FI12MA12
3)	0	.30	0.0	FI13MA13
4)	0	.30	0.0	FI14MA14
PLYS	**SET A,B,C,D ON LINES 5, 7 FOR NO. MECHANICAL THERMAL**			
	NUMBER CYCLES	FIBER LIMIT	MATRIX LIMIT	INTERFACE LIMIT
5)	3	1000000.	1000000.	1000000.
6)	0	400.	400.	400.
PRESS ENTER TO CONTINUE TO CONTROL OPTION PANEL 3 ASC-020				

Table 4-5. Metal Matrix Composite AnalyzerFile Names

METCAN ——— METAL MATRIX COMPOSITE ANALYZER FILE NAMES ——— METCAN		
(L)	(A)	
INPUT	**SET A ON LINES 1 THRU 2**	
1)	DEMOME00.METCAN.IN	METCAN.IN
2)	DEMOME00.DATABNK.DB	DATABNK.IN
OUTPUT	**SET A ON LINES 3 THRU 15**	
3)	DEMOME00.LISTING	OUTPUT LISTING
4)	DEMOME00.MPOST01	01 PLY TEMP
5)	DEMOME00.MPOST02	02 FIBER PROP
6)	DEMOME00.MPOST03	03 MATRIX PROP
7)	DEMOME00.MPOST04	04 INTERF PROP
8)	DEMOME00.MPOST05	05 PLY PROP
9)	DEMOME00.MPOST06	06 FIBER STRESS
10)	DEMOME00.MPOST07	07 MATRIX STRESS
11)	DEMOME00.MPOST08	08 INTERF STRESS
12)	DEMOME00.MPOST09	09 LAMINA STRESS
13)	DEMOME00.MPOST10	10 LAMINA PROP
14)	DEMOME00.MPOST11	11 GRAPHICS
15)	DEMOME00.MPOST012	12 GRAPHICS
PRESS ENTER TO SAVE FILE NAMES AND RETURN ASC-027		

4.4 JOB LINKING

The job linking program (JOBLNK) is an executor program that allows the HITCAN user to organize a sequence of execution commands The EXEC routines usually set up the file definitions and make calls to the various GENOA modules. Depending on the size of the GENOA analysis project and the computer being used , execution of several GENOA modules may take from minutes to days. Input to the job linking program is referred to as a task. The following example (Table 4-6) will invoke the JOBLNK function in HITCAN program:

Table 4-6 Sample JOBLNK task file

TASK	FUNCTION
JOBLNK	HITCAN
TASK1: NESSUS	Read # of nodes, temp, and loads
TASK2: Parallel METCAN	Run parallel Constituent analysis
TASK3: METCAN	simulate micromechanics laminate [A], [B], [D]
TASK4: MAESTRO	Set up cascading optimized simulation
TASK5: Parallel NESSUS	Perform FEM from results of TASK2
TASK 6: Parallel METCAN	Update [A], [B], [D]
TASK7: Parallel NESSUS	Update FEM from results of TASK5

4.5 HELP MENU

Command help provides on-line procedural assistance of the use of each GENOA command. Term and qualifiers are included for ease of understanding and explanation of system input. The user may include the COMMAND, TERM, or QUALIFIER after help to display the documentation of use with examples of the particular item desired. The user may enter only the alias or the first four characters of the item to obtain HELP. The alias for help is "HE" and is described in Table 4-7.

Table 4-7. GENOA ALIAS for HELP

TERM	DESCRIPTION
ACRO	Defines ACRONYMS in GENOA
ALL	Lists the entire HELP file
COMMAND	Describes input instructions of GENOA
QUALIFY	Describes the use of QUALIFIERS in GENOA

COMMAND	ALIAS	DESCRIPTION
=	=	Jump Function
BACK	/	Back up to previous panel
COLID	CO	Column/Line Identifier
BYE	BY	Exit the GENOA program
DONE	DO	Exit the GENOA program
EXIT	EX	Exit the GENOA program
END	EN	Exit the GENOA program
HELP	HE	Provides on line common documentation
PROJECT	PR	Select another project profile
REFRESH	RE	Refresh the terminal screen with latest input
SAVE	SA	Save the current variable values in the profile
STOP	ST	EXIT the GENOA program
TERMINAL	TE	Identify user terminal

The term "COLID" defines the interface mode between GENOA and the user. To allow GENOA to function on various terminals, the simple line command input format is used. Each panel that accepts user input will have column identifier across the third line of the panel and numbers done the left hand margin to define the data grid pattern. To identify which data is being defined the user must proceed the data by the combination column line ID which is the ALPHA character for the column and integer number for the LINE and no space in between; i.e., to select column C on the third line enter C3 followed by the data. If no data follows the COLID, GENOA will use the system default value, as an Example of Table 4-5. The user can type A1 SSME.METCAN, and the value of SSME.METCAN is inserted in DATA location under column A line 1

INPUT SET A ON LINES 1 THRU 2				
1)	DEMOME00.METCAN.IN			METCAN.IN
INPUT SET A ON LINES 1 THRU 2				
1)	SSME.METCAN			METCAN.IN

A multiple command may also be entered on a line terminated with a single return//ENTER. The total length of any line entry is still 72 characters. The user can use the semi-colon " ; " as delimiter to separate commands.,

Example: setting up multiple klues-2 4 and 8 ON, 3 and 5 OFF and the screen refresh

A2 1;A8 1;A4 1;A3;A5;RE

4.6 GRAPHICAL USER INTERFACE

In order to provide the GENOA solvers, geometry modeler and data I/O managers with state-of-the-art commercially viable capabilities, we have implemented a task objective which will aid in industry acceptance of GENOA. This task objective has focused on the development of industry standard, fully portable, graphical user interfaces for GENOA's internal code. The user interface for the GENOA integrated software package is developed using Motif, PHIGS and X Windows, Using these tools together, will allow us to create the desired multiple windowed graphical user interface (GUI), employing PHIGS to produce the graphics to visualize the material analysis data and X Windows and Motif tool-kits to create the other user interface (UI) components. These tools are also all industry standards, so the user interface will be portable to any system that supports the X protocol. This section will further explain the tools being employed to develop GENOA and the benefits from there use.

The user interface for GENOA is based on the X Window system or simply X, which is a network-transparent windowing system. With X, multiple applications can run simultaneously in windows, processing user input, generating multi-font text and generating graphics in monochrome or color displays. Network transparency means that application programs can be run on machines scattered throughout the network. Thus an application can take advantage of a distributed computing environment all the while handling user input and output from a single display. Because X permits applications to be device-independent, applications need not be rewritten, re-compiled, or even re-linked to work with new display hardware. X provides the facilities for generating multi-font text and two-dimensional graphics (such as points, lines, arcs and polygons) in a hierarchy of windows allowing applications to be independent of any machine, operating system or graphics hardware. Thus X is acting as an intermediary between GENOA and the display, handling output from GENOA to display and forwarding input (entered with a keyboard or mouse) to GENOA for processing.

Since X does not specify any particular interface style or how applications should respond to user input, the Motif tool-kit was employed to create most of the user interface. Motif is an international standard and is built on top of the X tool-kit to create a set of user interface components known as widgets. The Motif widget set implements many standard user interface components such as scroll bars, menus, drawing areas, and buttons that can be combined to create complex user interfaces. The Motif widget set is smoothly integrated with the X tool-kit, so applications can use Motif functions as well as X functions. Motif also defines a look and feel, which means that once users become familiar with a single application in Motif, they will be familiar with most other applications since they all must use the same basic widgets to create the application. Thus, Motif widgets were used to create most of the input tools necessary to get appropriate user input and to create the look and feel of the GENOA integrated software package.

The final tool needed to implement the GENOA user interface is PHIGS, which stands for Programmer's Hierarchical Interactive Graphics System, and is a programming library for three dimensional graphics. PHIGS is an international standard, that provides an application with a standard interface to many graphics output devices and is supported by the X protocol. PHIGS provides a high level graphics tool-kit with over 400 functions, that can transform abstract object descriptions into graphics requests. Most graphical output from GENOA is using PHIGS to produce three dimensional displays that users will be able manipulate, to view a picture from different angles, to view a picture from a more distant or closer point of view, etc. These pictures can provide a more powerful means of communicating the results of any analysis GENOA generates than the textual data alone can ever accomplish. Thus, by combining this powerful graphics package with Motif and X, GENOA is able to produce wonderful three dimensional graphs and charts interactively while being user friendly.

Now that the tools being used to develop the user interface for GENOA have been defined, it is necessary to describe the use of these tools. First, a main control window is created in which GENOA requires seeing and pointing rather than remembering and typing operations, thus, keeping user interaction simple and consistent. The main window input devices includes such items as a Motif 'menu bar' located at the top of the window, the menu bar will contain a set of general headings such as: File, Edit Search etc. Clicking on a menu bar item results in a Motif 'menu' appearing with a selection of commands that the user can select from, or maybe even another level of menus that the user must traverse to get to the final command menu. These menus are ideal for novice and infrequent users, and by the use of accelerators (short codes that can bring about the same event) more experienced users can be accommodated too. When a menu command is selected, a response might be to bring up another window. In this window might be more input devices such as: Buttons, text input widgets, toggle buttons, etc. Then after all needed input is received, another window might be generated to display graphics illustrating the analysis from the input data into GENOA. The user might then decided to modify some input parameters and visualize the resulting incremental change from the data analysis then the user might choose to look at a lower or higher level of analysis of the material by simply making another menu command selection etc. This is just one possible way the GENOA integrated software package to interact with the user, but the general principal will be the same. Each result from GENOA will require well defined steps from the user; each step will be specified with menus or another set of input criteria that the user must interact with. Thus, GENOA is an application that will be simple and consistent enough to use so that even users with limited knowledge of materials will be able to interact and get results successfully with out a large learning curve.

Customizing an application is also an important feature that X provides, this allows applications to change characteristics without being re-compiled or re-linked and allows users to setup an application to act as they wish. This feature is called a resource manager, the resource manager defines characteristics of X and Motif widgets used by the application, the resource manager has a standard language to specify widget characteristics and can be modified to allow extensions to specify application dependent resources if desired, The resources are used to specify default behavior, like what values input parameters might be set to initially, or what default flat file should be used to define certain material characteristics, or maybe even what colors should be used when drawing charts for analysis. All of the resource definitions are located in a user-defined resource database that the X server uses when starting up the application. By

allowing the user the ability to modify the resource data base, we are giving that user the ability to make the application as intuitive and easy to use from the users point of view.

With all of these graphical user interface features we have at our disposal, we can build an application like GENOA that presents a unified display for user interaction and input while using many different pieces of analysis code to do the actual calculations to generate the data needed to display charts and graphs. The analysis code is not a required part for the development of GENOA's user interface. However, it is essential for the functionality of the individual commands offered through the user interface of GENOA. This is an important distinction, because most material analysis applications have very unfriendly user interfaces which are interwoven into the data analysis part of the application. It would take a lot of work to separate the data analysis and user interface portions of the code to make them independent. With GENOA we avoid this problem by keeping these portions of code independent from the start, developing the user interface and material analysis portion of GENOA separately and then linking these portions together to create the final unified application. This process allows GENOA to be very flexible, allowing modifications and additions to GENOA without compromising the overall design. An example of the X/Motif GUI is shown in Figure 4-1.

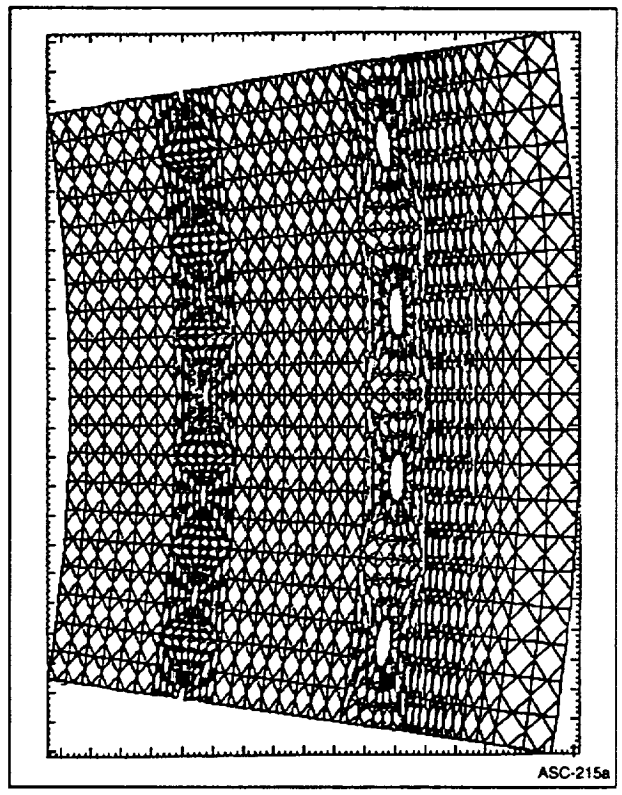


Figure 4-1. A Generic Finite Element Model by X/Motif GUI

Developing the graphical user interface of GENOA with the development tool-kits from Motif/X and PHIGS, and incorporating the user interface development independent of the data analysis, has made GENOA a application that is easy to use, understand, portable, configurable and most of all, a commercially viable material analysis software product. The combination of all these features in GENOA plus providing three dimensional graphics and powerful data analysis functions that users can interact with will make GENOA an application with many possible commercial, analytical and scientific applications.

5.0 Partitioning Finite Element Mesh For Concurrent Computing

Many large -scale computational problems are based on unstructured computational domains. Primary examples are unstructured grid calculations based on structural analysis problems utilizing finite element approximation. One of the key problems with implementing such large scale unstructured problems on a distributed memory machine is the question how to partition the underlying computational domain efficiently. Because of the recent interest in using multiprocessors, a large number of different algorithms for the partitioning problem was investigated. There are also many other approaches to the partitioning problem, which have been motivated by the evaluation of parallel processors. Simulated annealing has been used by William's [5.1, 2] and Nour-Omid et all [5.3]. Other algorithms motivated by physical consideration arising from structural analysis are a bisection algorithm based on the centroid of a structure and its principal direction. These have been proposed by Nour-Omid [5.4]. The Kernighan-Linalgorithm [5.5] a very popular algorithm bisection method, has been applied to structures by Vaughn [5.6]. Farhat [5.7,8,9] used a variation of the Greedy algorithm to obtain partitions that are suitable for implicit solvers, and an inertia type of algorithm for partitions that are suitable for implicit/explicit solvers where a local operator is factored but the interface problem is treated explicitly[5.10].

5.1 MESH PARTITIONING

We have investigated four algorithms for the partitioning problem for unstructured domains. All these algorithms considered here were recursive, i.e. the computational domain is subdivided by some strategy into two subdomains, and then same strategy is applied to the subdomains recursively. In this way a partition into $p = 2^k$ subdomains is obtained after carrying out k of these recursive partitioning steps. The four algorithms considered here thus only differ by partition strategy of a single domain into subdomains. The four algorithms are: 1) recursive coordinate bisection (RCB), 2) recursive graph bisection (RGB), 3) recursive spectral bisection (RSB) [5.1], 4) recursive Inertia Partitioning(RIP) , and 5) PEEL algorithm . Table 5-1 summarizes the five mesh portioning algorithms. The applicability of these techniques to the partitioning of structure problems was investigated. We have used RIP, and PEEL algorithm as the method of choice and integrated the software into the GENOA system.

5.2 RIP AND PEEL ALGORITHMS

Porting a finite element code from sequential machines to concurrent processors presents two problems. The first involves decomposition of the problem domain; the second is concerned with the development of parallel solution algorithms. The combination of the recursive inertial partitioning and the peeling algorithm is capable of partitioning a general finite element mesh into a set of balanced subdomains. It has also been shown to produce optimal partitions for a number of engineering problems.

Table 5-1. The Partioning Algorithms For Parallel Processing

Recursive coordinate bisection (RCB)
Determine longest expansion of domain (x,-, y-, or z-direction)
Sort vertices according to coordinate in selected direction
Assign half of the vertices to each subdomain
Repeat recursively (divide and conquer)
Recursive graph bisection (RGB)
Use of SPARSPAK RCM algorithm to compute a level structure
Sort vertices according to the RCM level structure
Assign half of the vertices to each subdomain
Repeat recursively (divide and conquer)
Recursive spectral bisection (RSB)
Compute Fiedler vector for graph using the Lanczos algorithm
Sort vertices according to the size of entries in Fiedler vector
Assign half of the vertices to each subdomain
Repeat recursively (divide and conquer)
PEEL ALGORITHM
Performs substructuring of the model based on geometric topology
Identifies exterior boundary nodes
Finds all attached elements to the exterior nodes and peels them off to produce super elements
RIP ALGORITM
Finds the least moment of inertia in the long direction
Minimizes the boundary region by rotation, and finally balances processors computational loadings.
Continues Partitioning until model domain is subdivided into the number of processors

Here we address the technical challenge of partitioning a finite element mesh into p substructures. The objective is to assign each element in the mesh to one of the p processors so that each processor is in charge of a single substructure. This step must be achieved such that the overall solution time is minimized. In the context of solving partial differential equations in space and time, domain decomposition refers to the technique of breaking the spatial domain of interest, Ω, into N smaller sub-domains Ω_i such that:

$$\Omega = \bigcup_{i=1}^{N} \Omega_i$$

and:

$$\Omega_i \cap \Omega_j = \phi$$

for $i \neq j$. Solving the differential equations on the whole domain Ω reduces to solving them on each sub-domain Ω_i, subject to common boundary conditions where the subdomains touch. If discrete solutions techniques are used to solve the differential equations, then each sub-domain is discretized with its own mesh and these multiple meshes form a composite grid. Denoting the set of all elements assigned to processor i by $\mathcal{E}_i$, this mesh partitioning process may be viewed as the discrete minimization problem

$$min_{\forall \varepsilon_i} \phi(\varepsilon_i, i = 1,...p) \tag{1}$$

where ϕ is a cost function that describes the overall analysis time. It reflects the imbalance in the computation load among the processors. The partition that minimizes ϕ keeps the number of interface nodes between pairs of substructures at a minimum to assure a small inter-processor communication time, while each processor performs almost an equal part of the overall computation. These objectives may be incorporated into a cost function that takes the form

$$\phi = \sum_{i=1}^{p} \sum_{j=i+1}^{p} [\gamma(\varepsilon_i) - \gamma(\varepsilon_j)]^2 + C(\varepsilon_i, \varepsilon_j) \tag{2}$$

where γ denotes the computation load and depends on the solution algorithm that is employed (e.g. explicit, implicit or semi-implicit schemes with direct or iterative solution procedures). C denotes the communication overhead between processors and also depends on the solution algorithm.

There are two factors that make the optimization problem in (1) difficult: (i) the evaluation of the cost function in (2) requires the solution time to be known a priori and thus is not easily computable, and (ii) the cost function in (2) has many local minimal. This makes standard optimization techniques ineffective. In fact, (1) falls into the category of N-P complete problems which includes the traveling salesman problem.

To overcome the first obstacle, a simplified cost function is implemented in (1). This function is obtained by assuming that γ is proportional to the number of elements in each substructure and C is proportional to the number of nodes shared by a pair of substructures.

The problem in (1) must be solved using techniques suitable for combinatorial optimization. In general, such techniques fall within the following two categories; (a) non-deterministic, e.g. the simulated annealing method, and (b) heuristic, e.g. the recursive inertial partition (RIP) method.

The method of simulated annealing is based on statistical mechanics. It provides an efficient simulation of the thermal motion of a collection of atoms to analyze aggregate properties of such systems. It is used to obtain the equilibrium state corresponding to a minimum energy at a given temperature. At each time-step, the state of the system is modified by a random disturbance and the resulting change in energy, $\Delta \phi > 0$, a probabilistic approach is used to determine if the new state is acceptable. The Boltzmann probability factor

$$P(\Delta\phi) = exp\left(-\frac{\Delta\phi}{k_b T}\right) \tag{3}$$

is computed and compared with a random number uniformly distributed in [0,1]. k_b is the Boltzmann constant and T is a temperature parameter. When this algorithm is applied to the above mesh partitioning problem, a change of state corresponds to exchange of elements among processors, thus

modifying $\mathcal{E}_i$, $i = 1,...p$. T takes the role of a free parameter and k_b becomes a normalizing constant. Although, this algorithm can find close to optimum partitions, the computation time may be substantial even with the most simplified cost functions.

In porting code from a sequential machine to a multiprocessor, of primary importance is the speedup factor. This factor is equal to the number of processors, p, when applied to a concurrent machine. Of course, even if the code is fully parallelizable, such a speedup is still unattainable due to the communication overhead on local memory machines and the problem of memory contention on shared memory machines. However, by breaking the problem up into p balanced subdomains with a minimum number of communication/interface nodes, it is indeed possible to approach the theoretical speedup.

In solving the partitioning problem, it is essential to produce p balanced subdomains. This implies that all processors have an equivalent amount of work to do, thereby minimizing the overall idle time. Since the concurrent portion of the code can only be executed as fast as the slowest processor, p balanced subdomains will yield the optimal speedup.

The solution process calls for an additional requirement of the partition. During this phase, the processors need to pass physical boundary information to their neighboring processors so that they may each work independently to solve their respective portion of the problem. In order to make this phase as efficient as possible, the algorithm must recognize the importance of interprocessor communication, and therefore produce a partition which contains a minimal number of interface nodes.

In essence, the partitioning strategy mimics the way a person would visually partition a structure. The algorithm ignores node and element numbering and concentrates on adjacency information to solve the problem. It recursively slices the structure in half, each time rotating the angle of the bisecting plane in order to minimize the number of nodes on the boundary. The recursive nature lends itself very well to application on a hypercube architecture since the partitioning strategy can easily be carried out in parallel.

Initially, a single processor (processor 0) is assigned all of the elements in the problem domain. After the first partition, processor 0 and 1 are each assigned approximately half of the elements, and the number of interface nodes between them has been minimized by rotating the partitioning plane (3-D) or partitioning line (2-D). Each of the domains in processors 0 and 1 are bisected separately and their interface nodes are minimized. The bisection process continues until p / 2 processors have completed the final partition and the results have been distributed evenly among all p processors. The RIP algorithm begins by calculating the long direction of the domain or subdomain. It then uses this vector along with a point exterior to the domain to define a plane which is efficiently passed through the structure until the domain is partitioned. The communication cost, which is defined as the number of nodes between the two domains, is then calculated for the partition.

In order to find an optimal partition, the plane is then rotated through a $\pm 45^{\circ}$ angle in the theta direction and then $\pm 45^{\circ}$ angle in the z direction. The communication costs for each partition are calculated and the vector associated with the minimum cost is then used to define the new bisecting plane. The partition associated with the final minimum cost is taken as the local optimum.

The first step in the algorithm is to find the long direction of the structure. In a problem domain with N_{elm} elements each with N_{node} nodes, the centroid x_k of the k-The element in the domain is calculated as:

$$x_k = \frac{1}{N_{node}} \sum_{i=1}^{N_{node}} x_i$$

Associated with the k-th element, a "computation mass" μ_k is assumed which is used to obtain a center of mass and a matrix of the moments of inertia. Thus, the computational centroid of the structure is calculated as:

$$x = \frac{1}{\mu} \sum_{i=1}^{N_{elm}} \mu_i x_i$$

where,

$$\mu = \sum_{i=1}^{N_{elm}} \mu_i$$

Then, for each cube dimension, the matrix of moments of computational inertia at this centroid are calculated according to:

$$\mathrm{J} = \sum_{i=1}^{N_{elm}} \mu_i (x_i - x)(x_i - x)^T$$

thereby producing the 3 x 3 matrix:

$$\mathrm{J} = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{xy} & J_{yy} & J_{yz} \\ J_{xz} & J_{yz} & J_{zz} \end{bmatrix}$$

The long direction is then the axis with the smallest moment of inertia which corresponds to the eigenvector with the smallest eigenvalue of the inertia matrix.

The long direction calculated above defines the normal to the plane that is used to partition the structure. The plane is positioned at a point exterior to the problem domain. The distance from the centroid of each element to the plane is then calculated and the list of element numbers are sorted according to distance. The first $\frac{1}{2} N_{\mathrm{elm}}$ elements are then assigned to one processor and the remaining elements to a second processor. This ensures that the subdomains are balanced assuming, as the RIP algorithm presently does, that all elements have an equal number of nodes per degree of freedom.

The RIP algorithm follows these general steps to partition a finite element model into substructures, S_i. Each S_i is a subset of nodes and elements from the original model. As a requirement of the multifrontal solution algorithm, use of RIP continues until partitioning of the substructures is not possible. This limit occurs when RIP cannot find moment of inertia axes. Each cut will produce two new substructures and a list of interface nodes, O_k^m where m is the level of the cut and k indexes the separator group which contains that set of interface nodes. The interface nodes are the nodes common between the two substructures. As an example, the mesh of Figure 5-1 is partitioned at three levels creating eight super elements. As shown each level correspondsto nodes such as Level 1 (with nodes 16, 17, 18, 19, and 20); Level 2 (with nodes 3, 8, and 13). Figure 5-2 shows the corresponding binary tree with the list of separator nodes. Each separator group O_k^m has at most two children and at most one parent in the binary tree. After RIP dismantles the original model, substructures are grouped together to make new aggregates. The set of O_k^m nodes within each aggregate specify an order for eliminating the degrees of freedom within the aggregate. The set of O_k^m nodes on the boundary of the aggregates specify the shared unknowns between substructures in neighboring processors.

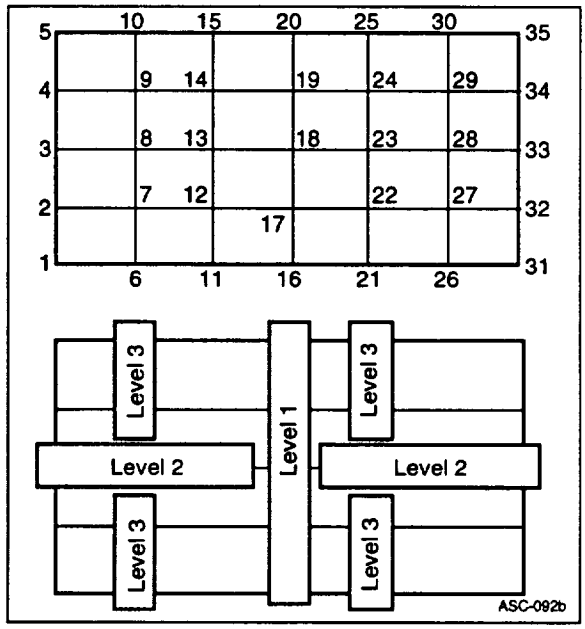


Figure 5-1. Application of RIP Algorithm 2D Model

5.3 FORCED PARTITIONING

The sensitivity of solutions to perturbations in geometric and material input parameters is often used in reliability analysis and design problems. In these cases one must run the problem repeatedly to obtain multiple solution results. For each new run, instead of performing calculations from the beginning of the finite element procedure, we use AMF to salvage most of the solutions from the first run. In other words, for each additional solution run after the first one, we just solve for only the region whose parameters are perturbed. The reason is simple as explained by referring to some AMF algorithms.

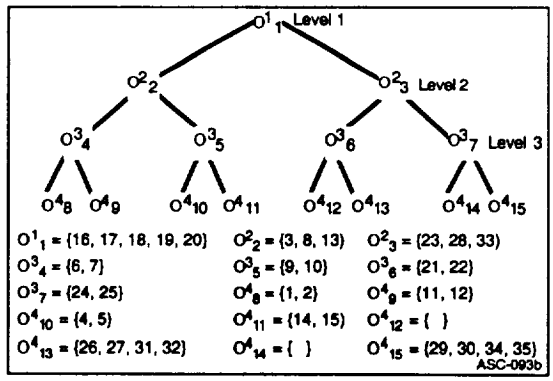


Figure 5-2. The Structure Of Binary Tree For The Domain Decomposition Of 2D Model. The List Of Nodes Of Separator Groups O_k^m Are Given Within The Braces.

In any direct solution algorithm for system of algebraic equations, one performs calculations which in form and steps are different from but mathematically equivalent to evaluating the inverse of the coefficient matrix. In AMF the finite element domain is first partitioned into several subdomain which

each constitute one super element. These super elements are combined to create new super elements in later steps after condensation procedure. In the process of condensing a super element, the coefficient matrix for the super element is partitioned so that the interior and exterior degrees of freedom are separated. Then, the interior coefficient matrix is decomposed into triangular matrices and processed further. This decomposition step is computationally the most expensive task performed on a super element. This is mathematically similar to calculating inverse of the interior matrix. The interior portion of coefficient matrix of all super elements are decomposed and stored in the climbing operation of AMF. Once the top of binary tree is reached, all the unknowns are calculated consecutively group by group by back substitution. The advantage of the method is that when parameters in a finite element subdomain are perturbed, only the condensed coefficient matrix for that super element and any other higher level super elements in that branch of binary tree are changed. Therefore, in a new solution the condensed matrices of untouched super elements of the binary tree are the same as in the first unperturbed solution. With careful domain partitioning, one can manage to collect all the perturbed elements in one super element so that only one branch of the binary tree be reevaluated for solution to the perturbed problem. As an example, the panel in Figure partitioned into substructures A and B (Figure 5-3). Each of these substructures are force partitioned further, and total of four new substructures, as shown in Figure 5-4, are created. The corresponding binary tree has three levels of interior nodes as shown in Figure 5-5 where S_i's are the list of interior nodes of the separators. S_1, S_2, and S_3 are the set of nodes shared by substructures A and B, I and III, and II and IV, respectively. S_4, S_5, S_6, and S_7 are the interior nodes of the four substructures (Level 3).

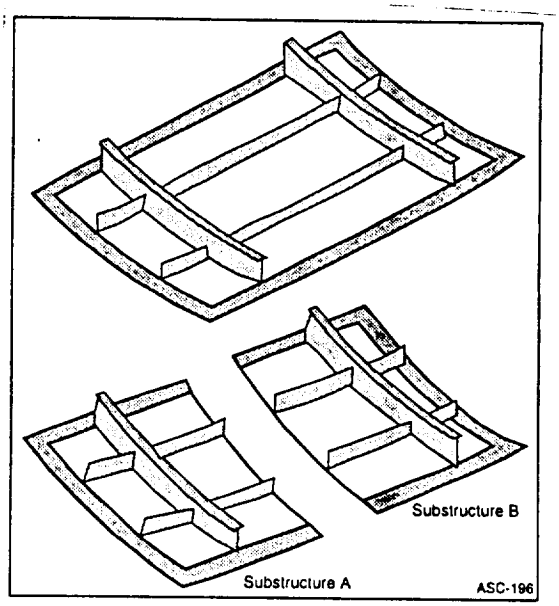


Figure 5-3. Panel Divided Into Two Substructures

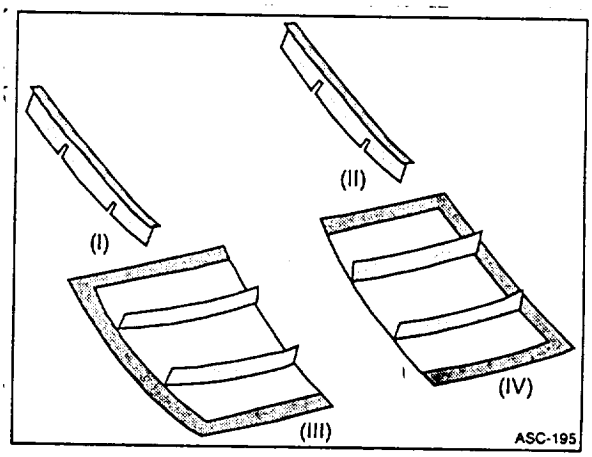


Figure 5-4. Forced Partitioning Into Four Substructures

According to AMF program, first the elements of each substructure are assembled and condensed at the third level of binary tree, followed by assembly and condensation procedure at higher levels. Now suppose that some geometric and/or material properties of substructure I is perturbed and another run is needed. In this case, the element stiffness (and the assembled matrix) in only substructure I is changed. Therefore, the assembled stiffness matrices, and more important of all, the condensed matrices of the remaining three substructures remain unchanged. However, the changes in substructure I leads to changes in super element 2 Level 2 which was created by combining exterior nodes of substructures I and III. Consequently, the condensation procedure for super element 2 must be repeated. Fortunately, at this level, the stiffness system to be condensed contains only the degrees of freedom

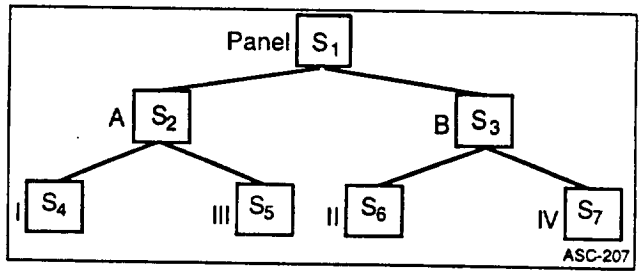


Figure 5-5. Management of Forced Partitioning

associated with the nodes on the intersection of substructures I and III which is relatively small compared to the whole substructure A . Similarly, the super element 1 at Level 1 is changed and only the degrees of freedom shared by substructures A and B are recondensed. Obviously, tremendous amount of computations is saved in the second run. Here, only three sub systems of equations are condensed. It is possible to reduce recomputations by using a combination of unsymmetric binary tree and force partitioning which actually shift the separator group S_4 to higher level in binary tree. An example of the binary tree for this case is shown in Figure 5-6 where S_i's are the same as the previous case The computation saving of the algorithm becomes even more apparent in large scale models. The larger the model is, the greater saving in computational time is achieved. For example, a perturbation in a panel of an aircraft requires solving only a system of equation as small as the degrees of freedom on the boundaries of that panel. That is reducing the size of the system to be solved from millions to hundreds. Figure 5-7 illustrates the forced natural, and mathematical partitioning. As shown, trade study can be performed by super elements (mathematical representation of a component) and material data in parallel processing environment. Changes in one substructure is automatically propagated into other substructures and the global panel by the management of forced partitioning. Therefore, a tremendous amount of computation is saved when sensitivity analysis is performed.

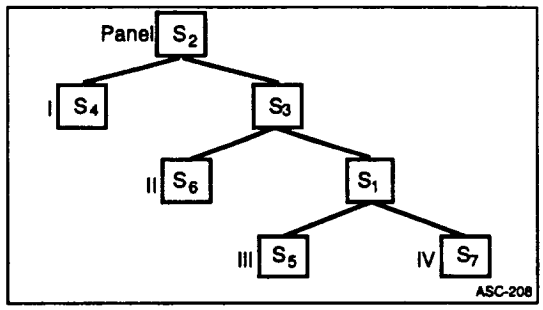
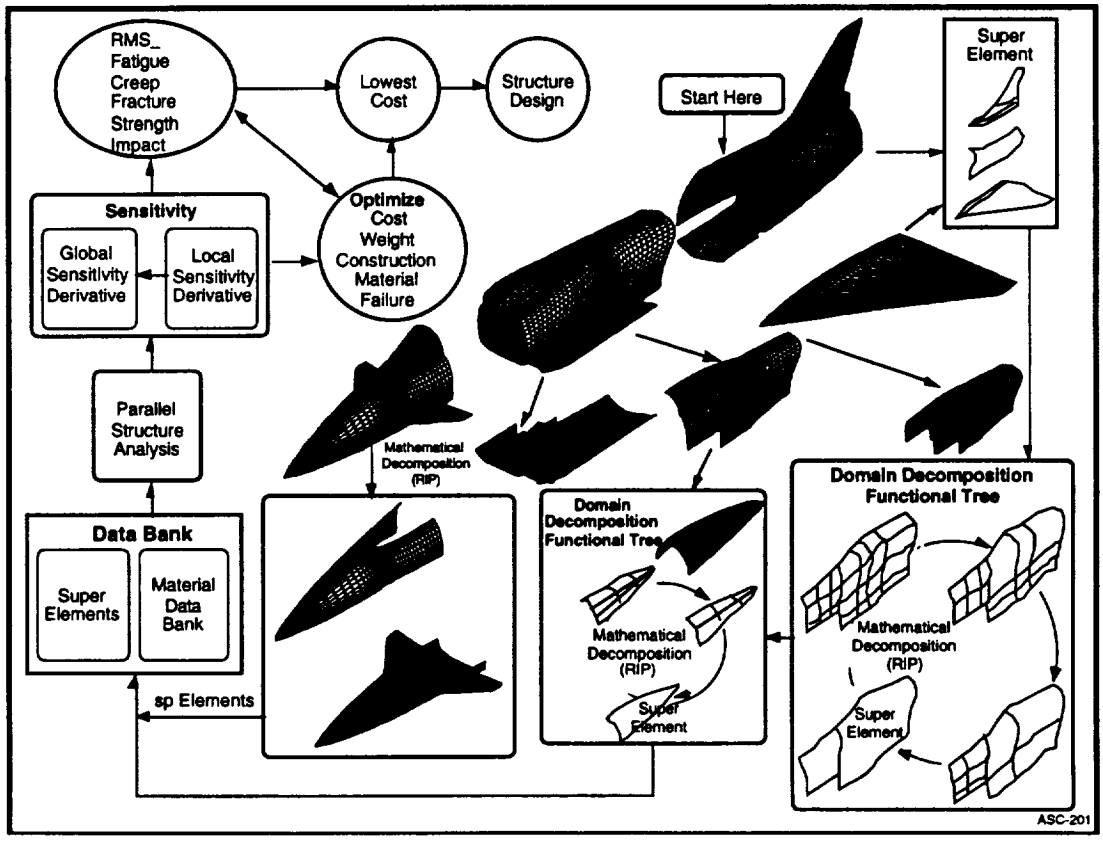


Figure 5-6. Changes In S4 Which Is Panel I Will Require Minimal Change For Global Solution



5.4 REFERENCES

5.1. A. Pothen, H. Simon and K.-P. Liou, "Partitioning Sparse Matrices With Eigenvectors Of Graphs," SIAM J. Mat. Anal. Appl., Vol. 11, No. 3, pp. 430-452,1990

5.2. R. D. Williams, "Performance of Dynamic Load Balancing Algorithms For Unstructured Mesh Calculations," Technical Report C3P913, California Institute of Technology, Pasadena, California, June 1990.

5.3. B. Nour-Omid, A. Raefsky and G. Lyzenga, "Solving Finite Element Equations On Concurrent Computers," Parallel Computations and Their Impact on Mechanics (edited by A. K. Noor), pp. 209-227, ASME, New York, 1986.

5.4. B. Nour-Omid, Private Communication, 1990.

5.5. B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure For Partitioning Graphs," The Bell System Technical Journal Vol. 49, pp. 291-307, 1970.

5.6. C. Vaughn, "Structural Analysis On Massively Parallel Computers," Proceedings of the Conference on Parallel Methods on Large Scale Structural Analysis and Physics Applications, Pergamon Press, Oxford, 1991.

5.7. C. Farhat, "On The Mapping Of Massively Parallel Processors Onto Finite Element Graphs," Computers and Structures, Vol. 32, No. 2, pp. 347-353, 1989.

5.8. C. Farhat and F. X. Roux, "An Unconventional Domain Decomposition Method For An Efficient Parallel Solution Of Large-Scale Finite Element Systems," Proceedings of the Copper Mountain Conference on Iterative Methods, Copper Mountain, Colorado, 1-5 April 1990.

5.9. C. Farhat, "A Simple And Efficient Automatic FEM Domain Decomposer," Computers and Structures, Vol. 28, pp. 579-602, 1988.

5.10. H. D. Simon, "Partitioning of Unstructured Problems For Parallel Processing." Computing Systems in Engineering Vol. 2, pp. 135-148, 1991.

6.0 Parallel Structural Mechanics Analysis Methods

When dealing specifically with the FE-based probabilistic structural analysis, there are several steps which must be considered in order to achieve optimal speed up on a parallel processing computer. Two levels of parallelism exist; a top level which results from parallelism associated with finite element aspects of the problem, and a lower level which results from structural material aspects of the problem.

Regardless of the level of parallelism, there are processor assignment routines, subdivision of simulation tasks, grouping of additional processors, memory allocation and information communication among processors assigned to individual sub-domains. Upon initiation of an analysis, there are numerous simulations necessary for each independent variable to be included in the structural response analysis. Having subdivided the available processors based on the structural model, it is then necessary to subdivide the simulation tasks and in, contrast, assign a task or set of tasks to a previously assigned processor or group of processors. The results of the simulated histories can then be shared by means of common blocks for developing element stiffness computations and assembly from common material property data. There are also the software portability to support all classes of the parallel computer architectures in a fashion which is portable; that is, the source code should be identical regardless of the hardware platform or memory architecture. This portability is acquired by using parallel-programming tool kits that disguise hardware and architecture-dependent code with common subroutine calls that do not change from machine to machine[6.1].

Therefore, simulation of large scale aerospace systems utilizing next-generation parallel computing architecture requires achieving a balance between requested time of the modeling and available properties of publicized resources. This chapter addresses in detail these specific issues [6.2 and 6.3] as well as *GENOA*/ MAESTRO and the multi-factor optimization which is a dynamic platform that continuously tests facility resources, distributes available computer resources, monitors machine utilization and analysis progress, controls environmental conditions in real time, and redistributes operational tasks in real time. *GENOA*/ MAESTRO (Real Time Parallelization Algorithm), when combined with multilevel design optimization (MDO) methodology, provides speed and efficiency not previously obtained in the design and analysis of large scale, high temperature, composite structure systems.

6.1 PROBLEM SCALING BY EXPLOITING MULTI-LEVEL PARALLELISM

The development and integration of the Hybrid Linear Solver and the Cascading Processor Assignment routines utilizing parallelization routines, has provided the GENOA solver with an efficient and scaleable capability for micro-scale parallelization.

Having selected the hypercubed architecture for the development of GENOA, we can now present some innovative concepts which will take advantage of scalability and exploit the inherent parallelism associated with the specific methodologies for an integrated software package. Before doing this, it is important to first understand the primary driver which dictates the level of parallelism which can be exploited. This driver is the ratio of available processor, P, to the problem size. In the case of structural analysis using a finite element mesh, this ratio is given as:

$$\eta = \log\left(\frac{P}{N_{\text{elm}}}\right)$$

where N_{elm} is the number of elements needed to model the structural response. In the past, it was common to assume that η was small e.g., there were far more structural elements than available processors. This prompted the development of algorithms like RIP and Peel performing domain substructuring to optimize processor loading. As multi-processor computers take advantage of

substructuring to optimize processor loading. As multi-processor computers take advantage of scalability and evolve into massively parallel architectures, it is safe to assume that the next generation hardware will provide for much larger values of η. When η does become larger (e.g., > 1), either by increasing the number of processors or reducing the number of elements, a need will arise for software which is specifically developed to further optimize processor loading based on additional levels of parallelism residing in the problem.

In a closely related effort, "Probabilistic Structural Mechanics Research for Parallel Processing Computers [Sues, *et al.*, 1991 [6.4]]," a detailed examination of the various levels of inherent parallelism was performed. The work presented in this report focused on the parallelism obtainable in performing a probabilistic structural analysis utilizing Monti Carlo simulation and reliability methods with finite element codes. Valuable insight is given with respect to scaling probabilistic structural analysis problems for computation in a parallel environment. It did not, however, account for the vast quantity of micro-scale parallelism associated with probabilistic micromechanics when specializing in structures made from laminated materials. Therefore, it is at this level that our efforts will focus.

Utilizing some of the findings of Sues and co-workers, the development of a new approach was built on the assumption that next generation hardware and finite element techniques will provide for $\eta > 1$ to incorporate the macro levels of parallelism with material dependent micro-scale parallelism. The key to realizing this capability then becomes the development of a processor assignment routine which mitigates the computationally intense repetition associated with probabilistic structural mechanics and probabilistic micromechanics while optimizing the processor loads. For descriptive reasons, we have named this routine "Cascading Processor Assignment." As depicted by the name, the approach will assign available processors to the cascading levels of parallelism (Figure 6-1).

Balancing computational processor loads is the primary motivation in determining processor assignment. Unlike the methods presented by Sues, the initial assignment of processors will be determined by the granularity of the structural finite element model rather than the number of simulation histories and recursive subroutines. Partitioning the structural model such that each element has been assigned a processor will guarantee that the initiation of the problem solution is performed in a discretely optimal manner.

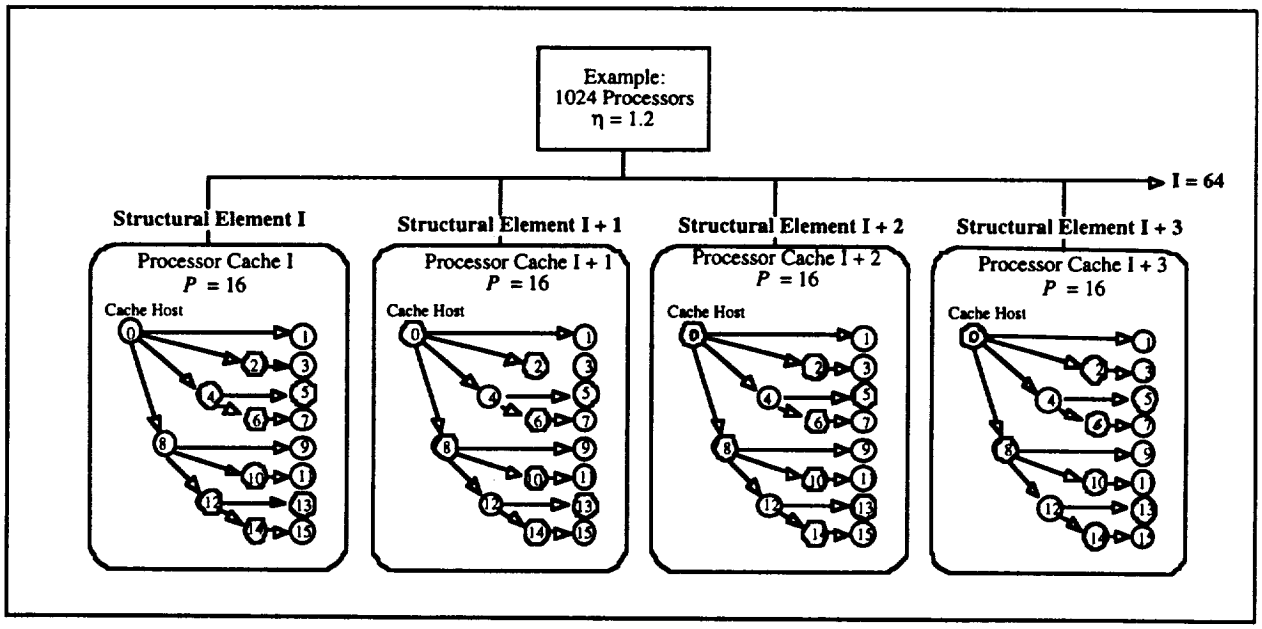


Figure 6-1. Illustration of a Cascading Processor Assignment for a Structural Model Having 64 Elements on a NCUBE with 1024 Processors

The concurrent computing environment is achieved by four key elements: (1) estimating and optimizing the requested memory and time interface connection among processors, (2) automatically analyzing and reconfiguring algorithms for the distributed and massively parallel architecture, (3) monitoring by generating memory and time allocation tables, and (4) managing data storage, work load balance, inter-processor communication, buffering, and message exchange among the active and non-active processors.

The first step in the RPA/optimization process is the selection of a set of processor resources (real and virtual) for the inherent connection to the separator tree operations. The last step is to control and manage the best combination of available computer resources such as machine type, network level, number of processors, available memory, and inter-processor communication message maps to exploit the optimal combination.

The concurrent computing environment is achieved by four key elements:

1) Estimating and optimizing the requested memory and time interface connection among processors. Figure 6-2 demonstrates testing Inter-Processor Communication, REAL *4, INTEGER *4, and LOGICAL <,> operations for network of machines in a facility. As shown based on the received information, a task assignment loading can be initiated.

Operation	HP 720	HP 735	IBM RS6K (1-16)	IBM RS6K (17-32)
IPC 400 KB	0.004770	0.004770	0.002345	0.003057
REAL *4	1.63	0.3899	0.02	0.19
INTEGER *4	0.61	0.2600	0.29	0.62
LOGICAL < , >	2.715	1.005	2.1	3.62

ASC-145

Figure 6-2. Results of the TFU and IPC for Various Computer Architectures

2) Selecting the initial guess for the arithmetic operation based on intended (dummy) real calculation , and estimation of the work load for the processors task. Assuming a symmetric LU decomposition, the work at each pivot I is calculated

3) Assuming that each element of the structural model has been assigned a cache of processors and a set of simulation tasks, the algorithm uses a modified logarithmic-cost fan-out to assign processors to sublevels of analysis. The element structural and materials analysis is performed by the local cache of processors.

4) Determining the simulation complexity of the material system, such as METCAN, and assigning a computational load for each processors to reduce the idle time between iterations and convergence evaluations.

6.2 ALPHA STAR MULTIFRONTAL (AMF) ALGORITHM

The Alpha Star multifrontal algorithm (AMF) analysis begins with the implementation of Recursive Internal Partitioning (RIP) binary tree operation which utilizes the FEM mesh to partition the entire domain into sub-domains and prepare the corresponding separate tree with desired number of levels. All the calculation associated with each sub-domain is assigned to an independent processor. At the first level the elements of each sub-domain are partially assembled into a super-element and then the interior nodes are removed by condensation. At the next level the condensed super-elements of previous level are grouped into several new super-elements and condensed again. As shown in Figure 6-3 this process is continued until the last level where a single super-element is left. At this level the unknowns at the remaining nodes are calculated and then the unknowns at the eliminated nodes of previous levels are recursively calculated by back substitution. AMF is designed to perform well for both symmetric and unsymmetric trees.

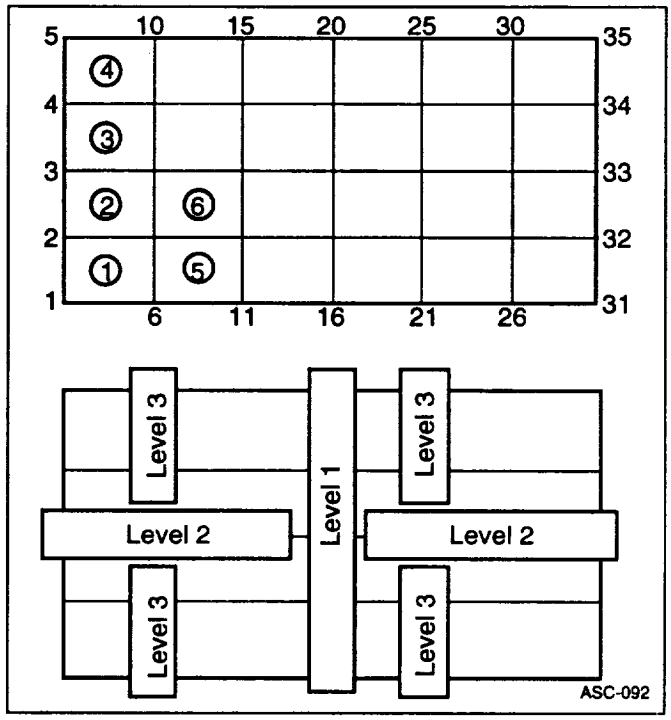


Figure 6-3. Application of RIP Algorithm to 2D Model

The program AMF was first developed on Ncube2 hypercube. The first version of the code was an extension of Calalo's work [6.5]. AMF has been modified to work on various computers with different architectures. It consists of 21 FORTRAN files, two input files, and one parameter file. The parameter file contains the maximum values of important parameters to allocate memory for running the code. A dynamic memory allocation has been assigned to AMF. The map of the AMF program is given in Figure 6-4.

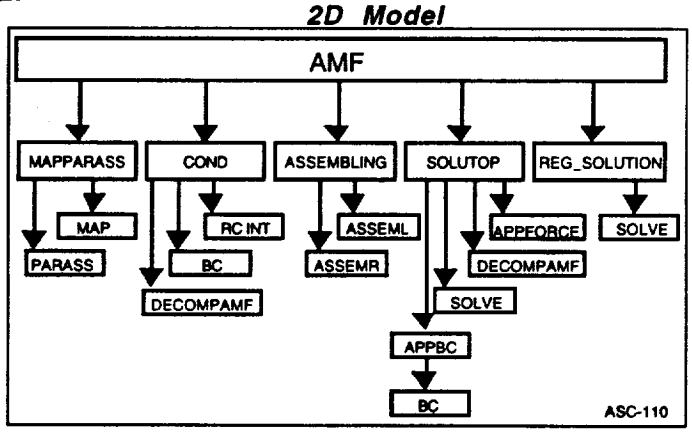


Figure 6-4. Map of the AMF Program

6.2.1 Matrix Assembly

As a first step in any finite element code, the element stiffness matrices and force vectors must be calculated. Any element equation generator can be used to calculate these matrices in a global system of coordinates. At this point AMF can take over and perform a series of steps of assembly and condensation, and finally solve the nodal unknowns, recursively. There are two possible assembly approaches: (1) node by node assembly in which the contributions from all elements to each node is determined and added node by node; (2) element by element assembly in which the contributions from each element to the global assembled matrices are summed. The AMF assembly is based on element by element approach which is very efficient. AMF has the capability to assemble mixed elements with different number of nodes.

The main payoff in AMF assembly is that the elements of each super-element are assembled separately. Therefore, several processors are utilized to perform the task in parallel and independently of each other. Another attractive feature of AMF is that no interprocessor communication is necessary during and at the end of assembly procedure. This saves time due to lack of overhead from the prosessor's synchronization. However, some interprocessor communication is needed after matrix condensation , where only portions of processed assembled matrices are to be shared among the processors.

6.2.2 Triangular Decomposition

In the discussion to follow it is assumed that the coefficient matrix has nice properties such that interchanges of rows and/or columns is never necessary in order to solve the equations. This is true in cases where A is symmetric, positive (or negative) definite. It may or may not be true when the equations are unsymmetric, or indefinite, conditions which may occur when the finite element formulation is based on a mixed variational principle or some weighted residual method. In these cases some checks or additional analysis are necessary to ensure that the equations can be solved [6.6]. If interchanges are necessary, furthure modification must be made [6.6, 7, 8].

First consider the coefficient matrix that can be written as the product of a lower triangular matrix with unit diagonals and upper triangular matrix, i.e.,

$$A=LU \tag{1}$$

This step is called the triangular decomposition of A. The construction of the triangular decomposition of the coefficient matrix is accomplished using a compact Crout method which is a variation on Gauss elimination. In practice, the operations necessary for the triangular decomposition are performed directly in the coefficient array: however, to make the steps clear, the basic steps are shown here using separate arrays.

The Crout variation of Gauss elimination is used to successively reduce the original coefficient array to upper triangular form. The lower triangular portion is not actually set to zero but is used to construct L. It is convenient to consider the coefficient array to be divided into three parts: part one being the region that is fully reduced, part two the region which is currently being reduced (called the active zone), and part three the region which contains the original unreduced coefficients. These regions are shown where the j^{th} column above the diagonal and the j^{th} row below the diagonal constitute the active zone. The algorithm for the triangular decomposition of an n x n square matrix is represented in Figure 6-5 and the following equations:

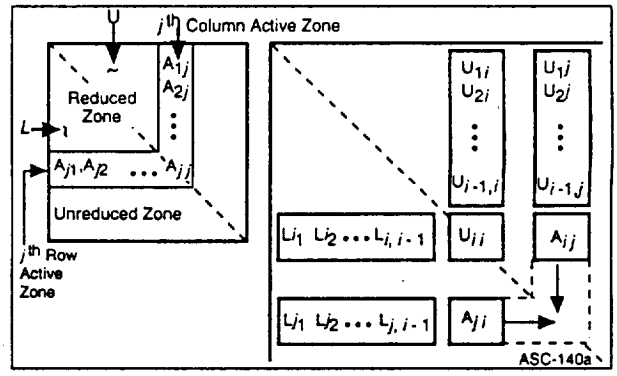


Figure 6-5. Illustration of Triangular Decomposition

$$L_{11} = A_{11} \tag{2a}$$

$$L_{11} = 1 \tag{2b}$$

For each active zone j from 2 to n

$$L_{j1} = A_{j1} / U_{11} \tag{3a}$$

$$U_{1j} = A_{1j} \tag{3b}$$

Then

$$L_{ji} = \left(A_{ji} - \sum_{m=1}^{i-1} L_{jm} U_{mi} \right) / U_{ii} \tag{4a}$$

$$(i = 1, 2, \ldots, j-1)$$

$$U_{ij} = \left(A_{ij} - \sum_{m=1}^{i-1} L_{im} U_{mj} \right) \tag{4b}$$

and finally

$$L_{jj} = 1 \tag{5a}$$

$$U_{jj} = A_{jj} - \sum_{m=1}^{j-1} L_{jm} U_{mj} \tag{5b}$$

The ordering of the reduction process and the terms used are shown in Figure 6-5.

The above mentioned algorithm is programmed in subroutine " decompamf.f". In order to reduce the total storage space, further decomposition is performed in " decompamf.f". For symmetric positive definite matrices, the relation $U_{ij} = L_{ji} U_{ii}$ and diagonal elements of **U** are used to rewrite the decomposition into the new form:

$$\mathbf{A = LDL^T} \tag{6}$$

where **D** is just a diagonal matrix. The diagonal elements of both upper triangular matrix $\mathbf{L^T}$ and lower triangular matrix **L** are unity. Only the upper and the diagonal matrices are stored after "decompamf.f." The nonzero elements of **D** are stored in diagonal of $\mathbf{L^T}$ since it is known by definition that the diagonal elements of $\mathbf{L^T}$ are all unity.

Matrix Condensation

Each time finite elements or super-elements are assembled, interior nodes are removed by matrix condensation. Rearranging the nodes and partitioning the matrix equation **KX=F** one can write

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{12}^T & K_{22} \end{bmatrix} \begin{Bmatrix} X_1 \\ X_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}$$

Where X_2 contains exactly all the fully summed (interior) nodes. A new condensed system of equations governing X_1 is created by removing X_2. The resulting equation is:

$$\overline{K}_{11} X_1 = \overline{F}_1 \tag{8}$$

where

$$\overline{K}_{11} = K_{11} - K_{12} K_{22}^{-1} K_{12}^T \tag{9}$$

and

$$\overline{F}_1 = F_1 - K_{12}K_{22}^{-1} \tag{10}$$

These condensed matrices are calculated in subroutine "cond.f" without inverting K_{22}. First "decompamf.f " is used to decompose K_{22} into triangular form $K_{22} = LDL^T$. The decomposition procedure is explained in the previous section with the only difference that, here, the matrix K_{22} is decomposed instead of matrix **A**.

After the decomposition of K_{22} is completed, the expressions $V = L^{-1} F_2$ and $W = L^{-1} K^T_{12}$ are calculated by forward reduction. Finally K_{11} and F_1 are evaluated by a series of matrix multiplications as follows:

$$\overline{K}_{11} = K_{11} - W^T D^{-1} W \tag{11}$$

$$\overline{F}_1 = F_1 - W^T D^{-1} V \tag{12}$$

The matrix decomposition not only simplifies the evaluation of reduced condensed system but also leads to new matrices which can be used later to solve for the condensed degrees of freedom by a simple back substitution algorithm.

6.2.3 Climb Operation

The multifrontal algorithm is a very powerful algorithm. It can be easily described by considering the partitioned structure of Figures 5-1 and 5-2, again. At the last level each super-element is assembled to create eight equations in the form

$$KX = F \tag{13}$$

where **K** is the stiffness matrix, **F** is the force vector, and **X** is the vector of unknowns in the super-element. The assembly can be performed simultaneously by different processors. Rearranging the nodes and partitioning **K**, **X**, and **F** one as in equation (7) where , X_2 contains exactly all the fully summed nodes. For each super-element, equation (7) is condensed and the interior nodes are removed. The details of condensation is given by equations (8) to (12). Subroutine "decompamf.f" which is responsible for the decomposition procedures and in turn called by subroutine "cond.f" for matrix condensation actually deals with equations (11) and (12) instead of (9) and (10).

Several processors are used in parallel, each responsible for setting up equations (1)-(10) for one or several super-elements. According to the binary tree, the condensed equation (8) of every two neighboring super-elements are combined (assembled) to form the super-elements of the next level. The governing equations for the new super-elements are similar to equation (13). Rearrangement of nodes, partitioning of matrix equations, and the condensation process can be repeated for the new super-elements. After the condensation of all super-elements, the binary tree is climbed up level by level. At each level a new parent super-element is created by assembling a pair of child super-elements. At the top of the binary tree (Level 1) only one super-element is created. All the nodes at the top are interior and fully summed. That means the entire vector **X** at this level is X_2 type. Therefore, all the unknowns at top level can be calculated at this point. These solutions are substituted in equation (7) governing super-elements of the next lower level (Level 2). The unknowns at the nodes of separator groups O^2_2 and O^2_3 are calculated. Similarly, the unknowns at the nodes of separator groups of lower levels in the binary tree can be evaluated by back substitution at each level, recursively. An obvious advantage of multifrontal algorithm is that all operations can be fully parallelized.

6.2.4 Triangular Solvers

Traditionally, the triangularized system of equation, **L U X = F** is solved by a combination of two steps:

Step 1: Forward elimination

In this step the product **U X = F** is calculated by solving the system **L Y = F**. The matrix **L** is a lower triangular matrix whose diagonal entries are all unity. Therefore, the elements of **Y** are calculated consecutively as followings:

$$y_1 = f_1 \tag{14a}$$

$$y_i = f_i - \Sigma_{j=1}^{i-1} L_{ij}\, y_i \qquad (i= 2, 3, ..., n) \tag{14b}$$

Step 2: Back substitution

In this step the system **U X = Y** is solved for **X**. Here, the matrix **U** is an upper triangular matrix whose diagonal entries are not necessarily unity. The solution procedure is similar to forward elimination with the exception that: (1) The elements of **X** are calculated from the last to first. (2) Generally, for each x_i calculation, a division by the corresponding diagonal entries of **U** is necessary. In terms of the elements of the equations, the solution is

$$x_n = y_n / U_{nn} \tag{15a}$$

$$x_i = (y_i - \Sigma_{j=i+1}^{n} U_{ij}\, y_j) / U_{ii} \qquad (i=n-1, n-2, ..., 1) \tag{15b}$$

The solver in AMF does not really need to perform both of the above mentioned steps. The reason is because the array **V** and the matrix **W**, which were previously calculated for matrix condensation in climbing operation, are utilized here also. The actual equation that AMF tries to solve is the one which governs the degrees of freedom at eliminated nodes at each separator group, namely:

$$K_{22}\, X_2 = F_2 - K^T_{12}\, X_1 \tag{16}$$

Here, the elements of X_1 are as calculated in higher levels of binary tree. The only unknown is the array X_2. Using the decomposed form of $K_{22} = L\, D\, L^T$, substituting $V = L^{-1}\, F_2$ and $W = L^{-1}\, K^T_{12}\, X_1$, and performing matrix algebra, equation 16 is rewritten as:

$$L^T X_2 = D^{-1} (V - W X_1) \tag{17}$$

In the climbing operation, the matrices **L, D, V,** and **W** at each separator group of the binary tree were calculated and stored. In the descending operation, the vector X_1 is known from the solutions to the equations at higher levels. Therefore, AMF solver has to solve equation (17) only. This means forward elimination step is no longer necessary. Equation (17) is solved in subroutine "solve.f" by a simple back substitution as described by equations (15) in step 2.

6.3 MESSAGE-PASSING SYSTEMS IN GENOA

The message passing paradigm is the conventional approach to parallel programming. In fact, most UNIX-based operating systems provide native, hardware-specific, function calls for performing message-passing. All implementations of message-passing provide nearly identical capabilities; thus, a program which is written using one message-passing toolkit can easily be ported to another message-passing toolkit. For this reason, where choosing a message-passing toolkit, the primary concerns are those of hardware support, stability and reliability, and acceptance in the community of hardware and software manufacturers.

GENOA took advantage of two different messsge-passing system which are used very often in scientific and commercial parallization systems. GENOA paralellization system was implemented on the nCUBE processor utilizing the mathematical library supported by this system. To increase the independence GENOA from architecture and operating system, it was decided to use PVM libraries to create a software which is portable to most significant system architecture on the market. Therefore, GENOA can be ported to Shared Memory, Distributed Memory, and Hybrid systems, (as shown in Figure 6-6). GENOA is designed to utilize and optimize any kind of message-passing parallelization system as required by the system administrator. GENOA has been ported to machine architecture and operating systems supported by the PVM, as shown in Table 6-1.

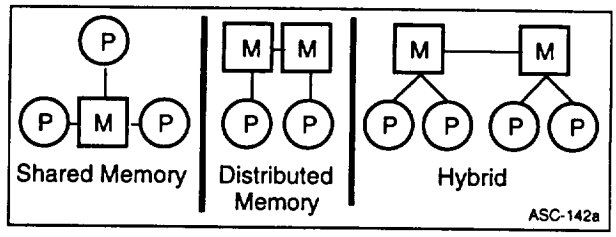


Figure 6-6. Three Types of Parallel Architecture

Table 6-1. GENOA Port is Utilized by PVM Architecture

PVM_ARCH	Machine Architecture	
HPPA	HP-9000 PA-RISK	OSF
RS6K	IBM/RS6000	MIMD
CSPP	Convex Exemplar SPP-1000	SIMD

In this section a short description of both systems is described. First nCUBE and its physical specifications, and interprocessors communication are described. Secondly, the physical environment of PVM and its architecture independent features of these libraries are described.

The nCUBE 2 Processor

The nCUBE 2 Series Parallel Computer supports a hypercube array of up to 2 power 13 (8,192) individual nodes (Figure 6-7 shows a four dimensional Hypercube). Each node consists of an nCUBE 2 Processor and 1 to 64 Mbytes of Memory.

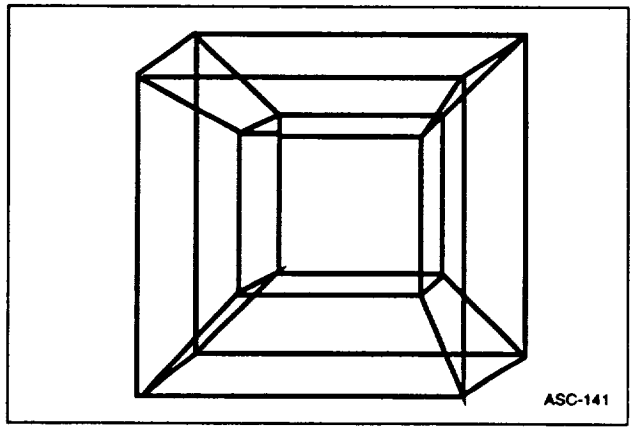


Figure 6-7. A Four Dimensional Hypercube

Each nCUBE 2 Processor consists of:

1- A general-purpose 64-bit central processing unit (CPU). The CPU includes:

2- An Instruction Decoder and Interrupt Controller (ID/ICU)

- an Execution Unit (EU)
- an Operand FIFO/Cache
- An error-correcting memory management unit (MMU)

3- A network communication unit (NCU) which includes 14 direct memory access (DMA) ports which support the hypercube interconnection scheme. The DMA ports include:

bi-directional interprocessor communication ports (26 unidirectional channels) for communicating with processors that are part of the local hypercube

bi-directional system interconnect (SI) I/O port, consisting of the local hypercube space (processors which are not configured as part of the local hypercube) or foreign (non-hypercube) systems. Table 6-2 illustrates the main features of an nCUBE message passing library.

Table 6-2. Highlights OF nCUBE

Transferring a Message to Local Memory	Meeting a set of condition, a processor will transfer a message to local memory
Forwarding a Message	Forwarding is used to communicate between remote cube spaces and to create arbitrary paths to specific nodes
Broadcast Facility	Provides the ability to create complex tree-structured paths for delivery of a message to multiple destinations
Local Broadcast	Initiated by the CPU, is the only way a single message can be routed to all the nodes in a subcube
Remote Broadcast	Minimize the number of open channels when communicating with a subcube that does not include the source node
Protection Logic	Protects system software and allow multiple processors per node
Interrupts Signal	Created by CPU when a channel is available and whether an error occurred during data transmission
External Interrupt Facility	Allows the processor to interrupt, or to be interrupted by, another processor

Communication in nCUBE

The nCUBE 2 Processor's Network Communication Unit (NCU) has been designed to provide full communication support for hypercube parallel processing systems.

Communication between processors is accomplished via messages. Message transmission proceeds in three stages: Path Creation, Data Transmission, and Path Removal. Each stage is performed in turn by each processor in the message path .

The architecture of the NCU has three layers: the Interconnect Layer, the Routing Layer, and the Message Layer.

- Twenty-eight independent serial DMA channels provide 14 full-duplex I/O pots, allowing systems to be designed with up to a dimension 13 hypercube.
- Direct interconnection of I/O ports eliminates off-chip communication logic.
- Variable communication speed of I/O ports allows matching the port speed to the signal propagation time of the interconnect.
- Automatic cut-through message routing allows messages to pass through intermediate nodes without interrupting the intermediate node CPUs.
- Automatic overrun control, via internode handshaking, allows reliable traffic through intermediate nodes across interconnects of differing speeds.

Figure 6-8 shows the interconnection of I/O channels though the communication ports.

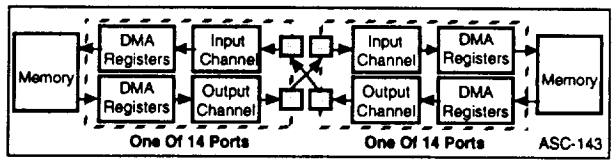


Figure 6-8. The Interconnection of I/O Channels Through Communication Ports

PVM LIBRARIES

PVM can be used on hardware consisting of different machine architectures , including single CPU systems, vector machines, and multiprocessors such as: 1) Simple Instruction Multiple Data, 2) Multiple Instructions Multiple Data, 3) Distributed Computing workstations (OSF), and 4) HYPERNODE as shown in Figure 6-9. These computing elements may be interconnected by one or more networks, which may themselves be different (e.g. one implementation of PVM operates on Ethernet, the Internet, and a fiber optic network). These computing elements are accessed by applications via a standard interface that supports common concurrent processing paradigms in the form of well-defined primitives that are embedded in procedural host languages. Application programs are composed of components that are subtasks at a moderately large level of granularity. During execution, multiple instances of each component may be initiated.

Application programs view the PVM system as a general and flexible parallel computing resource that supports shared memory, message passing, and hybrid models of computation. This resource may be accessed at three different levels: the transparent mode in which component instances are automatically located at the most appropriate sites, the architecture-dependent mode in which the user may indicate specific architectures on which particular components are to execute, and the low-level mode in which a particular machine may be specified. Such layering permits flexibility while retaining the ability to exploit particular strengths of individual machines on the network. The PVM user interface is strongly typed; support for operating in a heterogeneous environment is provided in the form of special constructs that selectively perform machine-dependent data conversions where necessary. Inter-instance communication constructs include those for the exchange of data structures as well as high-level primitives such as broadcast, barrier synchronization, mutual exclusion, global extrema, and rendezvous.

The most important advantage of PVM compare to other parallel systems are:

- Architecture Independence
- Heterogeneous Applications
- Network Portability

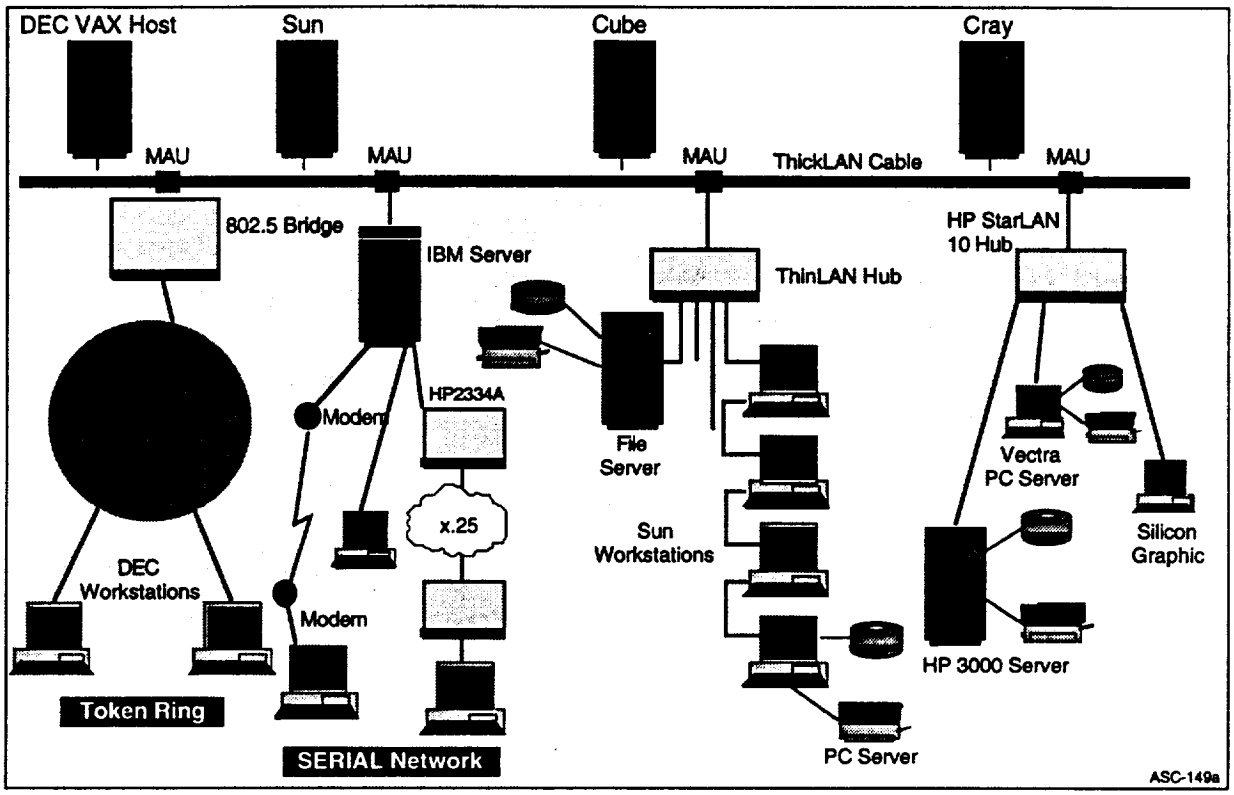


Figure 6-9. A Possible Combination of Various Networks and Hosts Building Parallel Virtual Machines

Heterogeneous applications are those that are composed of subtasks that differ significantly from one another. Particularly in scientific computing, there are many such applications. The components of such applications exhibit diverse characteristics including vector processing, large-grained SIMD computing, and interactive 2-D and 3-D graphics. The traditional solution to this problem is to execute each component separately on the most suitable architecture and construct manual, application-specific interfaces among them.

Several different network architectures, as shown in Figure 6-9, are supported by the PVM system, both for reasons of wider applicability as well as better exploitation of specific features of particular networks. PVM supports different methods for communication between processors over the network. Table 6-3 shows the most significant features of PVM Libraries

Table 6-3. Different Aspects of PVM

Peer to Peer Message Passing	A processor can send a message directly to other processor
Multicasting	A processor can send a message to multiple destinations
Message Packing	Coding/Decoding routines for messages to support different architectures
Blocking/Nonblocking Message Passing	Syncronization
Different size of message Buffers	Application can be tuned up for different network with different Bandwidth
Static Load Balancing	The problem can be divided an distributed at the begging of the run.
Dynamic Load Balancing	Master/Slave Method (pool of tasks paradigm)
Mixture of Static / Dynamic Load Balancing	Using the best combination of processor available on the network

6.3.1 Messages And Routing In PVM

Messages are sent by calling function send message( ), which routes the message by its destination address. If for remote destination, message fragments are attached to packets and delivered by the packet routing layer. If the message is addressed to the PVMD itself, send message( ) simply passes the whole message descriptor to net entry( ), the network message entry point, avoiding the overhead of the packet layer. This loopback interface is used often by the PVMD (PVM Demon). For example, if it schedules a request and chooses itself as the target, it doesn't have to treat the message differently. It sends the message as usual and waits for a reply, which comes immediately. During a complex operation, net entry( ) may be reentered several times as the PVMD sends itself messages. Eventually the stack is unwound and a reply goes to the originator. Figure 6-10 shows the general schematic of Packet and Message Routing.

Direct routing allows one task to send messages to another through a TCP link, avoiding the overhead of copying them through the PVMDs. This mechanism is implemented entirely in libpvm, by taking advantage of the notify and control message facilities. By default, any message sent to another task is routed to the pvmd, which forwards it to the destination. If direct routing is enabled (pvmrouteopt = PvmRoute Direct) when a message (addressed to a task) is passed to mroute( ), it attempts to create a direct route if one doesn't already exist. The route may be granted or refused by the destination task, or fail (if the destination doesn't exist). The message and route (or default route) are then passed to mxfer( ). The state diagram for a connection is shown in Figure 6-11.

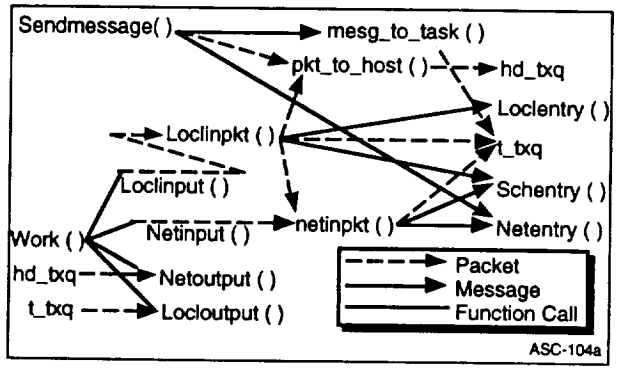


Figure 6-10. Packet and Message Routing In PVM

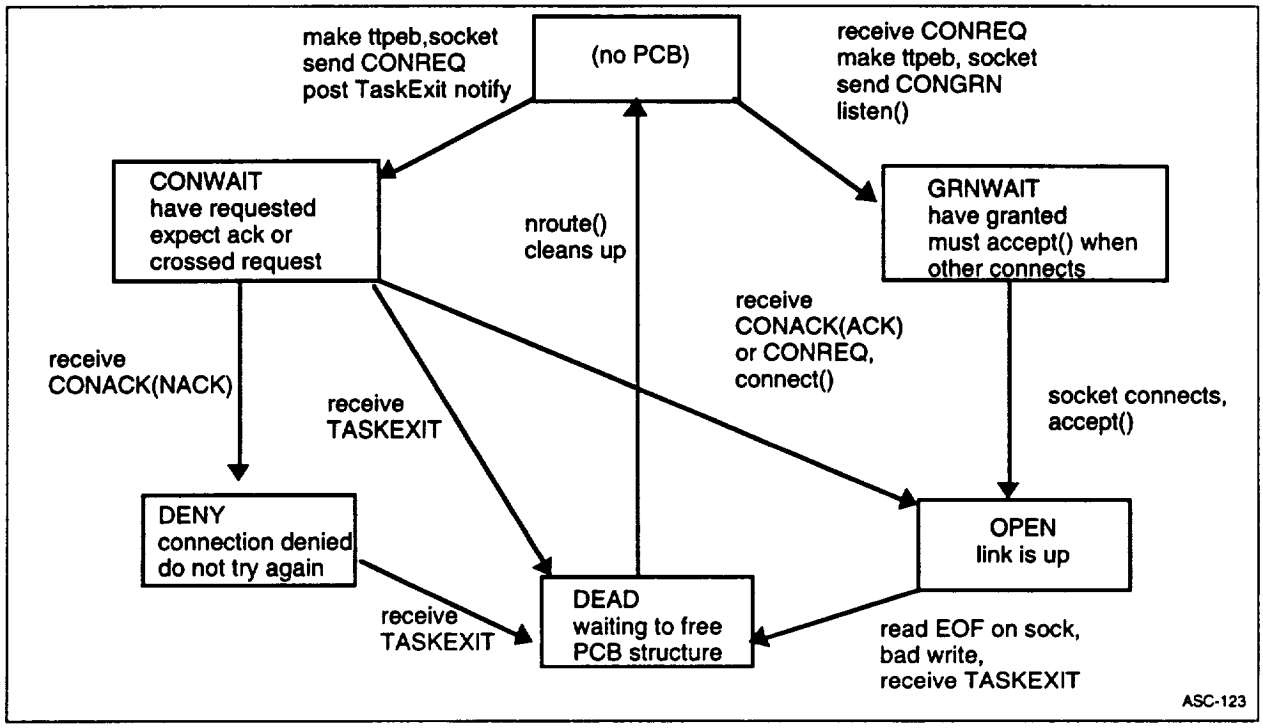


Figure 6-11. Task-Task Connection State Diagram - Direct Routing Allows One Task to Send Messages to Another Through TCP Link

6.4 MAESTRO: A DYNAMIC LOAD BALANCING SYSTEM

Figure 6-12 illustrates the flow diagram of the dynamic load balancing "MAESTRO" Real time parallelization system. The software performs numerical integer optimization for parallel processor loading based on the instantaneous and real time testing of computer facility. The individual elements of the MAESTRO are depicted as: 1) Domain Decomposer, 2) Supervisor, 3) AMF, 4) PVM, and 5) optimizer. Tables 6-4 and 6-5 illustrates the Task Allocation table before and after the MAESTRO optimization, as shown reduced number of operations during optimization (Table 6-5) will result in significant CPU reduction.

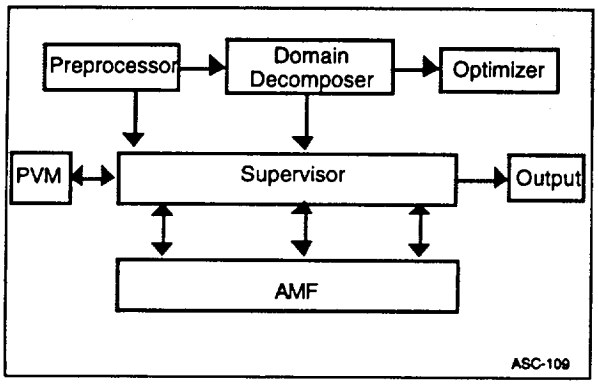


Figure 6-12. Flow Diagram of MAESTRO

Table 6-4. The Four Processors' Activity After Optimization on nCube-1024

Proc No. 1	Proc. No. 2	Proc No. 3	Proc No. 4
MP&Pr 8	MP&Pr 11	MP&Pr 14	MP&Pr 12
Condn 8	Condn 11	Condn 14	Condn 12
MP&R 9	MP&R 10	MP&R 15	MP&R 13
Condn 9	Condn 10	Condn 15	Condn 13
Assem 4	Assem 5	Assem 7	Assem 6
Condn 4	Condn 5	Condn 7	Condn 6
WT 2 S 4	RD 1 S 4	WT 4 S 7	RD 3 S 7
.....	Assem 2	.....	Assem 3
.....	Condn 2	.....	Condn 3
.....	RD 4 S 3	.....	WT 2 S 3
.....	Assem 1	.....	.....
.....	Condn 1	.....	.....
END OF CLIMBING			
.....	Solv TOP	.....	.....
.....	WT 4 S 1	.....	RD 2 S 1
.....	Solvx 2	.....	Solvx 3
RD 2 S 2	WT 1 S 2	RD 4 S 3	WT 3 S 3
Solvx 4	Solvx 5	Solvx 7	Solvx 6
Solvx 9	Solvx 10	Solvx 15	Solvx 13
Solvx 8	Solvx 11	Solvx 14	Solvx 12

ASC-087

Table 6-5. The Four Processors' Activity Before on nCube2-1024

Proc No. 1	Proc. No. 2	Proc No. 3	Proc No. 4
MP & PR12	MP&Pr 13	MP & Pr 14	MP & Pr15
Cond 12	Cond 13	Cond 14	Cond 15
MP&Pr 8	MP&Pr 9	MP&Pr 10	MP&Pr11
Cond 8	Cond 9	Cond 10	Cond 11
.....	WT 3 S13	RD 2 S13	.....
.....	RD 3 S10	WT 2 S10	.....
WT 3 S12	.....	RD 1 S12	.....
.....	.....	WT 4 S14	RD 3 S14
RD 2 S 9	WT 1 S 9	.....	.....
.....	RD 4 S11	.....	WT 2 S11
Assem 4	Assem 5	Assem 6	Assem 7
Cond 4	Cond 5	Cond 6	Cond 7
.....	RD 4 S 7	.....	WT 2 S 7
.....	RD 3 S 6	WT 2 S 6	.....
RD 2 S 5	WT 1 S 5	.....	.....
Assem 2	Assem 3	.....	.....
Cond 2	Cond 3	.....	.....
RD 2 S 3	WT 1 S 3	.....	.....
Assem 1	.....	.....	.....
Cond 1	.....	.....	.....
END OF CLIMBING			
Solv TOP	.....	.....	.....
WT 2 S 1	RD 1 S 1	.....	.....
Solvx 2	Solvx 3	.....	.....
WT 2 S 2	RD 1 S 2	.....	.....
.....	WT 3 S 3	RD 2 S 3	.....
.....	WT 4 S 3	.....	RD 2 S 3
Solvx 4	Solvx 5	Solvx 6	Solvx 7
.....	WT 4 S 3	.....	RD 2 S 5
WT 2 S 4	RD 1 S 4	.....	.....
.....	.....	RD 4 S 7	WT 3 S 7
RD 3 S 6	.....	WT 1 S 6	.....
.....	WT 3 S 5	RD 2 S 5	.....
.....	RD 3 S 6	WT 2 S 6	.....
Solvx 8	Solvx 9	Solvx 10	Solvx 11
Solvx 12	Solvx 13	Solvx 14	Solvx 15

Figure 6-13 shows detailed subroutines of MAESTRO. The duties of these subroutines are briefly described in Table 6-6. Table 6-4 illustrates the Task Allocation Table (TAT) after the optimization. TAT creates the "optout.data." The "optout.data" is the result of optimization task . This file can continuously be updated if an iterative solution is performed.

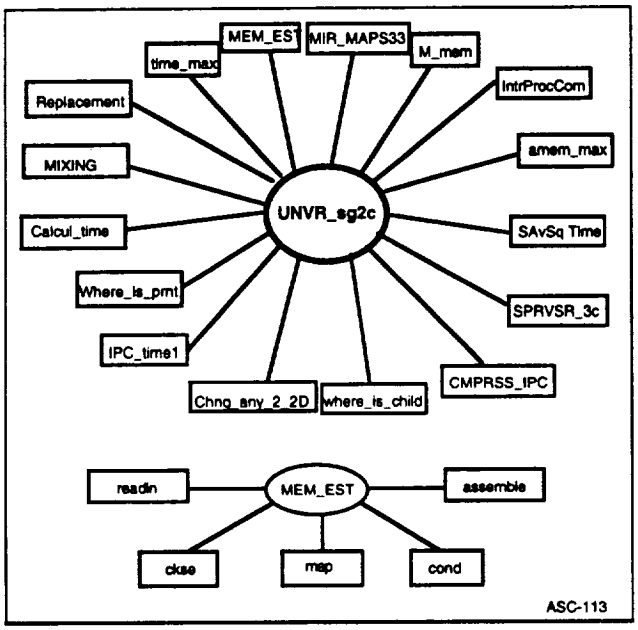


Figure 6-13. Subroutines of MAESTRO

Table 6-6. Multi Factor Optimization Subroutines and Their Functionality

MAESTRO
The MAESTRO contains the following subroutines:
MIR_MAPS33: creates and initiates the table of interprocess communication, the table of task distribution, and so on.
M_mem: estimates the memory usage of each processor.
IPC_time1: estimates the interprocessor communication time.
Calcul_time: estimates the time usage for each processor.
Replacement: reorganize Map_use if there are more super element than processors.
Where_is_prnt: determines the parent of some separate group.
Where_is_chld: determines the child of some separate group.
CMPRSS_IPC: changes the INTERCOM_strg (IPC) table.
IntrProcCom: initiates the table for interprocessor communication.
SPRVSR_3c: simulates the running of supervisor program.
time_max: returns the maximum usage time between processors.
amem_max: returns the maximum memory between processors
MIXING: mixes Map_use using some random generator.
SAvSqTime: evaluates function of time function depending on number of the functional.
Chag_any_2_2D: returns some value depending on the position of the separator group in the tree.
• **MEM-est:** estimates the memory usage by running the following:
• 1. **readin:** reads the input files test.data.
• 2. **map:** simulates the subroutine map of AMF.
• 3. **cond:** simulates the subroutine cond of AMF.
• 4. **assemble:** simulates the subroutine assemble of AMF.
• 5. **ckse** simulates the subroutine ckse of AMF.

6.4.1 Domain Decomposer

The domain decomposer module is responsible for partitioning a finite element model within MAESTRO. It cuts the model into many pieces. These pieces can be processed in parallel on different processors. The SUPERVISOR receives these pieces and distributes them as tasks to assigned processors. The domain decomposition is performed by the RIP and PEEL algorithms. These algorithms exploit the discretization of the FEM and perform balanced partitioning and natural subdivision of the grid topology to minimize the computational bottleneck. The PEEL algorithm performs natural substructuring of the model based on geometric topology. It identifies the exterior boundary nodes, finds all the attached elements to the exterior nodes and peels them off to produce super-elements. The RIP algorithm is applied to further subdivide the model. It first identifies the smallest moment of inertia in the long direction, minimizes the boundary region by rotation, and finally balances processor computational loading. The RIP algorithm follows these general steps to partition a finite element model into substructures, S_i. Each S_i is a subset of nodes and elements from the original model. As a requirement of the multifrontal solution algorithm, use of RIP continues until partitioning of the substructures is not possible. This limit occurs when RIP cannot find moment of inertia axes. Table 6-7 illustrates the input and output of the domain decomposer. Table 6-8 illustrates the "test.data" file which contains all the logical binary tree resultant information.

Table 6-7. DOMAIN DECOMPOSER Input And Output

DOMAIN DECOMPOSER
• **Input:** The input of the DOMAIN DECOMPOSER is the • PATRAN Neutral File.
• **Output:** The output of the DOMAIN DECOMPOSER is • **test.data** which is one of the inputs to SUPERVISOR.

Table 6-8. Logical Binary Tree Information File

test.data
This file is result of Domain Decomposition and contains entirely positive format free integers. The first line contains the following six numbers: number of nodes, *non* number of finite element, *noe* number of nodes per finite element, *nonpe* number of super element, *nose* number of separator groups, *nosg* number of levels of super elements, *nolse*
After the first line, there are the following three blocks: • This block contains two lines for each finite element (2 x *noe* lines together). The first line contains the element number, super-element number containing the element, and number of nodes in the finite element. The node numbers contained in the finite element are listed in the second line. • The next block of data consists of a pair of lines for each separator group. The first line of the pair contains separator group number, level number, separator group number for parent, separator group number for left child, separator group number for right child, the corresponding lowest level super element number (0, if does not belong to the lowest level), and number of nodes in the separator group. The nodes in the separator group, if any, are listed on the second line of the pair. This second line must be skipped when there is no nodes in the separator group. • The last block of data consists of a pair of lines for each super-element. The first line of the pair contains super-element number, separator group number for the super-element, and number of elements in the super-element. The elements in the super-element are listed on the second line of the pair.

6.4.2 Alpha Star Multifrontal Algorithm (AMF)

AMF is introduced as a means of solving finite element systems of equations that are too large to reside in the main memory of a computer. AMF reduces the global memory by reducing the number of the global finite element equations. Moreover, it determines the order of eliminating unknowns within the solution. AMF consists of 20 FORTRAN files, two input files, and one parameter file. The parameters

(paramh.h) contains the maximum value of important parameters to be used in memory allocation task. For an efficient run these maximum values should be set to their actual values. These values are described in Table 6-9. As shown in Figure 6-4, AMF assembles, decomposes, and solve system of equations. The output of AMF is the displacements at each node. The displacement output can be used in postprocessing of the adjoin finite element program to calculate strains and stresses.

Table 6-9. Parameters and Subroutines of Alpha Star Multi Frontal Algorithm (AMF)

AMF	
Parameters	
KU	a positive integer (this integer just shifts the integer number used in naming direct files)
MNONPE	maximum number of nodes per element
MNDOFPN	maximum degree of freedom of a node
MNOE	maximum number of elements
MNON	maximum number of nodes
MNBC	maximum number of boundary conditions
MNL	maximum number of levels in binary tree
MNOSE	maximum number of super elements
MNOSG	maximum number of separator groups
MNEISE	maximum number of elements in a super element
MNONISG	maximum number of nodes in a separator group
MAX	maximum number of nodes in a super element
AMF Subroutines and their functionality:	
1.	**mapparass:** This subroutine basically combines *map and parass* by calling them and collecting the inputs from and outputs.Mapparass assists in parallelization of AMF task.
2.	**map**: The subroutine *map* creates local renumbering of the nodes of a given super element. The mapping between local nodes of the super element and global nodes of the finite element mesh is evaluated and stored in Lr array in this subroutine.
3.	**parass:** The assembly of all elements in a given super element is created in *parass*. The assembly is performed element by element.
4.	**cond:** This subroutine separates interior nodes and applies boundary conditions to these nodes. It also calls *decompamf* to decompose the stiffness matrices.
5.	**rc_int:** This subroutine modifies a given stiffness matrix when two nodes are interchanged. The modification is done by interchanging the entire block of rows associated with the node numbers. Accordingly, the entire block of columns are interchanged..
6.	**BC**: This subroutine applies a boundary condition to a given stiffness matrix. This is done by resetting the appropriate rows and columns of the matrix equations.
7.	**decompamf:** This subroutine performs triangular decomposition of a given stiffness matrix. It also performs all the necessary matrix multiplication's needed during descend from the binary tree.
8.	**assembling:** This subroutine assembles a pair of given child of super elements into a parent super element by calling *asseml* and *assemr*.
9.	**assemI:** This subroutine accounts for the contribution of left hand side of right child stiffness equations to the assembly of the parent stiffness equations.
10.	**assemr**: This subroutine accounts for the contribution of right hand side of right child stiffness equations to the assembly of the parent stiffness equations.
11.	**solutop**: This subroutine performs the solution at the top of binary tree. The *solutop is* actually a collection of four subroutines *appforce, appbc, decompamf;* and *solve* which are called back to back to solve the displacement equations at the top.
12.	**appforce:** This subroutine applies external forces to interior stiffness equations.
13.	**appbc:** This subroutine applies boundary conditions to interior stiffness equations.
14.	**solve:** This e subroutine solves systems of equations using forward reduction on the columns of a decomposed matrix.
15.	**reg solution:** This subroutine prepares the system of equations to be solved as the binary tree is descended. This subroutine is not used at the top of the tree.

6.4.3 SUPERVISOR

Supervisor operation is based on dynamic monitoring, controlling, synchronizing and distributing the work load among available computer resources (Table 6-10). SUPERVISOR initiates and spawns predefined numbers of processes to predefined numbers of (virtual or real) processors. SUPERVISOR receives the input files "test.data" (Table 6-8), "testpt.data" (Table 6-11), a_a_f (dynamic service allocation file), and "optout.data" from PREPROCESSOR, DOMAIN DECOMPOSER, and OPTIMIZER. SUPERVISOR manages execution of tasks between processors, data storage, and

6.4.3 SUPERVISOR

Supervisor operation is based on dynamic monitoring, controlling, synchronizing and distributing the work load among available computer resources (Table 6-10). SUPERVISOR initiates and spawns predefined numbers of processes to predefined numbers of (virtual or real) processors. SUPERVISOR receives the input files "test.data" (Table 6-8), "testpt.data" (Table 6-11), a_a_f (dynamic service allocation file), and "optout.data" from PREPROCESSOR, DOMAIN DECOMPOSER, and OPTIMIZER. SUPERVISOR manages execution of tasks between processors, data storage, and interprocessor communications. The management and decision making is based on the information from the Task Allocation Table matrices (optimization results). SUPERVISOR communicates with other algorithms and libraries such PVM, and AMF to conduct simulation for MAESTRO.

Table 6-10. SUPERVISOR Subroutines and Their Responsibilities

SUPERVISOR	
1.	**sprvsr4:** controls the distribution of logical parts of the binary tree of operation among the virtual processors. according to the intercommunication optimization and matrices of TAT. Realization of AMF algorithm based on AMF library. li
2.	**nREAD:** receives needed logical binary tree information from other processors by PVM (receiving, unpacking, and synchronizing routines)
3.	**nWRITE:** sends needed logical binary tree information to other processors by PVM (sending, packing, and synchronizing routines).
4.	**takingl:** takes intermediate information from it's own processor memory (condensed part of the stiffness matrices from the child binary tree node storage stack)..
5.	**taking2:** takes intermediate information from it's own processor memory (full stiffness matrices and right hand side from the parents binary tree node storage stack to the child node for the back substitution solution).
6.	**saving1**: stores in it's own processor memory stack (intermediate stiffness matrices and their attributes after assembly and condensation for the particular nodes of binary tree of parallelization).
7.	**Top _Solver**: decomposition and solving for the separation group # 1 on the top of the binary tree operation. Creates vector of displacement Y for the nodes of first separator group.
8.	**Reg _Solver**: solution by back substitution for the nodes of the binary tree operation
9.	**saving_y:** storage in the it's own processor memory the resultant Y displacement vector from the parents nodes. of the binary tree operation
10.	**taking_y:** takes vector of Y displacement from the storage stack for the particular node during the back substitution operation.
11.	**pvmamf:** checks how many processors should work in parallel to solve the problem. It creates as many tasks as there are available processors.

Table 6-11. Testpt.Data Functionality and Responsibility

testpt.data
This file contains: 1) boundary conditions, 2) element stiffness matrices, and 3) global force such as follows:
The first line contains only five integers. These are:
number of boundary condition, nobc *degree of freedom per node, dofpn* *number of nodes, non* *number of element, noe* *number of nodes per element, nonpe*
There are three blocks after the first line:
This block contains *nobc* lines (geometric boundary conditions). Each line contains the following three numbers:
1-the node number, 2-the degree of freedom, 3-the value of the prescribed boundary condition
This block consists of element stiffness matrices (row by row) with maximum five numbers on a line. These are *non* subblocks. There is a subblock for each finite element. each of this subblock consists of $(nonpe \times dofpn)^2$ numbers. The data for each new element starts from a new line. The last block consists of the global force vector using maximum of the five numbers on a line. There are ($non \times dofpn$) numbers.

6.4.4 Parallel Virtual Machine

PVM is a software system that enables a collection of heterogeneous computers to be used as a coherent and flexible concurrent computational resource. The individual computers may be shared-, or local-, memory multiprocessors, vector supercomputers, specialized graphics engines, or scalar workstations, that may be interconnected by a variety of networks, such as ethernet or FDDI. The MAESTRO accesses PVM through library routines to parallelize the computation of a model. Moreover, the MAESTRO uses the intercommunication facilities of PVM to create and synchronize the tasks

6.4.5 Multifactor Optimization

Multifactor optimization is formulated as an integer problem (optimization of the binary tree of the parallel operations) for which minimization of CPU time subject to the parallelism constraints is sought. The dependent and independent design parameters are defined as 1) number of demanded processors, 2) requested processor memory, 3) inter processor communication time, 4) number of the divisions of the model (superelements) , 5) TAT(2D matrix of the particular tasks distribution), and 6) processor sleeping time. The basic formulation of the optimization routine of MAESTRO is minimization of the maximum execution time among the available processors.,

The flowchart of the multifactor optimization program is the realization of the functional algorithm. Figures 6-14, and 6-15 illustrate the optimization flow diagram, the optimizer 1) takes the average of time of operations (integer, real, and logical) for each particular processor for the available set; 2) determines the necessary information about the structure of the binary tree of operation; 3) builds a precalculated number of the operation for each node of the binary tree; 4) after the reading information from the disk, starting an interactive dialog with the user (sizes of the memory constraints, number of the maximum global operation for the process and number of the available processor).

The Mem_EST subroutine calculates sizes of the stored stiffness matrices for the nodes of the binary tree, and OPTIMIZER creates the vector of memory distribution among the separation group for evaluation of the memory constraint.

Block B in Figure 6-14 and Table 6-12 analyzes the binary tree information and creates set of shell matrices maps for relative processing of the hierahical information topology tree such as: Map_use, Map_call, Map_tree, Map_wrt, etc. Based on the 2D mapping tree information and intermediate task distribution this block creates vectors of memory and time distribution among the processors and builds up the 3D matrix of the interprocessor communication.

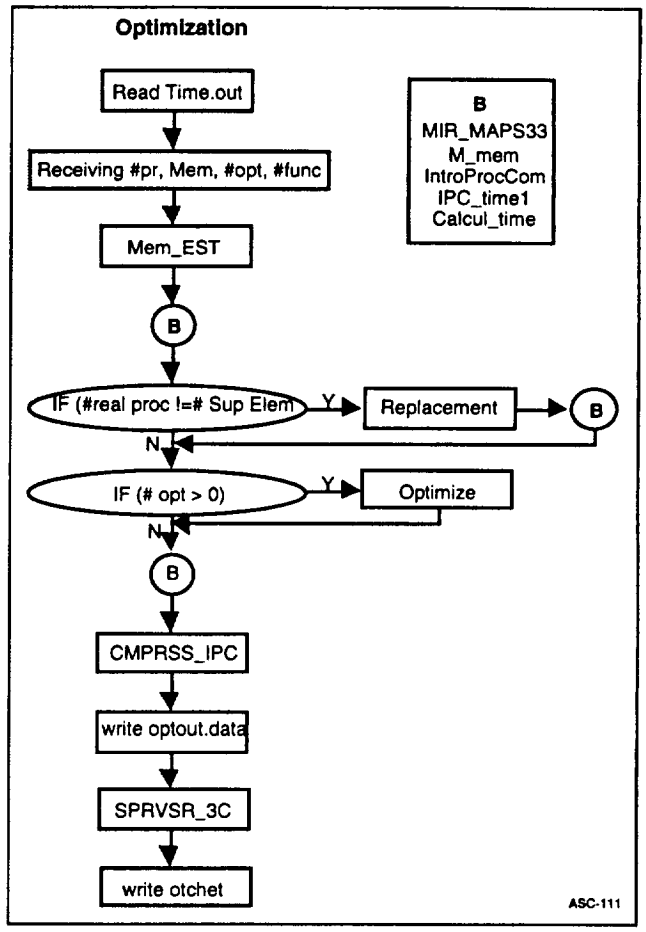


Figure 6-14. Flow Diagram of Multifactor Optimization in MAESTRO

Figure 6-14 illustrates the flow diagram of the concurrent computing "MAESTRO" multifactor optimization. The Concurrent computing environment is achieved by: 1) effective minimization of the interprocessor communication (time and memory), as shown in Figures 6-16, and 6-17, and 2) resource management of multiprocessing by utilization of the testing the facility usage (TFU) program, as shown in Figure 6-2. Table 6-12 illustrates the Testing Facility Usage (TFU) subroutines and their Functionality. The output of MAESTRO optimzer is illustrated in Table 6-13.

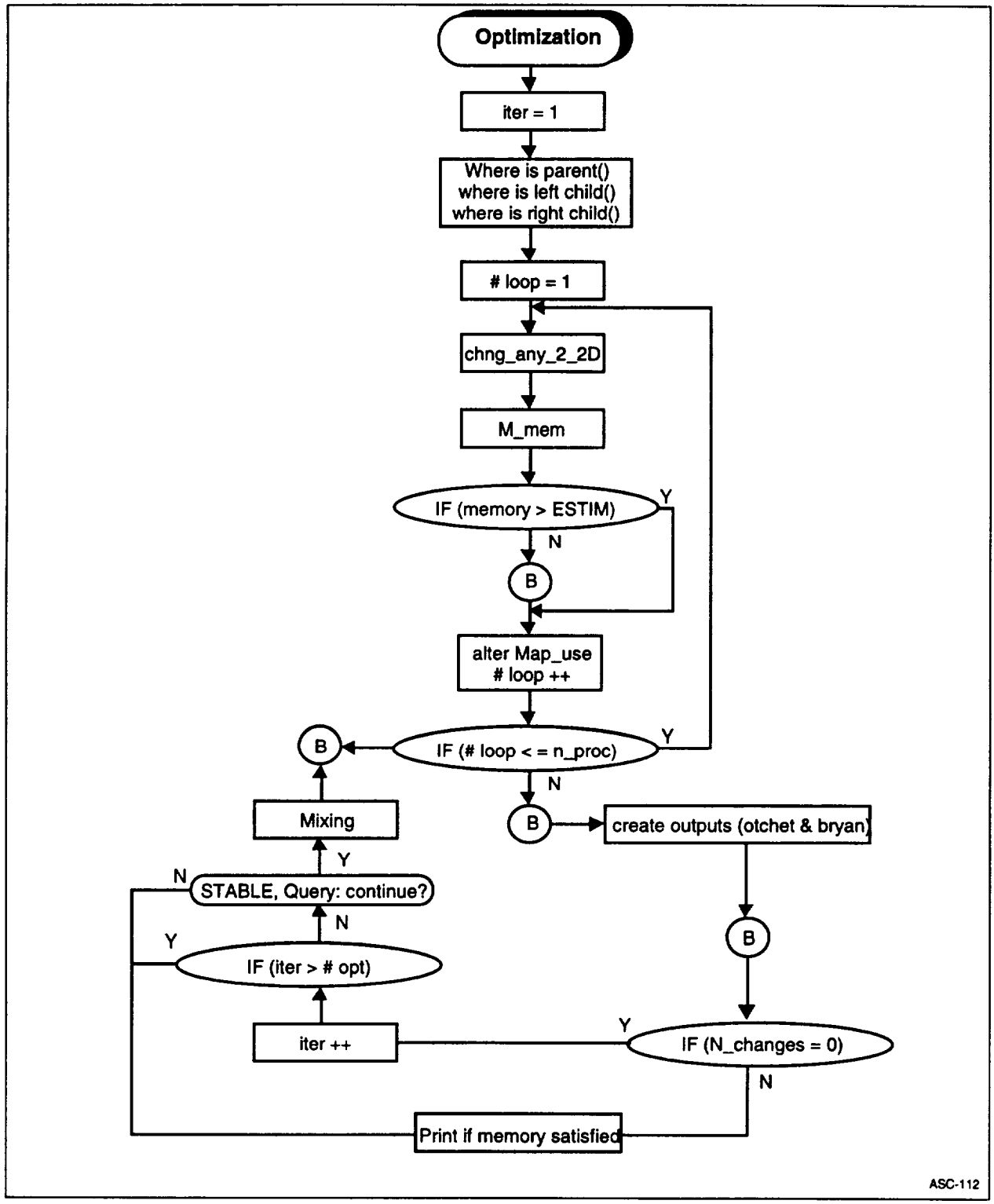


Figure 6-15. Flowchart of Multifactor Optimization

Table 6-12. Testing Facility Usage (TFU) Subroutines and Their Functionality

TFU:
Inter Processor Communication (IPC) Testing
1. **RprocN:** 1D array of the CHAR*8 of real hosts names. 2. **1ofwrd**: the length of each word of the message. 3. **nofwrds:** size of the message (words). 4. **N_proc1:** logical number of the processor which is writing. 5. **N_proc2:** logical number of the processor which is reading. 6. **n_atmpts:** number of the communication attempts. 7. **timel:** mean time for nofwrds portion after n_atmpts investigations. 8. **avrgtime:** the same relating to 1 byte (msec).
Timing of the processor's capacity
1. **RprocN:** 1D array of the CHAR*8 of real hosts names (the same as in 1). 2. **nf1top:** number of the floating operations for testing. 3. **LnRproc:** logical number of the particular host, corresponding with Rp. 4. **n_atmpts:** number of the experiments. 5. **TMFLOT:** adduced to one floating operations time of the testing (msec).
Function TMINTGR
1. **RprocN:** 1D array of the CHAR*8 of real hosts names (the same as in 1). 2. **nf1top:** number of the floating operations for testing. 3. **LnRproc:** logical number of the particular host, corresponding with Rp. 4. **n_atmpts:** number of the experiments. 5. **TMINTGR:** adduced to oneinteger operations time of the testing (msec).
IPC between virtual processors:
1. **RprocN:** 1D array of the CHAR*8 of real hosts names. 2. **1ofwrd**: the length of each word of the message. 3. **nofwrds:** size of the message (words). 4. **N_proc1:** logical number of the processor which is writing. 5. **N_proc2:** logical number of the processor which is reading. 6. **n_atmpts:** number of the communication attempts. 7. **timel:** mean time for nofwrds portion after n_atmpts investigations. 8. **avrgtime:** the same relating to 1 byte (msec).
Algorithm's steps
1. **NRproc:** number of the real processors. 2. **Nvproc**: number of the virtual processor. 3. **NSTGWRS:** number of the words which are stored on the disk or taken f. 4. **n_atmpts:** number of the. 5. **Tio:** meantime for the input/output operations with particular d. 6. **IODtim:** average time of the file positioning and settlement (open).

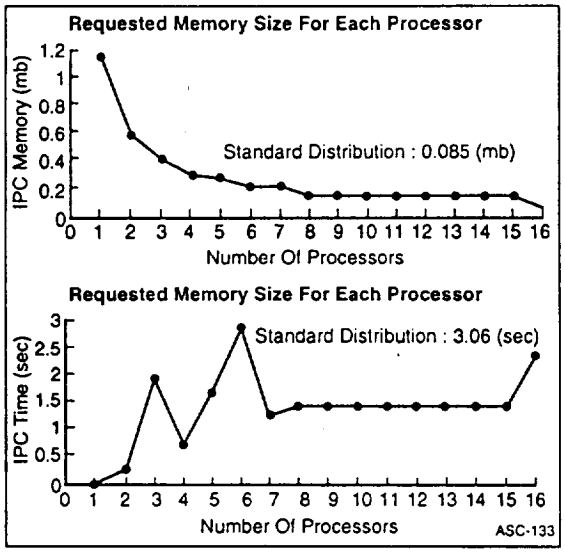


Figure 6-16. Minimized Number of Interprocessor Communication (Time and Memory) for 16 Processors

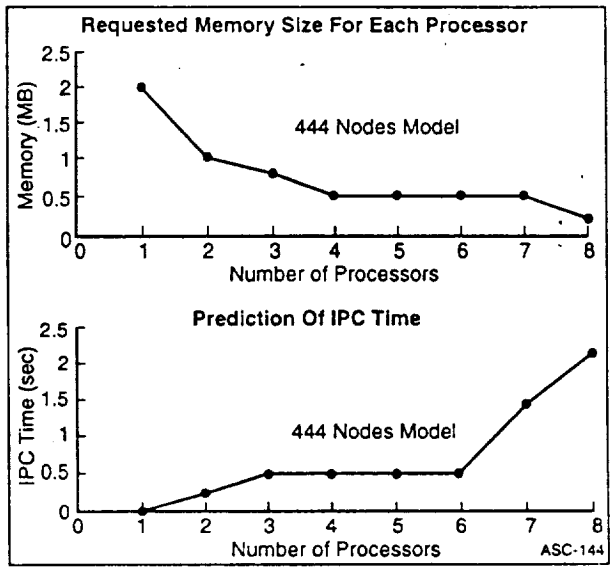


Figure 6-17. Minimized Number of Interprocessor Communication (Time and Memory) for 8 Processors

Table 6-13. The Output File of the MAESTRO Optimizer.

optout.data
This file is the result of the optimization. The fist row of this file contains 2 integers. The first integer L is the number of Logical Level (computed in optimization phase). The second number P is the number of the processors.
Furthermore, there are the following two blocks in this file after the first line:
• The first block contains L subblocks. The first line of each subblock contains only one positive integer. This number tells how many other lines are expected in this subblock. This number is 0 if no more line are expected. Each of these lines contain P integers. • The second block contains L lines and each line contains P integers

6.5 REFERENCES

6.1. R. H. Sues and G. S. Rhodes, "Portable Parallel Stochastic Optimization for Design of Aeropropulsion Components," *NAS3-26839*, October 1994.

6.2. F. F. Abdi, Y. Mirvis, C. Chamis, and P. Murthy, "Large-Scale System Optimization by Real Time Parallelization Algorithm Martin ," NASA Computational Aeroscience Workshop, March 1995.

6.3. R. H. Sues, H. C. Chen, A. L. Twisdale and W. R. Martin "Probabilistic Structural Mechanics Research for Parallel Processing Computers," *NAS3-25824*, August 1991.

6.4. F. F. Abdi, Y. Mirvis, and P. Murthy, "A Hierarchial Multilevel Optimization Solution for Massively Parallel Simulation of Composite Systems, 36 AIAA/ASCE/AHS/ASC SDM Conference, New Orleans, Louisiana, April 1995

6.5. R. Calalo, B. Nour-Omid, and M. Wright "An Implementation of the Multifrontal Solution Algorithm on MIMD Computers," AIAA, pp. 1-9, 1992

6.6. J. H. Wilkinson and Reinsch, Linear Algebra. Handbook for Automatic Computation, II, Springer-Verlag, 1971.

6.7. A. Ralston, "A First Course in Numerical Analysis," McGraw-Hill, 1965.

6.8. L. Fox, "An Introduction to Numerical Linear Algebra," Oxford Univ. Press, 1965.

7.0 Stochastic Structural Analysis Methods For Composites

Current methods of structural and materials probabilistic analysis tend to fall into three primary categories: simulation, perturbation and reliability methods. Each of these categories have specific areas in which their application is best fit. In some cases they can be implemented concurrently to obtain a desired analytical capability. For purposes related to the development of the integrated software package, we have narrowed the scope of this study to concentrate on reliability methods for structural analysis using distribution estimation for micro level uncertainty.

This level of analysis utilizes a micromechanics approach to evaluate the stochastic variation of constituent material properties and strengths subjected to primitive variables.

For probabilistic analysis on the micro-level, some preliminary capabilities have been demonstrated which implement distribution estimation and simulation. Boyce, et. al. [7.1, 7.2] have developed a code known as PROMISS (Probabilistic Material Strength Simulator). Their approach centers on the application of a general phenomenological constitutive relationship for composite materials utilizing maximum likelihood estimations as the basis for stochastic variation of constituent properties subjected to various primitive variables. This technique presents a unique opportunity for implementation in a concurrent environment. As presented in Section 3.4, these computational developments has been integrated with HITCAN which are uniquely capable of providing the probabilistic micromechanics analysis needed for the assessment of uncertainties at the micro-level of a finite element nodal components.

7.1 AN INTEGRATED MODULARIZED CONSTITUENT MATERIAL STRENGTH AND RISK ASSESSMENT ANALYSIS

The assessment of reliability distribution of material constitutive strength degradation is performed by PROMISS code. Metal Matrix Composite (MMC), Ceramic Matrix Composite (CMC), and Polymer Matrix Composite (PMC) constituent material strength and risk assessment analysis for each ply at each node following every load increment is performed by integration of METCAN [7.3], CEMCAN [7.4], and ICAN [7.5] modules with Probabilistic Material Strength Simulation. (PROMISS) code. PROMISS is integrated with these modules to perform probabilistic materials strength simulation in place of the deterministic form of the multifactor law for the fiber, interface and matrix. This integration is performed with inputs, outputs of constituent material analysis (METCAN, CEMCAN, and ICAN), as well as the associated data bank of these programs.

Material property data is passed through from the data bank to provide the current, ultimate, reference and empirical constant values. The form of the constitutive strength degradation equation used for the composite analysis is given as:

$$\frac{P}{P_o} = \left[\frac{S_{SF} - \sigma_S}{S_{SF} - \sigma_{So}}\right]^{a1} \left[\frac{\dot{S}_{SF} - \dot{\sigma}_{So}}{\dot{S}_{SF} - \dot{\sigma}_S}\right]^{a2} \left[\frac{T_F - T}{T_F - T_o}\right]^{a3} \left[\frac{\dot{T}_F - \dot{T}}{\dot{T}_F - \dot{T}_o}\right]^{a4}$$

Within each primitive variable term the current, ultimate and reference values and the empirical material constant may be modeled as either, normal, lognormal, or Wiebull random variables. Simulation is used to generate a set of realizations for normalized random strength, S/S_o, from a set of realizations for primitive variables and empirical material constants. Maximum penalized likelihood

is used to generate an estimate for the PDF of normalized strength, from a set of realizations of normalized strength. Integration of the PDF yields the CDF. Plot files are produced to plot both the PDF and the CDF. PROMISS also provides information on S/S_O statistics (mean, variance, standard deviation and coefficient of variation), Figure 7-1 illustrates the procedure in performing material reliability assessment.

A graphic user interface (GUI) is created for integration of PROMISS with analytical results of METCAN, CEMCAN, and ICAN. The GUI provides the user the following capabilities: 1) Interactive preparation of input data for PROMISS from material constituent analysis (METCAN, ICAN, CEMCAN) unit cell output stress graph, 2) Interactive selection of the ply, subregion, within unit cell, time step, and the stress direction to perform probabilistic failure analysis, 3) graphically display the failure probability distributions, i.e. PDF, and CDF. Procedure to evaluate the constituent material and risk assessment for strengths MMC, CMC, and PMC material systems is depicted in Table 7-1

Many uncertainties in the constituent properties such as fiber, matrix, Interface, and fabrication variables can be simulated . These constituent properties are illustrated in Table 7-2

7.1.1 PROMISS Integrated With METCAN

PROMISS integration with METCAN module provides the capability to perform probabilistic materials strength simulation in place of the deterministic form of the multifactor law for the fiber, interface and matrix which is performed by the subroutines MECH and MECF. Evaluation of the MMC constituent material strengths and risk assessment is performed by new subroutine METPRO. METPRO extracts information (Table 7-1) from METCAN data bank, and METCAN simulation output at each time increment for each ply in matrix at each node following every load increment.

Our efforts focused on integrating probabilistic materials strength simulation for the following stress regions and subregions

1) Matrix strength at time t_i for region A in the 11, 22, 12, 23, 13, 33 directions (S_{m11A}, S_{m22A}, S_{m12A}, S_{m23A}, S_{m13A}, S_{m33A})
2) Matrix strength at time t_i for region B in the 11, 22, 12, 23, 13, 33 directions (S_{m11B}, S_{m22B}, S_{m12B}, S_{m23B}, S_{m13B}, S_{m33B})
3) Matrix strength at time t_i for region C in the 11, 22, 12, 23, 13, 33 directions (S_{m11C}, S_{m22C}, S_{m12C}, S_{m23C}, S_{m13C}, S_{m33C})
4) Interface strength at time t_i for region Bin the 11, 22, 12, 23, 13, 33 directions (S_{d11B}, S_{d22B} ,S_{d12B} ,S_{d23B} ,S_{d13B}, S_{d33B}),
5) Interface strength at time t_i for region C in the 11, 22, 12, 23, 13, 33 directions (S_{d22C}, S_{d23C}, S_{d12C}, S_{d13C}, S_{d11C}, S_{d33C}),
6) Fiber strength for region C in the 11, 22, 12, 23, 13, 33 directions (S_{f11}, S_{f22}, S_{f12} , S_{f23}, S_{f13} , S_{f33}),

7.1.2 PROMISS Integrated With ICAN

PROMISS integration with the ICAN module provides the capability to perform probabilistic materials strength simulation in place of the deterministic form of the multifactor law for the fiber, interface and matrix which is performed by the subroutines MECH and MECF. Evaluation of the PMC constituent material strengths and risk assessment is performed by new subroutine IPRO. IPRO extracts information (Table 7-1) from ICAN data bank, and ICAN simulation output for primary and secondary composite system.

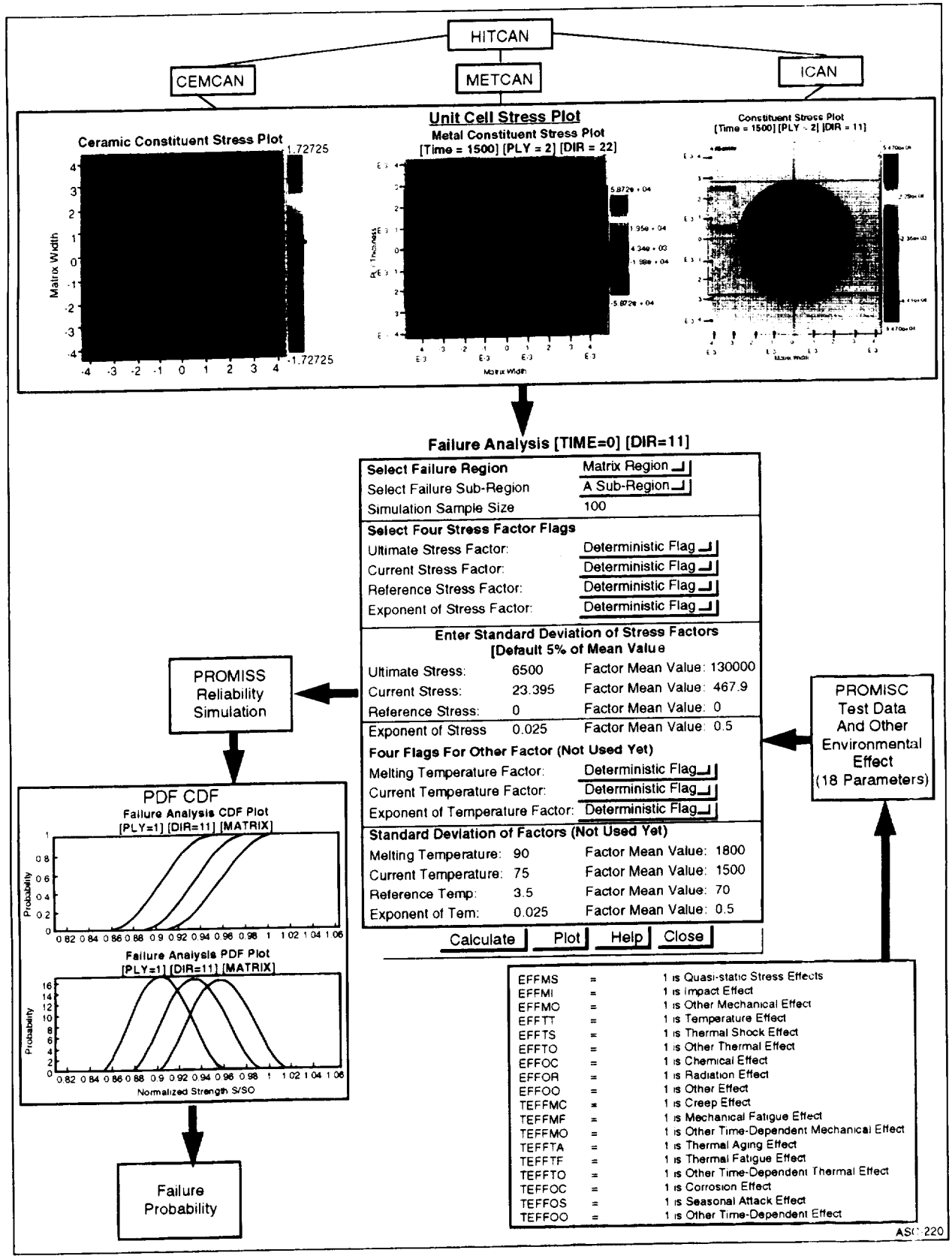


Figure 7-1. Procedure For Assessment of Material Reliability

Table 7-1. Procedure of Interactive Material Risk Assessment

READ Constituent DATA for failure assessment From GRAPHICS (GUI)
STEP 1: A-Select time step; B-Select Ply Number for failure assessment; C-Select the probability of failure (fiber, or Matrix, or Interface) due to stress in "11", "22", "33", "12", "13", "23"
STEP 2: Select deterministic variable,or random Variable with normal distribution, or random variable with lognormal distribution, or random variable with Weibull distribution from GUI
STEP 3: Program to Calculate or USER Input: STANDARD DEVIATION of Selected four primitive variables (A_i, A_{iF},and A_{io})and ai the empirical material constant from GUI
READ DATA From Constituent DATA BANK
STEP 4: Read Material data from METCAN.IN file; Extract Material data for Matrix, Fiber, and Interface; Read Melting Temperature of fiber, and Matrix; Read Fiber Tensile and Shear Strength; Read Exponent of stress and temperature
READ STORED DATA From Constituent SIMULATION
STEP 5: Read fiber stress for each subregion [C]; Read MATRIX stress and temperature for each subregion [A or B or C]; Read INTERFACE stress and temperature for each subregion [B or C]
X11/MOTIF POST PROCESSOR PLOTS
STEP 6: Prepare data input for PROMISS, Execute PROMISS, Generate PDF, CDF; Generate Post Processor PDF, and CDF Graphics and save the plots for STEP 7.
STEP 7: Repeat step 1 through 3 to perform sensitivity analysis; Re Generate Post Processor PDF, and CDF Graphics. Create failure probability plots

Table 7-2 Uncertainties In The Constituent Properties

FIBER	MATRIX	FABRICATION VARIABLE
Normal Modulus E_{f11}	Normal ModulusE_m	Fiber volume ratio, K_f
Normal Modulus E_{f22}	Poisson's ratio, v_m	Void volume ratio, k_v
Poisson's ratio, v_{12}	Tensile Strength, S_{mT}	Ply Thickness, t_i
Poisson's ratio, v_{23}	Compressive Strength, S_{mC}	Orientation angle θ
Shear Modulus, G_{f12}	Shear Strength, S_{mS}	
Shear Modulus, G_{f23}		
Tensile Strength, S_{fT}		
Compressive Strength, S_{fC}		

Our efforts focused on integrating probabilistic materials strength simulation for the following stress regions and subregions

1) 1) Matrix strength for region A in the 21, 22, 31, 32, 12, 13 directions (S_{m21A}, S_{m22A}, S_{m31A}, S_{m32A}, S_{m12A}, S_{m13A})
2) Matrix strength for region B in the 21, 22, 31, 32, 12, 13 directions. (S_{m21B}, S_{m22B}, S_{m31B}, S_{m32B}, S_{m12B}, S_{m13B})
3) Fiber strength for region B in the 21, 22, 31, 32, 12, 13 directions (S_{f21B}, S_{f22B}, S_{f31B} , S_{f32B}, S_{f12B} , S_{f13B}),

7.1.3 PROMISS Integrated With CEMCAN

PROMISS integration with the CEMCAN module provides the capability to perform probabilistic materials strength simulation in place of the deterministic form of the multifactor law for the fiber,

interface and matrix which is performed by the subroutines MECH and MECF. Evaluation of the CMC constituent material strengths and risk assessment is performed by new subroutine CEMPRO. CEMPRO extracts information (Table 7-1) from the CEMCAN data bank, and the CEMCAN simulation output at each time increment for each ply in matrix at each node following every load increment.

Our efforts focused on integrating probabilistic materials strength simulation for the following stress regions: 1) ply stress, 2) individual slice stress, 3) matrix stress, 4) fiber stress, and 5) interface stress in 11, 22, 33, 12, 13, and 23 directions.

7.1.4 Integration of PROMISS With HITCAN

The programmatic integration of PROMISS capabilities with the HITCAN code are principally directed at the micromechanics analysis performed by the sub-program METCAN.

As the nodal stresses are passed from NESSUS to METCAN, a multilevel micromechanical analysis of the composite behavior is initiated. The first level is purely programmatic, initializing dynamic memory allocation and file control. The actual evaluation of the composite response begins at the second level (henceforth referred to as the ply-level) residing in the subroutine METCAN. This subroutine performs several vital functions in the analysis of the composite behavior which include:

1) Reading the nodal stress state determined by the structural finite element analysis;
2) Loading the nodal through thickness composite material description;
3) Initialization of material properties;
4) Performing laminate theory analysis to determine composite thermal and mechanical response;
5) Calculation of ply properties and strengths using the unique set of micromechanical equations for metal matrix composites;
6) Decomposition of composite stresses and transformation to ply tensor stresses;
7) Evaluation of composite level convergence using a mechanical equilibrium criteria.

A detailed derivation of these equations is presented in Reference 7.6. In order to provide reference for the probabilistic application of the properties incorporated in these equations, we have restated the resulting formulation in Tables 7-3 through 7-5.

The third level of analysis (henceforth referred to as the constituent-level) is driven by the subroutine PLYMAT. This subroutine has the multifaceted task of performing micromechanical analysis for each ply of the composite. It decomposes and calculates the constituent stresses, as well as degraded strengths and properties, and then synthesizes the constituent response to evaluate ply level convergence.

By integrating a modified form of the PROMISS code with the calls made from the subroutines STRGM and STRGF (shown in Figure 7-2), a probability density function (PDF) is generated for the selected strengths for each ply at user defined nodes and time steps. A cumulative distribution function (CDF) is also calculated from the integration of the PDF with each being sent to a post-processing file by the subroutines CDFOUT and PDFOUT. The subroutine METPRO sets up the input parameters for performing the PROMISS analysis. Material property data is passed through this routine from METCAN data bank to provide the current, ultimate, reference and empirical constant values.

The PROMISS "flexible" model parameters is implemented as shown in Table 7-6.

Table 7-3. Micromechanics Equations Derived by Hopkins and Chamis for the Analysis of MMC Ply Mechanical and Thermal Properties.

Micromechanics Equations for the Ply Mechanical Properties

$$E_{l\,11} = k_m E_{m11} + k_f \left\{ \left[1 - \left(\frac{D}{D_o}\right)^2 \right] E_{d11} + \left(\frac{D}{D_o}\right)^2 E_{f11} \right\}$$

$$E_{l\,22} = E_{m22} \left\{ (1 - \sqrt{k_f}) + \frac{\sqrt{k_f}\left(1 - \frac{D}{D_o}\right)}{1 - \sqrt{k_f}\left(1 - \frac{E_{m22}}{E_{d22}}\right)} + \frac{\sqrt{k_f}\left(\frac{D}{D_o}\right)}{1 - \sqrt{k_f}\left[1 - \left(1 - \frac{D}{D_o}\right)\right]\frac{E_{m22}}{E_{d22}} - \left(\frac{D}{D_o}\right)\frac{E_{m22}}{E_{f22}}} \right\} = E_{l\,33}$$

$$G_{l\,12} = G_{m12} \left\{ (1 - \sqrt{k_f}) + \frac{\sqrt{k_f}\left(1 - \frac{D}{D_o}\right)}{1 - \sqrt{k_f}\left(1 - \frac{G_{m12}}{G_{d12}}\right)} + \frac{\sqrt{k_f}\left(\frac{D}{D_o}\right)}{1 - \sqrt{k_f}\left[1 - \left(1 - \frac{D}{D_o}\right)\right]\frac{G_{m12}}{G_{d12}} - \left(\frac{D}{D_o}\right)\frac{G_{m12}}{G_{f12}}} \right\} = G_{l\,13}$$

$$G_{l\,23} = G_{m23} \left\{ (1 - \sqrt{k_f}) + \frac{\sqrt{k_f}\left(1 - \frac{D}{D_o}\right)}{1 - \sqrt{k_f}\left(1 - \frac{G_{m23}}{G_{d23}}\right)} + \frac{\sqrt{k_f}\left(\frac{D}{D_o}\right)}{1 - \sqrt{k_f}\left[1 - \left(1 - \frac{D}{D_o}\right)\right]\frac{G_{m23}}{G_{d23}} - \left(\frac{D}{D_o}\right)\frac{G_{m23}}{G_{f23}}} \right\}$$

$$\nu_{l\,12} = k_m \nu_{m12} + k_f \left\{ \left[1 - \left(\frac{D}{D_o}\right)^2 \right] \nu_{d12} + \left(\frac{D}{D_o}\right)^2 \nu_{f12} \right\} = \nu_{l\,13}$$

$$\nu_{l\,23} = \frac{E_{l\,22}}{2G_{l\,23}} - 1$$

Micromechanics Equations for the Ply Thermal Properties

$$C_l = k_m \left(\frac{\rho_m}{\rho_l}\right) C_m + k_f \left\{ \left[1 - \left(\frac{D}{D_o}\right)^2 \right] \left(\frac{\rho_d}{\rho_l}\right) C_d + \left(\frac{D}{D_o}\right)^2 \left(\frac{\rho_f}{\rho_l}\right) C_f \right\}$$

$$K_{l\,11} = k_m K_{m11} + k_f \left\{ \left[1 - \left(\frac{D}{D_o}\right)^2 \right] K_{d11} + \left(\frac{D}{D_o}\right)^2 K_{f11} \right\}$$

$$K_{l\,22} = K_{m22} \left\{ (1 - \sqrt{k_f}) + \frac{\sqrt{k_f}\left(1 - \frac{D}{D_o}\right)}{1 - \sqrt{k_f}\left(1 - \frac{K_{m22}}{K_{d22}}\right)} + \frac{\sqrt{k_f}\left(\frac{D}{D_o}\right)}{1 - \sqrt{k_f}\left[1 - \left(\frac{D}{D_o}\right)\right]\frac{K_{m22}}{K_{d22}} - \left(\frac{D}{D_o}\right)\frac{K_{m22}}{K_{f22}}} \right\} = K_{l\,33}$$

$$\alpha_{l\,11} = k_m \left(\frac{E_{m11}}{E_{l\,11}}\right) \alpha_{m11} + k_f \left\{ \left[1 - \left(\frac{D}{D_o}\right)^2 \right] \left(\frac{E_{d11}}{E_{l\,11}}\right) \alpha_{d11} + \left(\frac{D}{D_o}\right)^2 \left(\frac{E_{f11}}{E_{l\,11}}\right) \alpha_{f11} \right\}$$

$$\alpha_{l\,22} = \frac{E_{m22}}{E_{l\,22}} \left\{ (1 - \sqrt{k_f})\,\alpha_{m22} + \frac{\left(1 - \frac{D}{D_o}\right)\left[(1 - \sqrt{k_f})\,\alpha_{m22} + \sqrt{k_f}\,\alpha_{d22}\right]}{1 - \sqrt{k_f}\left(1 - \frac{E_{m22}}{E_{d22}}\right)} + \frac{\sqrt{k_f}\,\alpha_{m22} - k_f\left[\alpha_{m22} - \left(1 - \frac{D}{D_o}\right)\alpha_{d22} - \left(\frac{D}{D_o}\right)\alpha_{f22}\right]}{1 - \sqrt{k_f}\left[1 - \left(1 - \frac{D}{D_o}\right)\frac{E_{m22}}{E_{d22}} - \left(\frac{D}{D_o}\right)\frac{E_{m22}}{E_{f22}}\right]} \right\} = \alpha_{l\,33}$$

Table 7-4. Micromechanics Equations Derived By Hopkins And Chamis For The Analysis Of MMC Ply Strengths.

Micromechanics Equations for the Ply Uniaxial Longitudinal Strengths

$$S_{l\,11T} = S_{f11T}\left\{ k_m\left(\frac{E_{m11}}{E_{f11}}\right) + k_f\left[\left(\frac{D}{D_o}\right)^2 + \left\{1 - \left(\frac{D}{D_o}\right)^2\right\}\frac{E_{d11}}{E_{f11}}\right]\right\}$$

$$S_{l\,11C} = \text{MIN.}\begin{cases} S_{f11C}\left\{ k_m\left(\frac{E_{m11}}{E_{f11}}\right) + k_f\left[\left(\frac{D}{D_o}\right)^2 + \left\{1 - \left(\frac{D}{D_o}\right)^2\right\}\frac{E_{d11}}{E_{f11}}\right]\right\} \\ S_{m11C}\left\{ k_m + k_f\left[\left(\frac{D}{D_o}\right)^2\frac{E_{f11}}{E_{m11}} + \left\{1 - \left(\frac{D}{D_o}\right)^2\right\}\frac{E_{d11}}{E_{m11}}\right]\right\} \\ G_{m12}\left\{\dfrac{1}{k_m + k_f\left[\left(\frac{D}{D_o}\right)^2\frac{G_{m12}}{G_{f12}} + \left\{1 - \left(\frac{D}{D_o}\right)^2\right\}\frac{G_{m12}}{G_{f12}}\right]}\right\} \\ S_{m11C} + S_{l\,12} \end{cases}$$

Ply Uniaxial Transverse and Shear Strengths

$$S_{l\,11T,C} = \frac{S_{m22T,C}}{\left\{1 - \sqrt{k_f}\left[1 - \left(1 - \frac{D}{D_o}\right)\frac{E_{m22}}{E_{d22}} - \left(\frac{D}{D_o}\right)\frac{E_{m22}}{E_{f22}}\right]\right\}\sqrt{\left[1 + \phi(\phi - 1) + \frac{1}{3}(\phi - 1)^2\right]}}$$

$$\text{WHERE; } \phi = \frac{1}{\sqrt{\frac{\pi}{4k_f - 1}}}\left\{\sqrt{\frac{\pi}{4k_f}} - \frac{\left(\frac{E_{m22}}{E_{f22}}\right)}{1 - \sqrt{k_f}\left[1 - \left(1 - \frac{D}{D_o}\right)\frac{E_{m22}}{E_{d22}} - \left(\frac{D}{D_o}\right)\frac{E_{m22}}{E_{f22}}\right]}\right\}$$

LOWER BOUND;

$$S_{l\,22T,C} = \left(1 - \sqrt{\frac{4k_f - 1}{\pi}}\right) S_{m22T,C}$$

EQUATIONS FOR INTRALAMINAR SHEAR STRENGTH, ($S_{l\,12}$), ARE ANALOGOUS WITH E AND $S_{m22T,C}$ REPLACED BY G AND S_{m12} RESPECTIVELY.

Table 7-5. Micromechanics Equations Derived by Hopkins and Chamis for the Analysis of MMC Constituent Material Subregional Microstresses.

Micromechanics Equations for the Fiber Microstresses

$$\sigma_{f11} = \left[\frac{\sigma_{l\,11}}{E_{l\,11}} + \Delta T(\alpha_{l\,11} - \alpha_{f11})\right] E_{f11}$$

$$\sigma_{f22} = \frac{E_{f22}\left\{\frac{\sigma_{l\,22}}{E_{l\,22}} + \Delta T\left[\alpha_{l\,22} - (1-\sqrt{k_f})\,\alpha_{m22}^{(C)} - \sqrt{k_f}\left(1-\frac{D}{D_o}\right)\alpha_{d22}^{(C)} - \sqrt{k_f}\left(\frac{D}{D_o}\right)\alpha_{f22}^{(C)}\right]\right\}}{1-\sqrt{k_f}\left[1-\left(1-\frac{D}{D_o}\right)\frac{E_{m22}}{E_{d22}} - \left(\frac{D}{D_o}\right)\frac{E_{m22}}{E_{f22}}\right]}$$

$$\sigma_{f12} = \frac{\left(\frac{\sigma_{l\,12}}{G_{l\,12}}\right) G_{f12}}{1-\sqrt{k_f}\left[1-\left(1-\frac{D}{D_o}\right)\frac{G_{m12}}{G_{d12}} - \left(\frac{D}{D_o}\right)\frac{G_{m12}}{G_{f12}}\right]}$$

EQUATIONS FOR σ_{f23} ARE ANALOGOUS TO EQUATIONS FOR σ_{f12}

Micromechanics Equations for the Interphase Microstresses

$$\sigma_{d11} = \left[\frac{\sigma_{l\,11}}{E_{l\,11}} + \Delta T(\alpha_{l\,11} - \alpha_{d11})\right] E_{d11}$$

$$\sigma_{d22}^{(B)} = \frac{E_{d22}^{(B)}\left\{\frac{\sigma_{l\,22}}{E_{l\,22}} + \Delta T\left[\alpha_{l\,22} - (1-\sqrt{k_f})\,\alpha_{m22}^{(B)} - \sqrt{k_f}\,\alpha_{d22}^{(B)}\right]\right\}}{1-\sqrt{k_f}\left(1-\frac{E_{m22}}{E_{d22}}\right)}$$

$$\sigma_{d22}^{(C)} = \frac{E_{d22}^{(C)}\left\{\left(\frac{\sigma_{l\,22}}{E_{l\,22}}\right) + \Delta T\left[\alpha_{l\,22} - (1-\sqrt{k_f})\,\alpha_{m22}^{(C)} - \sqrt{k_f}\left(1-\frac{D}{D_o}\right)\alpha_{d22}^{(C)} - \sqrt{k_f}\left(\frac{D}{D_o}\right)\alpha_{f22}\right]\right\}}{1-\sqrt{k_f}\left[1-\left(1-\frac{D}{D_o}\right)\frac{E_{m22}}{E_{d22}} - \left(\frac{D}{D_o}\right)\frac{E_{m22}}{E_{f22}}\right]}$$

$$\sigma_{d12}^{(B)} = \frac{\left(\frac{\sigma_{l\,12}}{G_{l\,12}}\right) G_{d12}^{(B)}}{1-\sqrt{k_f}\left(1-\frac{G_{m12}}{G_{d12}}\right)}$$

$$\sigma_{d12}^{(C)} = \frac{\left(\frac{\sigma_{l\,12}}{G_{l\,12}}\right) G_{d12}^{(C)}}{1-\sqrt{k_f}\left[1-\left(1-\frac{D}{D_o}\right)\frac{G_{m12}}{G_{d12}} - \left(\frac{D}{D_o}\right)\frac{G_{m12}}{G_{f12}}\right]}$$

EQUATIONS FOR $\sigma_{d23}^{(B,C)}$ ARE ANALOGOUS TO EQUATIONS FOR $\sigma_{d12}^{(B,C)}$

Micromechanics Equations for the Matrix Microstresses

$$\sigma_{m11} = \left[\frac{\sigma_{l\,11}}{E_{l\,11}} + \Delta T(\alpha_{l\,11} - \alpha_{m11})\right] E_{m11}$$

$$\sigma_{m22}^{(A)} = \left[\frac{\sigma_{l\,22}}{E_{l\,22}} + \Delta T(\alpha_{l\,22} - \alpha_{m22}^{(A)})\right] E_{m22}^{(A)}$$

$$\sigma_{m22}^{(B)} = \frac{E_{m22}^{(B)}\left\{\frac{\sigma_{l\,22}}{E_{l\,22}} + \Delta T\left[\alpha_{l\,22} - (1-\sqrt{k_f})\,\alpha_{m22}^{(B)} - \sqrt{k_f}\,\alpha_{d22}^{(B)}\right]\right\}}{1-\sqrt{k_f}\left(1-\frac{E_{m22}}{E_{d22}}\right)}$$

$$\sigma_{m22}^{(C)} = \frac{E_{m22}^{(C)}\left\{\frac{\sigma_{l\,22}}{E_{l\,22}} + \Delta T\left[\alpha_{l\,22} - (1-\sqrt{k_f})\,\alpha_{m22}^{(C)} - \sqrt{k_f}\left(1-\frac{D}{D_o}\right)\alpha_{d22}^{(C)} - \sqrt{k_f}\left(\frac{D}{D_o}\right)\alpha_{f22}\right]\right\}}{1-\sqrt{k_f}\left[1-\left(1-\frac{D}{D_o}\right)\frac{E_{m22}}{E_{d22}} - \left(\frac{D}{D_o}\right)\frac{E_{m22}}{E_{f22}}\right]}$$

$$\sigma_{m12}^{(A)} = \left(\frac{\sigma_{l\,12}}{G_{l\,12}}\right) G_{m12}^{(A)}$$

$$\sigma_{m12}^{(B)} = \frac{\left(\frac{\sigma_{l\,12}}{G_{l\,12}}\right) G_{m12}^{(B)}}{1-\sqrt{k_f}\left(1-\frac{G_{m12}}{G_{d12}}\right)}$$

$$\sigma_{m12}^{(C)} = \frac{\left(\frac{\sigma_{l\,12}}{G_{l\,12}}\right) G_{m12}^{(C)}}{1-\sqrt{k_f}\left[1-\left(1-\frac{D}{D_o}\right)\frac{G_{m12}}{G_{d12}} - \left(\frac{D}{D_o}\right)\frac{G_{m12}}{G_{f12}}\right]}$$

EQUATIONS FOR $\sigma_{m23}^{(A,B,C)}$ ARE ANALOGOUS TO EQUATIONS FOR $\sigma_{m12}^{(A,B,C)}$

Modifications are made to interactively generate sensitivity of primitive parameters from METCAN in order to simulate the distribution of the matrix and fiber, and interface for specific subregions of the material microstress. With the ability to save the perturbed distribution in the Fail Probplot subroutine, the randomness in the calculated ply stress/modulus, by constituent modulus, volume fraction, etc. from METCAN is simulated from the anticipated respective probabilistic distributions. For demonstration purposes, these distributions were selected to be:

1) Lognormal for ($E_{l\,11}$, $E_{l\,22}$, $\sigma_{l\,11}$, $\sigma_{l\,22}$, k_f) with 5% deviation of the present or data base mean,

2) Weibull for (E_{f11}, E_{f22}, E_{m11}, E_{m22}, E_{d11}, E_{d22}) with 5% deviation of the present or data base mean,

3) Normal for (α_l , α_f, α_m, α_d) with 3% deviation of the present or data base mean.

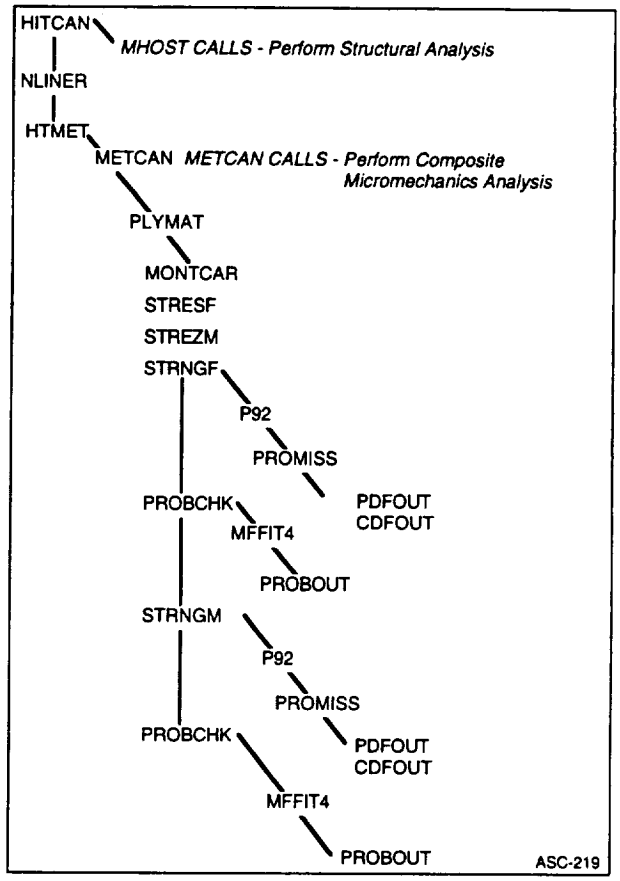


Figure 7-2. Schematic Diagram of the Subroutine Call Modifications Made to the Program HITCAN by Adding the Subroutines METPRO, PROMISS, PDFOUT, CDFOUT, PROBCHK, MFFIT4 and PROBOUT.

The calculated stress distributions from METCAN are saved in a file at time t with the ply information needed to perform risk assessment in the subroutine PROBPLOT. This subroutine plots the appropriate CDF and PDF using PROMISS with degraded constituent material strengths. A realization of the probability of failure is performed by finding the intercept of the three probability densities functions and adding the common populations so that the sum can be compared to the cumulative distribution function of degraded strength. The result is then used as an estimated probability of failure for the specific subregion of the material.

Beginning with the ply-level of analysis, we have concentrated on the computations performed by subroutine METCAN. In order to transform the deterministic analysis of this subroutine into nondeterministic form, it was necessary to examine the laminate theory and constitutive micromechanical equations utilized to calculate the composite response, ply properties and ply strengths.

After initializing variables on the first iteration loop, a call to subroutine COMSA (Composite Stress Analyzer) initiates the laminate theory analysis to compute the ply stresses and strains based on the time step nodal stresses, ply temperature and local curvature. The calculations performed in this analysis center on solving the following relationships given by:

Ply Strain } $$\{\varepsilon_l\} = [R_{l\varepsilon}]\langle\{\varepsilon_{co}\} - z_l\{x_c\}\rangle$$

Table 7-6. PROMISS "flexible" Model Parameters Passed by Subroutine MT Using Materials Data Base and Calculations for Nodal Stresses and Temperature From HITCAN

Effect or Primative Variable	Variable Symbol	Units	Distribution Type	Mean Used	Deviation (% of Mean)
Quasi-Static Stress	S_{SF}	ksi	Weibull	Database	5.0
	σ_S	ksi	Lognormal	Calculated	7.0
	σ_{So}	ksi	Lognormal	Database	5.0
	a_1	N/A	Normal	Database	3.0
Stress Rate	$\dot{S}_{SF}$	ksi/sec	Weibull	Database	5.0
	$\dot{\sigma}_S$	ksi/sec	Lognormal	Calculated	7.0
	$\dot{\sigma}_{So}$	ksi/sec	Lognormal	Database	5.0
	a_2	N/A	Normal	Database	3.0
Temperature	T_F	°F	Weibull	Database	3.0
	T	°F	Weibull	Calculated	3.0
	T_o	°F	Weibull	Database	3.0
	a_3	N/A	Normal	Database	3.0
Temperature Rate Change	$\dot{T}_F$	°F	Weibull	Database	3.0
	$\dot{T}$	°F	Weibull	Calculated	3.0
	$\dot{T}_o$	°F	Weibull	Database	3.0
	a_3	N/A	Normal	Database	3.0

Ply Stress }
$$\{\sigma_l\} = [E_l]\left([R_{l\varepsilon}]\{\varepsilon_l\} - \Delta T\{\alpha_l\}\right)$$

Force Displacement }
$$\begin{Bmatrix}\{\varepsilon_{co}\}\\ \{x_{co}\}\end{Bmatrix} = \begin{bmatrix}[A_c] & [C_c]\\ [C_c] & [D_c]\end{bmatrix}^{-1}\begin{Bmatrix}\{N_c\}+\{N_{cT}\}\\ \{M_c\}+\{M_{cT}\}\end{Bmatrix}$$

Laminate Response }
$$[A_c, C_c, D_c] = \sum_{i=1}^{n_l}\left\langle (Z_t - Z_b), \frac{1}{2}\left(Z_t^2 - Z_b^2\right), \frac{1}{3}\left(Z_t^3 - Z_b^3\right)[R_l]^T[E_l][R_l]\right\rangle_i$$

Probabilistic treatment of these relationships will require the application of procedures in which the independent variables having uncertainties of interest will utilize an assumed probabilistic distribution. Random selection of values from these distributions can be computationally simulated with routine from PROMISS.

With respect to the selection of specific uncertainties involved in the laminate theory analysis, the calculation of the effective ply properties $[P_l]$ will play a major roll in the probabilistic transformation

of the micromechanics analysis. The uncertainties related to the ply properties are a result of the initial assumptions incorporated in the rule of mixtures type relationship used to calculate equivalent ply properties. For example, the derivation of $[E_l]$ begins with the relationship:

$$E_l = \sum_{i=1}^{n} E_i V_i = \sum_{i=1}^{n} E_i \left(\frac{A_i}{A_c}\right)$$

where E_i and V_i are the modulus and volume fraction of the i^{th} constituent, respectively. This relationship is extended to the specific geometry of the unit cell and interphase region resulting in the final form of the relationship for the ply modulus, $[E_l]$, (also given in Table 7-4) as:

$$E_l = k_m E_m + k_f \left\{ \left(\frac{D}{D_o}\right)^2 E_f + \left[1 - \left(\frac{D}{D_o}\right)^2\right] E_d \right\}$$

where k_m and k_f are the volume fractions in terms of the original fiber and matrix volume fractions before interphase growth. Therefore, a considerable portion of the composites material uncertainty can

be evaluated by properly simulating the resulting effects on the $[E_l]$ matrix with randomized selection of the independent variable distributions E_i and k_i for the fiber, matrix and interphase. In accordance with Tables 7-3 and 7-4, the simulation of constituent modulus distributions and geometric uncertainties will allow further simulations for each of the ply properties and strengths.

7.1.5 Micromechanical Risk Assessment

Up to this point in our description of the GENOA concept, we have presented an approach for transforming the deterministic micromechanics analysis of METCAN into one which is probabilistic. All of this effort has centered on extracting probability density functions and cumulative distribution functions for an array of ply and constituent material properties, strengths and stresses. After developing all of these probabilities, one may be forced to question the purpose. The purpose is simple; it is to provide enough information about the uncertainties incorporated into the analysis that a reliable assessment of the risk can be made following an evaluation of all the physical processes contributing to the probability of failure.

In order to perform a micromechanical risk assessment, a point in the multi-level iterative analysis must be chosen such that enough information is present to provide a comprehensive evaluation of probable failure. With respect to a probabilistic METCAN analysis, this point follows each realization of ply and constituent material strength degradation. Risk assessment also requires that a failure mode be used which is representative of a realistic situation which would cause probable failure. For the purposes of evaluating probable failure in metal matrix composites, a failure mode can be defined for both the ply-level and the constituent-level in the analysis. At the constituent-level, the failure mode represents a probable event which would initiate failure in the form of crack growth or local yielding. This mode is then defined as the event that a particular microstress is greater than the corresponding constituent material strength. The probability of constituent failure initiation is then calculated by:

$$P_f(i,j,k) = \int f_{\sigma_{ijk}}(x)\, F_{S_{ijk}}(x)\, dx$$

where $f_{\sigma_{ijk}}$ is the probability density function of the normalized microstress, σ/S_o, at the i^{th} structural node, the j^{th} ply and the k^{th} constituent material. $F_{S_{ijk}}$ is the cumulative distribution function for the degraded strength, S/S_o, at the i^{th} structural node, the j^{th} ply and the k^{th} constituent material. At the ply-level, the failure mode represents a greater risk of catastrophic failure. This mode is defined as the event when a ply stress is greater than the corresponding ply strength. Graphically, the assessment of risk can be shown as the overlap of the probability density functions for the stress and strength (illustrated in Figure 7-3).

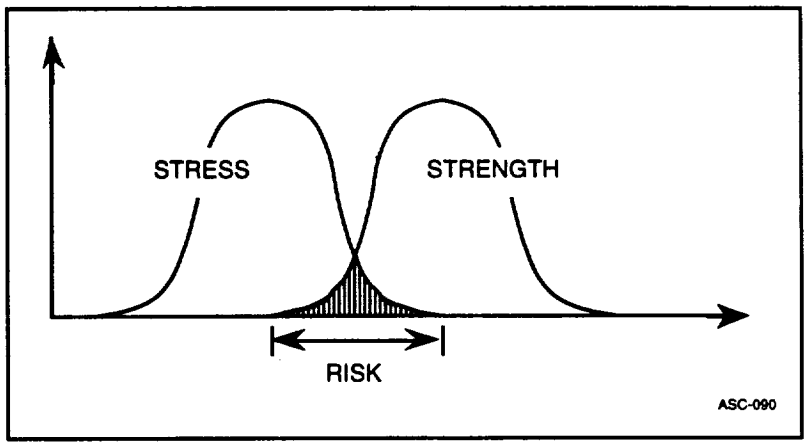


Figure 7-3. Illustration Of The Overlapping Probability Densities Functions Representative Of The Probable Event Of Failure.

It is anticipated that it will be feasible to perform risk assessment at each structural node, for each ply and ply strength, as well as each constituent and constituent material strength. With this type of analytical capability, the design of new aerospace structures and components could push the limits of new composite material systems. The use of over-bearing design allowables would become a method of the past. The ability to assess micromechanical damage would also provide a unique tool for validating and upgrading numerous mechanism-based methodologies. In short, an analytical tool providing this type of detailed structural and material response, would represent an innovative step forward in the design process of aerospace structures.

7.1.6 Integration Of Environmental And Test Data With A Stochastic Simulation

Test data can be merged into the a unified simulation analysis by using PROMISS analysis . Statistical scatter of experimentally observed properties can be explained by reasonable statistical distribution of input parameters in composite micromechanics and laminate theory predictive models (refer to Figure 7-1).

(It is important to note that other uncertainties can also be included at this level in the micromechanics analysis. An example is the probabilistic treatment of the ply alignment or misalignment q_l which is used to calculate the ply transformation matrix $[R_l]$. Small variations in the values of this matrix could have wide spread effect on the predicted behavior of the composite and could account for many of the geometric uncertainties associated with laminated composites.)

Once the various densities and distributions have been simulated, they must be stored for subsequent analyses. One feasible method is to store the probability densities and cumulative distributions of each variable as a probability array which are indexed in such a way that a pseudo-random number can rapidly extract a value from the realization. This method is utilized by PROMISS and has only one drawback; it consumes a large amount of memory when dealing with large arrays of variables.

7.2 REFERENCES

7.1. L. Boyce, "Development of Advanced Methodologies for Probabilistic Constitutive Relationships of Material Strength Models, Phase 2," NASA NAG 3-867, 1989.

7.2. L. Boyce and T. B. Lovelace, "PROMISS User's Manual," Final Technical Report NAG 3-867, Appendix 1, 1990.

7.3. D. A. Hopkins and P.L.N Murthy, "METCAN - The Metal Matrix Composite Analyzer," Lewis Structures Technology 1988, Vol. 2, Structural Mechanics, NASA CP-3003-VOL-2, pp. 141-156, 1988.

7.4. S. K. Mital, "Ceramic Matrix Composite Analyzer, "CEMCAN User's Guide," NASA Lewis Research Center, Cleveland, Ohio, 1994.

7.5. P.L.N Murthy and C. C. Chamis, "Integrated Composite Analyzer (ICAN), Users and Programmers Manual," NASA TP-2515, 1986.

7.6. D. A. Hopkins and C. C. Chamis, "A Unique Set of Micromechanical Equations For High Temperature Metal Matrix Composites," NASA TM-87154, 1985.

8.0 Verification And Demonstration

8.1 NUMERICAL RESULTS OF PARALLEL ANALYSIS

Space Shuttles Main Engine Turbo Blade Parallel Analysis

Space shuttles main engine blades (SSME) model with 4,725 dof. was considered for benchmark, as shown in Figure 8-1. RIP was used to partition the model into an increasing number of sub domains (domain = 64).

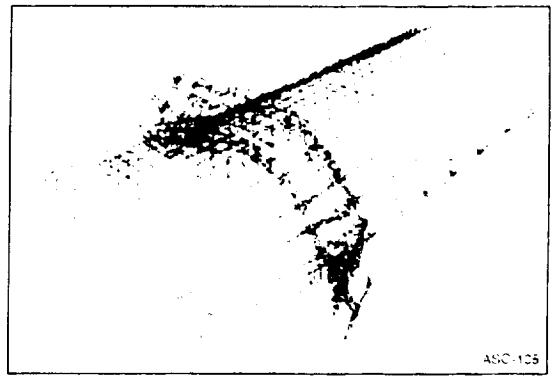


Figure 8-1. SSME Turbine Blade Model

The results with and without MAESTRO optimization on *NASA Lewis Advanced Computing Environment (LACE)* two IBM-SP2 nodes and two HP workstations are shown in Figure 8-2. It is shown that the optimization reduces the CPU time specially for large number of divisions.

We also pre calculated the optimized running time using up to 64 divisions and 64 processors and the results are shown in Figure 8-3. As shown for a fixed number of the processors, more subdivisions of the finite element model leads to smaller super elements and hence speed up the parallel computations. The replacement of the operations distribution and optimization reduced the maximum processor's activity by orders of magnitude even for the case of single processor run. The speed up may be achieved by increasing both the number of the divisions and the number of the super elements. However, an increase in the number of the processors reduces the CPU time initially but as the IPC increases, the total time would no longer after a point.

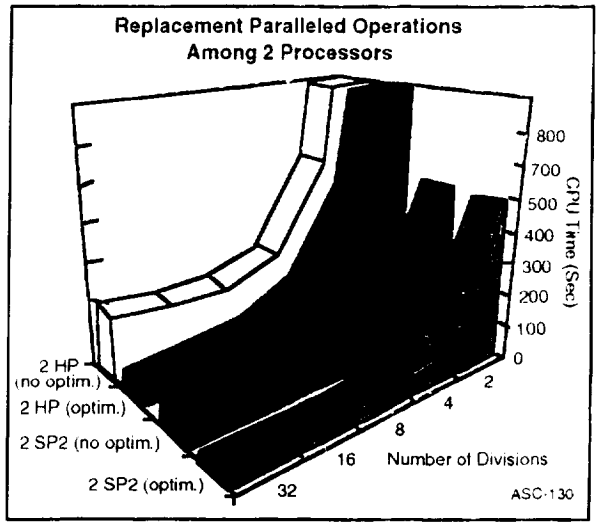


Figure 8-2. Comparison of CPU Time After PRPA Replacement of Operations for SSME Turbine Blade Model

Turbo blade solution timing by the Maestro solver on LACE RS/6000 is shown in Figure 8-4 is an enlargement of Figure 8-3. As shown in the upper chart the MAESTRO precalculates and optimizes the timing the number of divisions and the number of processors. The program estimated that the best CPU solution can be achieved with 32 divisions, and 8 processors The lower chart shows the optimized run time CPU solution can be achieved using 32 divisions on 16 processors.

Figure 8-5 shows the comparison of the CPU time for each processor. The upper chart compares the CPU distribution with and without optimization. As shown a more effective load distribution can be achieved with optimization. The lower chart compares all benchmarks for different number of divisions for the different number of processors. As shown, best CPU can be achieved using 32 divisions and 16 processors

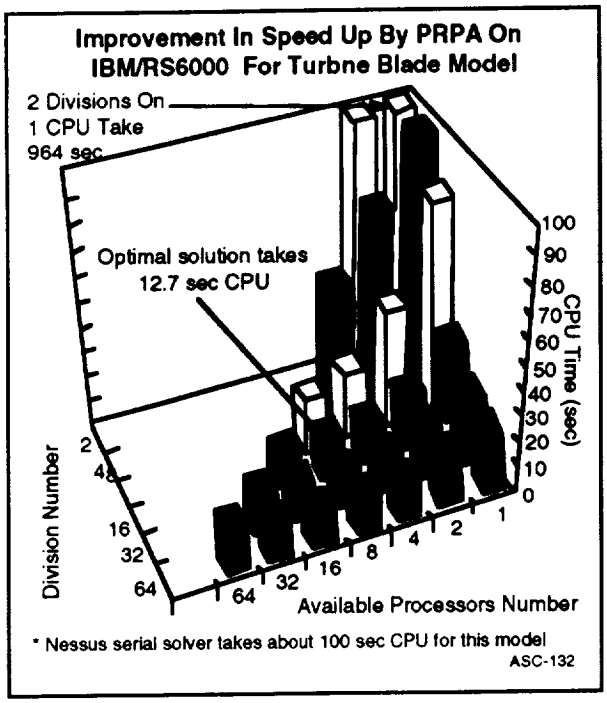


Figure 8-3. Improvement in Speed up by PRPA on IBM/RS6000 for Turbine Blade Model

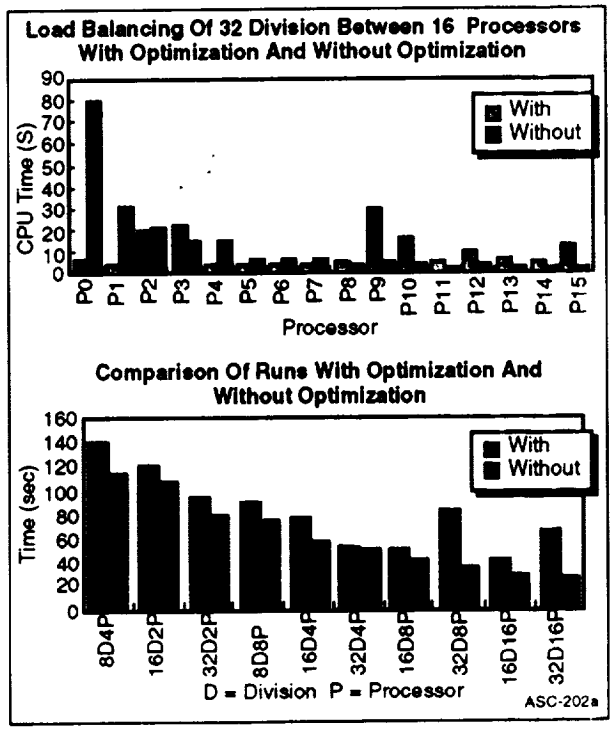


Figure 8-5. Load Balancing and Comparison of Several Runs With and Without Optimization

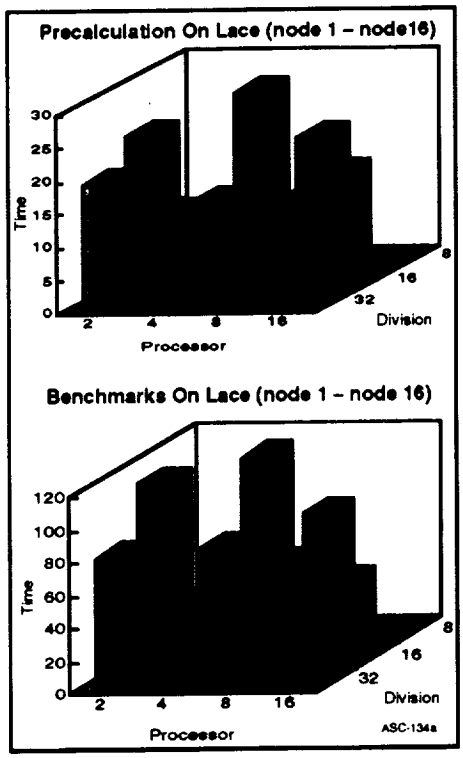


Figure 8-4. An Enlargement of Critical Region of Figure 8-3

Composite Tensile Coupon Model

We first consider a test specimen with 2664 degrees of freedom and partition its domain with various levels of cut. The number of super elements is increased with each cut resulting in smaller super elements. The individual as well as the total time for assembly of super elements is reduced as the number of cuts (and therefore the number of super elements) is increased. The total assembly time initially is reduced rapidly as shown in Figures 8-6(a) and 8-6(b). The major time in solving the systems of algebraic equations is the time used for condensation and decomposition of stiffness matrices, which is performed in subroutine COND of GENOA software. The total time for these major operations, like the total assembly time, is dramatically decreased initially. Figures 8-6(a) and 8-6(b) shows that the multifrontal algorithm with even one processor (sequential processing) can save significant amount of time in solving finite element problems. The CPU time can be further reduced by using several processors in parallel and optimizing their tasks. However, as shown in Figure 8-6(a), the IPC time is increased with an increase in the number of processors. Initially, the total processors activity is reduced as the number of processors is increased.

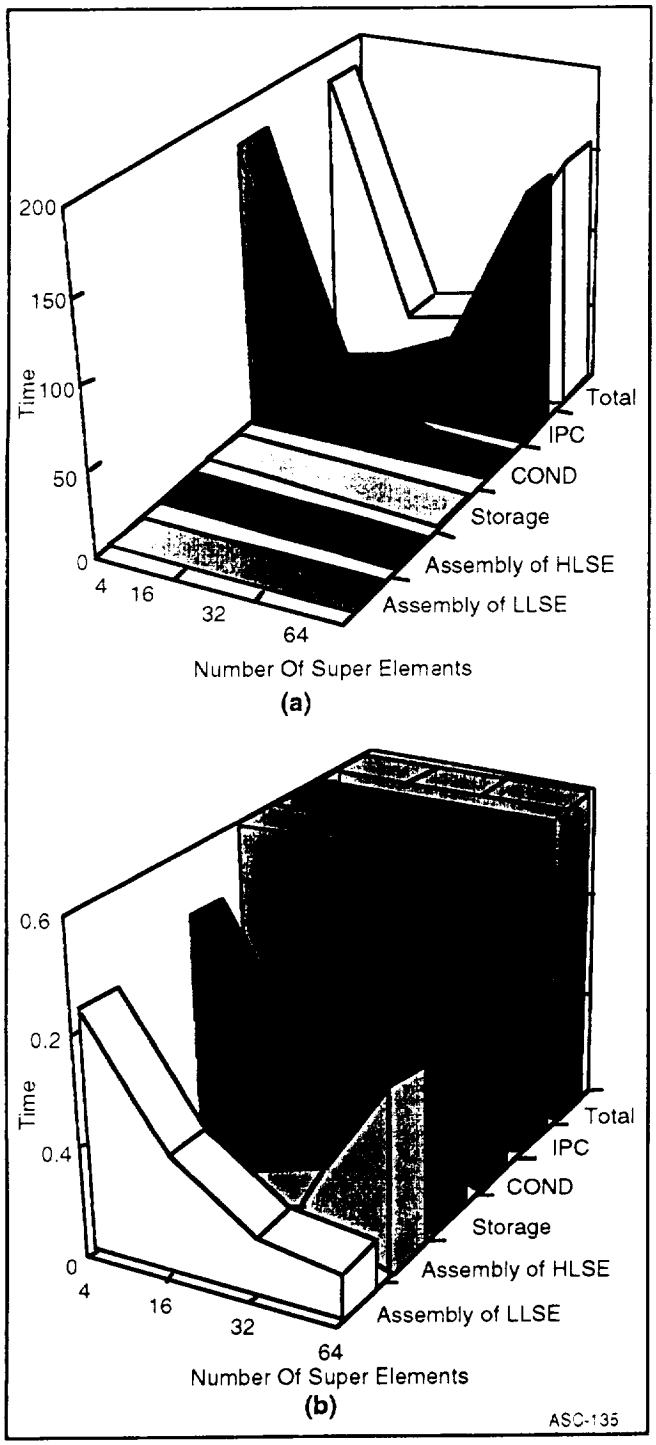


Figure 8-6(a), (b). An Increase in Number of Processors Initially Speeds Up the Solutions

reduced as the number of processors is increased. This is due to the sharp decrease in assembly, condensation, and decomposition times. Eventually, this speed up is overcame by an increase in IPC time when 16 or more processors are used.

High Speed Civil Transport Model(HSCT)

High performance supersonic civil transport (HSCT) is used to demonstrate large scale computing simulation as shown in Figure 8-7. Figure 8-7 demonstrates natural partitioning by PEEL program. As shown PEEL subdivides the original model to upper and lower fuselage (each with 1666 nodes, 1552 shell elements) and the wing (with 3276 nodes, 3162 shell elements). RIP is used to further partition these subdomains. We calculated the running time using up to 64 divisions (6 levels) and 64 processor. All the operations as percentage of the total time are presented in Figures 8-8(a) and 8-8(b). The time for assembly of super elements and storage are so small that can not be distinguished from the abscissa in Figure 8-8(a). The magnified time of these operations are given in Figure 8-8(b). Again, the time for assembly of super elements of the lowest levels rapidly decreases with an increase in the number of divisions. The IPC and subroutine COND are the major time consuming parts of the code. Eventually, as the number of divisions and processors is increased, the contribution of COND subroutine to total CPU time is reduces and nearly the entire time is spent for IPC.

The CPU time of optimized solution for five different levels of partitioned up to 32 processors is shown in Figure 8-9. For a fixed number of processors, more subdivisions of finite elements model leads to smaller super elements and hence speed up the parallel computations. The replacement of operations distribution and optimization reduced the maximum processor's activity by orders of magnitude even for the case of single processor run. The speed up may be achieved by increasing both the number of divisions and the number of super elements. However, an increase in the number of processors reduces the CPU time initially but as the IPC increases, the total time would no longer decrease after a point.

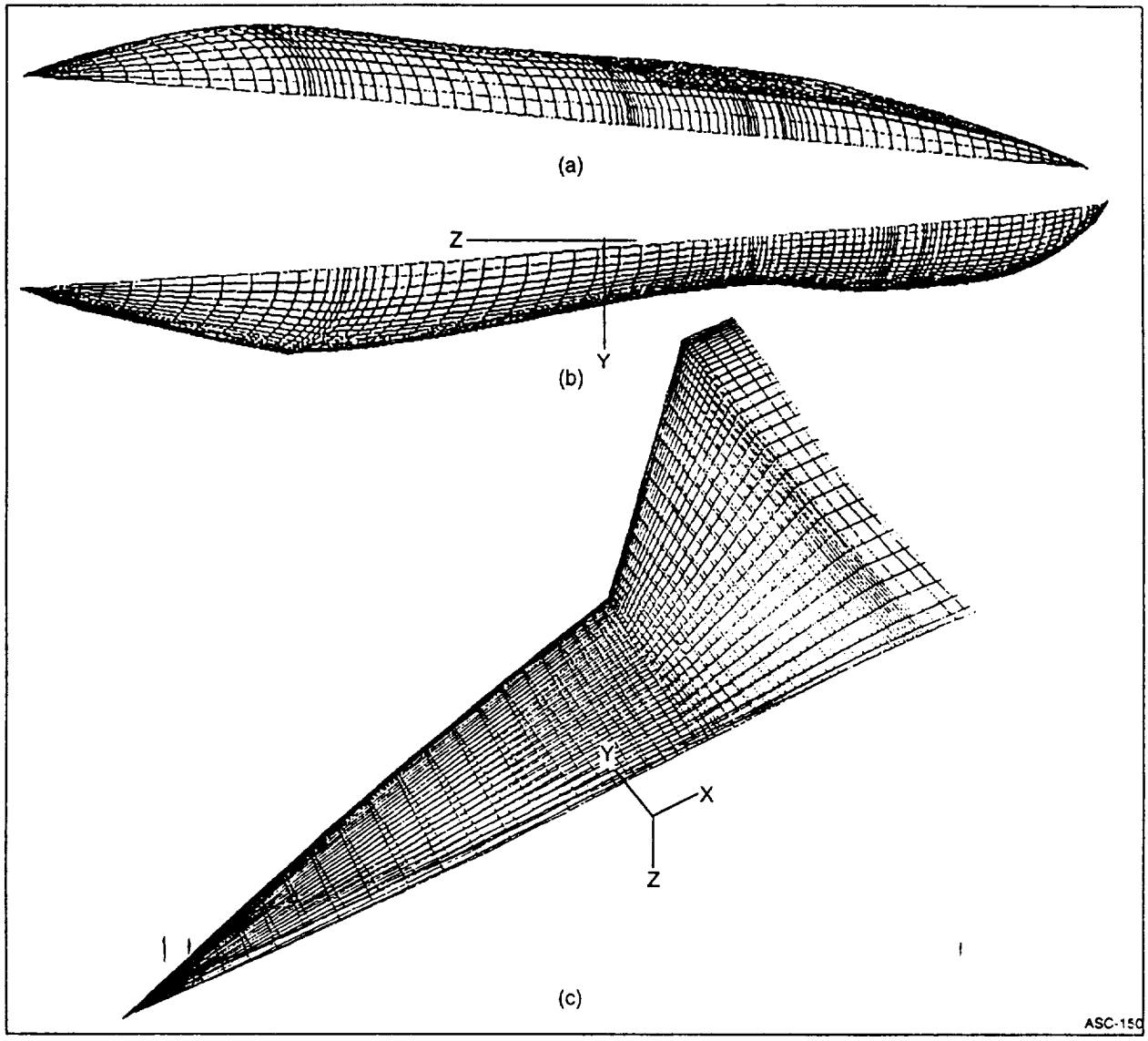


Figure 8-7. Natural Partitioning of HSCT Model

In all of the above calculations, the processor memory is used to store various parts of the stiffness matrices. A part of these stored data are used in descending the binary tree to recover the eliminated nodes. The storage time for this case is small, but if we use disk space for storage, the time would be much larger than it is now.

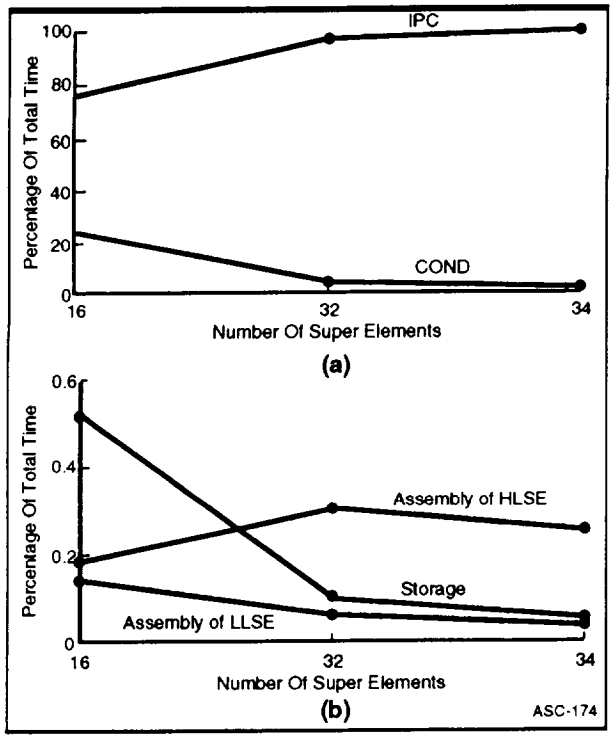


Figure 8-8. (a) The CPU Time for Various Tasks as Percentage of the Total (b) Magnified Details of (a)

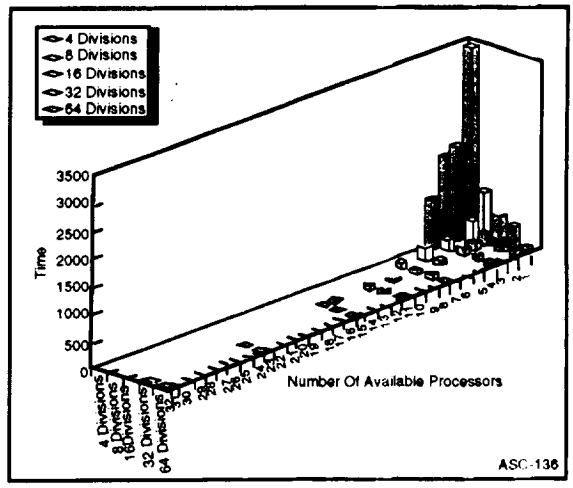


Figure 8-9. Comparison of CPU Time for Various Distribution of Super Elements Among Processors

8.2 METCAN AND PROMISS RESULTS

The probabilistic material strength simulation approach has been incorporated into METCAN. The updated version of METCAN provides the capability of evaluating the instantaneous probability distribution function of constituent property degradation due to a number of diverse random effects and their mutual interactions. Two example problems and the experimental data were used to verify the accuracy of METCAN prediction. In each example, good agreement between prediction and test data was observed. For instantaneous probabilistic analysis of constituent property, reason conclusions were derived, while the accuracy of the probabilistic evaluation will depend on the failure criteria and confidence level provided by a specific problem.

The Probability simulation of materials property degradation on the constituent level is performed by integrating finite element method, composite laminate theory, composite micromechanics, and probability theory. A multifactor interaction model (MFIM) subjected to various random effects and their mutual interactions is implemented, and the degradation of material properties of fiber, matrix, and interface is evaluated. In order to account for the statistical nature in MFIM, appropriate types of probability distribution functions (such as normal, log normal, and Weibull) are considered for primary variables. Two numerical examples are presented for probability distribution of constituent property degradation and fatigue behavior simulation of composites. It is shown that the results are in excellent agreement with the experimental results.

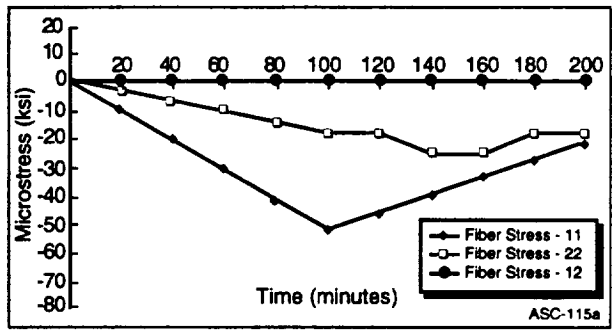


Figure 8-10. SiC/Ti-6-4 Fiber Microstress Versus Time (Ply No. 1)

RESULTS

With the incorporation of probabilistic materials strength simulation into METCAN, it was used to evaluate the instantaneous probability distribution of constituent property degradation for SiC/Ti-6-4 and SiC/Ti-24Al-11Nb under different thermal and mechanical loading conditions.

The first example simulates the stress-strain behavior of unidirectional SiC/Ti-6-4 composite under static loading. The fiber volume ratio is 0.34. Prior to the mechanical loading, the composite cools down from processing temperature (1500°F) to room temperature (70°F) as represented by the first segment in thermal loading pr°File. The microstresses of fiber and matrix in subregion A as a function of time together with the instantaneous probability distribution of property degradation are shown in Figures 8-10 and 8-11, respectively. Figure 8-10 demonstrates the fiber microstress (Sf11, Sf22, and Sf12) as function of time for ply 1. Figure 8-12 demonstrates the fiber probability distribution function (CDF) of fiber strength degradation at times (t=40, 80, 120, and 160 minutes). Figure 8-11 demonstrates the matrix microstress in subregion A (SA11, SA22, and SA12) as function of time for ply 1. Figure 8-13 demonstrates the matrix probability distribution function (CDF) of

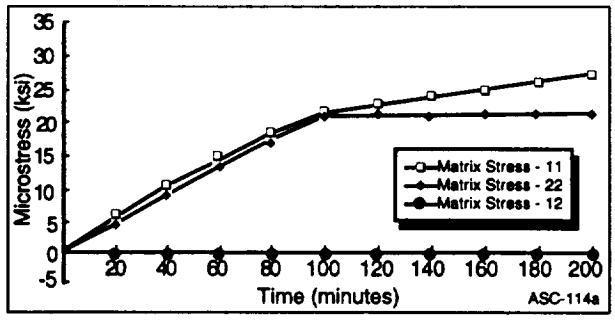


Figure 8-11. SiC/Ti-6-4 Matrix Microstress (Subregion A) Versus Time (Ply No. 1)

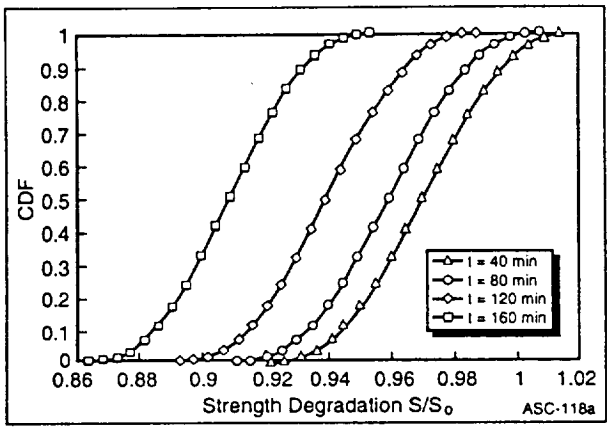


Figure 8-12. Probability Distribution Function (CDF) of Fiber Strength Degradation at Different Times

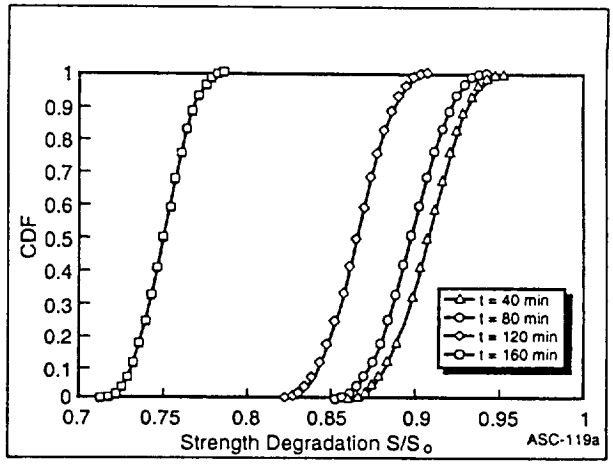


Figure 8-13. Probability Distribution Function (CDF) of Matrix (Region A) Strength Degradation at Different Times

matrix strength degradation at times (t=40, 80, 120, and 160 minutes). Comparison between METCAN simulation results of fiber volume ratio (FVR=.34) and test data of transverse strength at different temperatures (73°F, 600°F, and 800°F) are shown in Figure 8-14. Figure 8-15 compares the transverse stress-strain curve for SIC/T-6.4 with FVR=0.34 of METCAN prediction against the test data the at room temperature. An excellent agreement is observed.

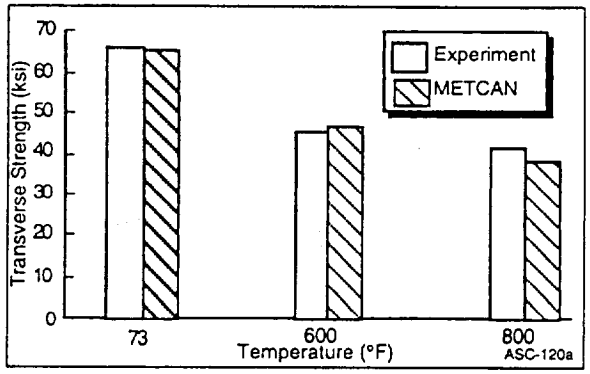


Figure 8-14. METCAN Prediction of Transverse Strength of SiC/Ti-6-4 at Different Temperatures (FVR= 0.34)

In the second example, METCAN was used to simulate the fatigue behavior of unidirectional [0] SiC/Ti-24Al-11Nb composite under in-phase cyclic thermal and mechanical loading. As shown in Figure 8-16, from t = 1000 to 3000 minutes, the laminate is subject to 100,000 mechanical load cycles with 50 ksi in longitude direction, and 200 thermal load cycles with temperature range from 70°F to 1000°F. METCAN simulation predicts that the lamina failed at N ~ 105, which is in good agreement with the reported data (Gambone, 1990). The probabilistic materials strength simulation is initiated at t = 1200 and 2000 minutes, corresponding to N = 104 and 5x104 respectively. The resulted CDF curves are shown in Figures 8-17 and 8-18 for fiber and matrix, respectively. It is clearly seen that the fiber and matrix strengths are significantly degradated after the lamina experienced 50,000 mechanical load cycles and 100 thermal load cycles.

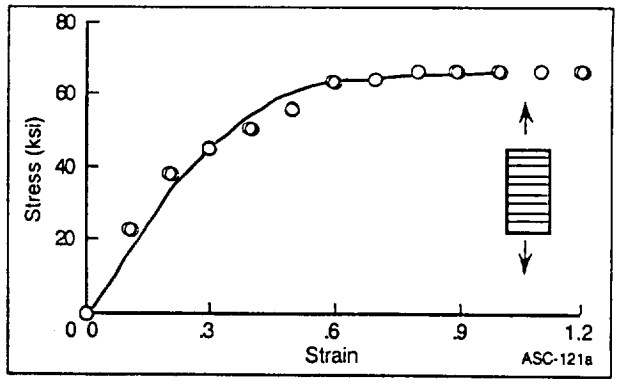


Figure 8-15. METCAN Accurate Simulation of Transverse Stress-Strain Curve of SiC/Ti-6-4 (T-73°F, FVR = 0.34)

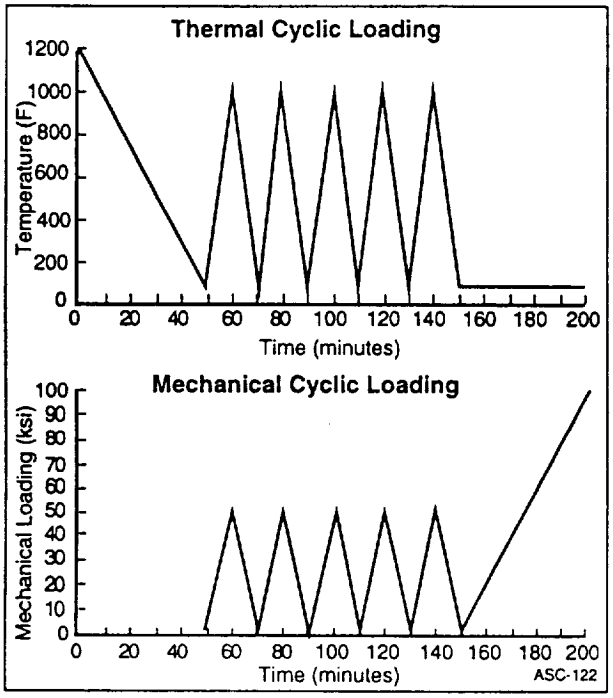


Figure 8-16. Thermal and Mechanical Cyclic Loadings of Unidirectional SiC/Ti-24Al-11Nb

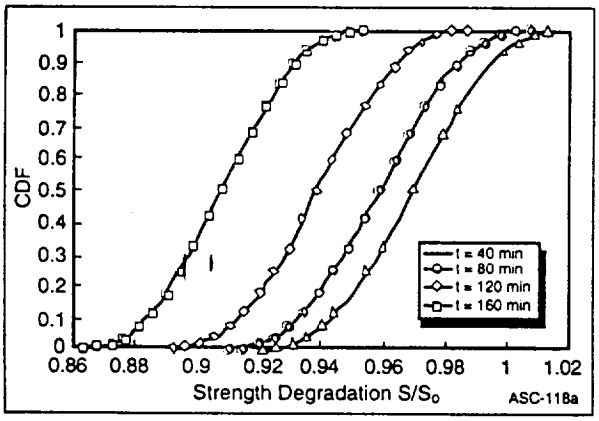


Figure 8-17. Probability Distribution Function (CDF) of Fiber Strength Degradation Due to TMF Loading

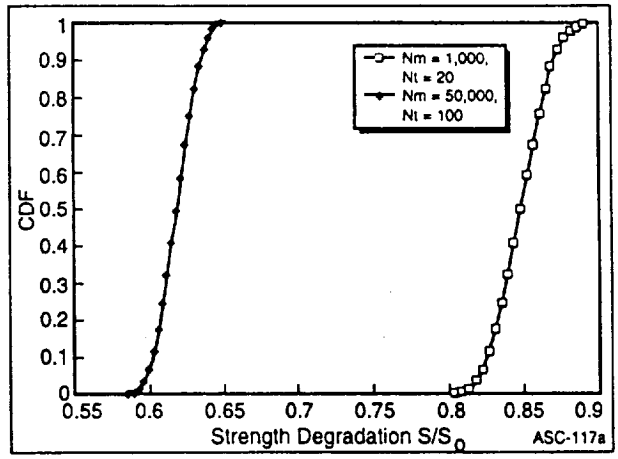


Figure 8-18. Probability Distribution Function (CDF) of Matrix Strength (Subregion A) Degradation Due to TMF Loading

8.2.1 Composite Cantilever Beam Analysis

The first demonstration of the modified HITCAN analysis having probabilistic materials strength simulation and risk assessment capability is performed for a composite cantilever beam. The finite element model and composite material description are shown in Figure 8-19. The model input deck is identical to that provided with the HITCAN demo V-1. However, the load profile was change in order to perform a dynamic loading analysis aiding in the evaluation of the codes ability to track, update and maintain the correct response information as a function of time. To this end, a linear load and temperature increase with time was performed. An initial applied force of 50 lbs. at 1000°F represented the starting conditions with an R.T. reference temperature. Over a period of one hour, the applied load was linearly increased to 100 lbs. and the applied temperature was linearly increased 200°F to 1200°F.

Using the cantilever beam model with the load profile described, it was quickly realized that several modifications were needed to provide load stepping capability. Most of these modifications were programmatic requiring additional data sets and common blocks to maintain incremental load response information. Other change required the addition of post-processing subroutines to reduce the overload of data being output for restart files.

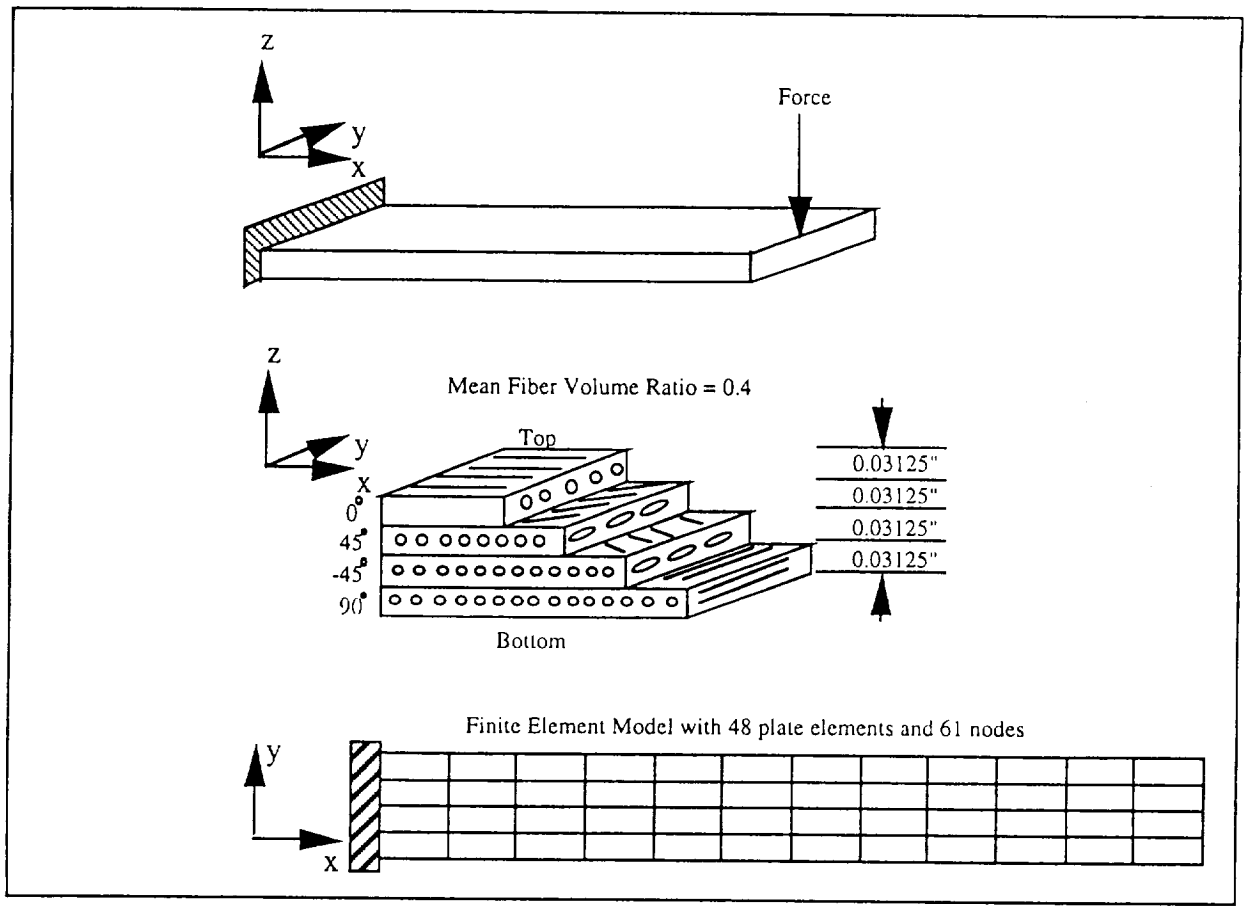


Figure 8-19. Composite Cantilever Beam Model Used To Demonstrate The Integrated Capabilities of HITCAN and PROMISS

Once all the modifications were completed, the actual demonstration produced some promising results. Figures 8-20 through 8-22 show the stress contour plots and probable failure distributions for the first load step. As can be seen in Figure 8-20, the average effective ply stresses appear to be reasonably representative of the ply layup and load condition. The top ply with 0°F orientation is in tension while the two alternating 45°F oriented exhibit varying degrees of compressive stress associated with their respective ply angles and location through the thickness of the beam. The bottom 90°F oriented ply exhibits high levels of compressive stress which is to be expected.

Post-processing files produced by the subroutine PROBCHK provide a single value of probable failure for each node with respect to the individual constituent strengths and ply numbers. As can be seen in Figures 8-21 and 8-22, the location of highest probable failure is in the bottom 90°F oriented ply where compressive matrix stress distributions are predicted to be overlapping the matrix compressive strength probability density distribution. The highest resulting probability of failure after the first load step was 27 percent located at the restrained corners of the bottom ply.

Figures 8-23 through 8-25 show the same type of results for the final load step. As can be seen in Figure 8-25, the predicted probability density functions for the matrix compressive stress and the degraded matrix compressive strength overlap considerably with the sum of the populations representative of a 62 percent probability of failure.

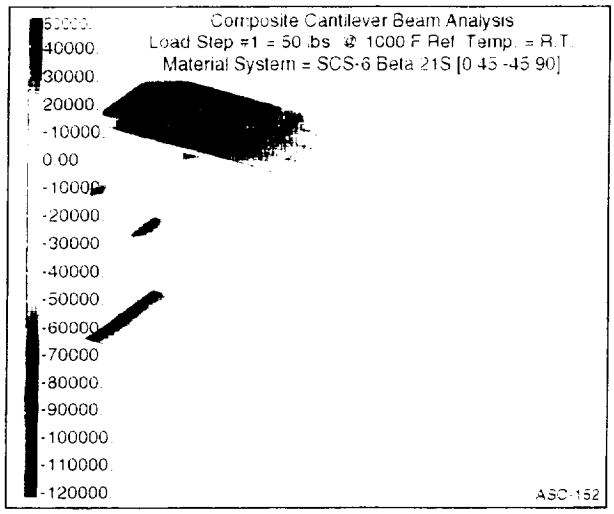


Figure 8-20. Demonstration HITCAN/PROMISS Analysis For A Composite Cantilever Beam Showing Mean Effective Ply Stress Contours After The First Load Step With 50 Lbs Force Applied @ 1000°F

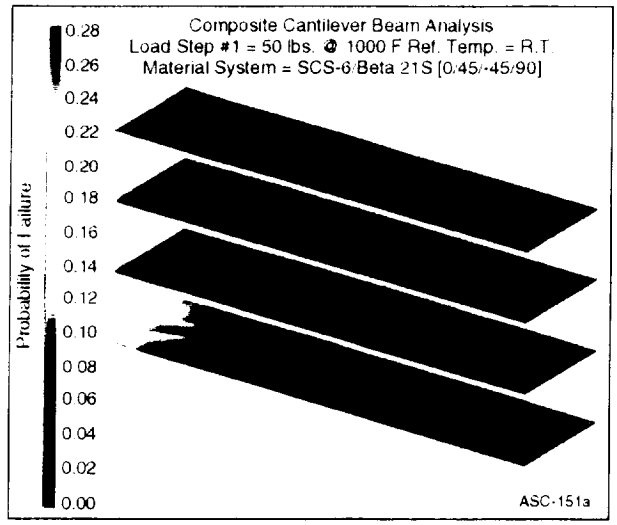


Figure 8-21. Demonstration HITCAN/PROMISS Analysis For A Composite Cantilever Beam Showing Probability Of Failure Distributions For The Matrix After The First Load Step With 50 Lbs Force Applied @ 1000°F

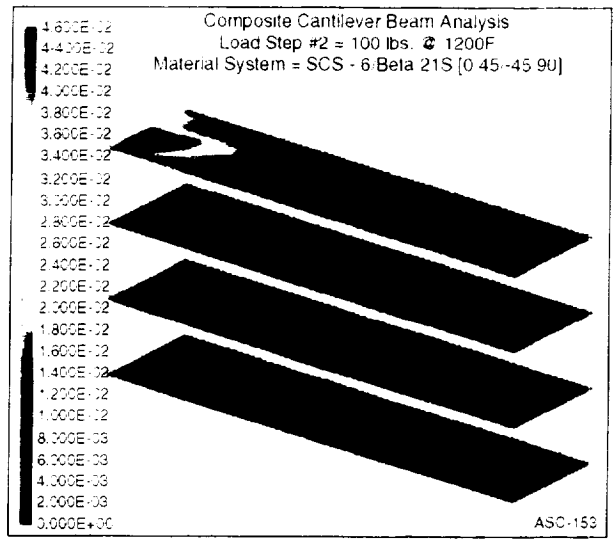


Figure 8-22. Demonstration HITCAN/PROMISS Analysis for a Composite Cantilever Beam Showing Probability of Failure Distributions for the Fiber After The First Load Step With 50 Lbs Force Applied @ 1000°F

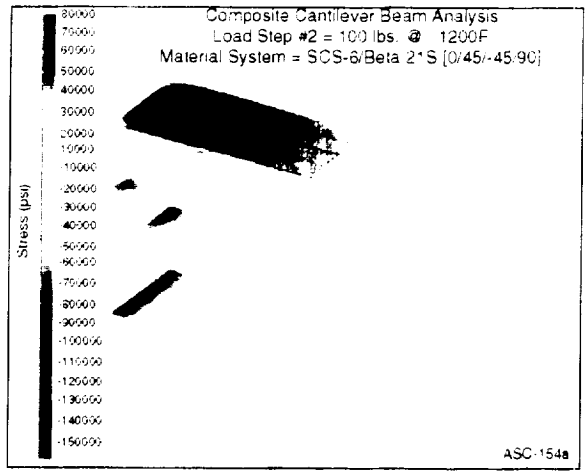


Figure 8-23. Demonstration HITCAN/PROMISS Analysis For A Composite Cantilever Beam Showing Mean Effective Ply Stress Contours After The Last Load Step With 100 Lbs Force Applied @ 1200°F

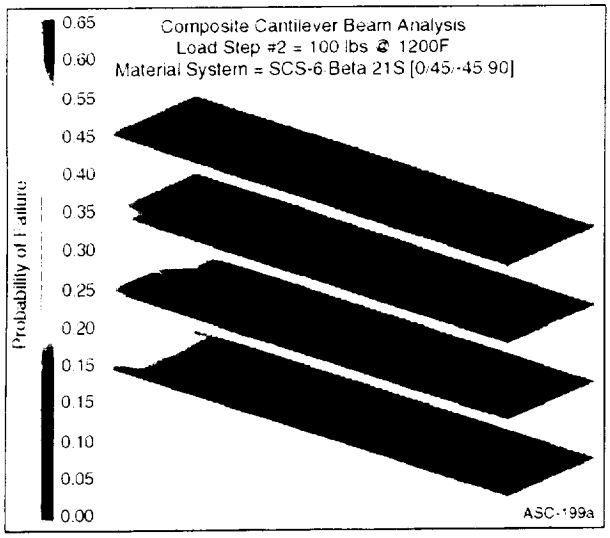


Figure 8-24. Demonstration HITCAN/PROMISS Analysis For A Composite Cantilever Beam Showing Probability Of Failure Distributions For The Matrix After The Last Load Step With 100 Lbs Force Applied @ 1200^{O}F

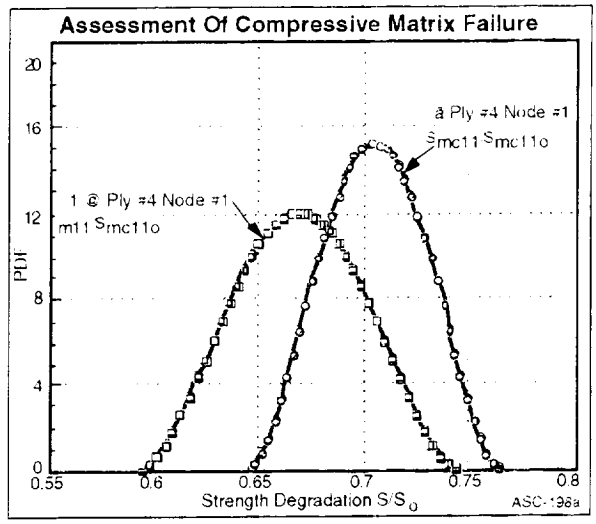


Figure 8-25. Probability Density Functions For The Matrix Microstress 11 In Region A And The Degraded Compressive Matrix Strength After The Last Load Step With 100 Lbs. Force Applied @ 1200^{O}F

Composite Tensile Coupon Analysis:

To further demonstrate the integrated HITCAN/PROMISS capability, a composite tensile coupon model was developed providing a means of directly comparing analytical results with published tensile strength data for SCS-6/Beta 21S. The tensile coupon model (shown in Figure 8-26) was developed to represent a 6" dogbone tensile coupon with a 0.5" gage section. The through thickness material model is given by the inputs for SCS-6/Beta 21S with [±45]s layup and a mean fiber volume fraction of 0.40. Thermal gradients for the gripped sections were also included to better simulate an actual elevated temperature tensile test. The same variable distributions and standard deviations were used as those in the cantilever beam analysis.

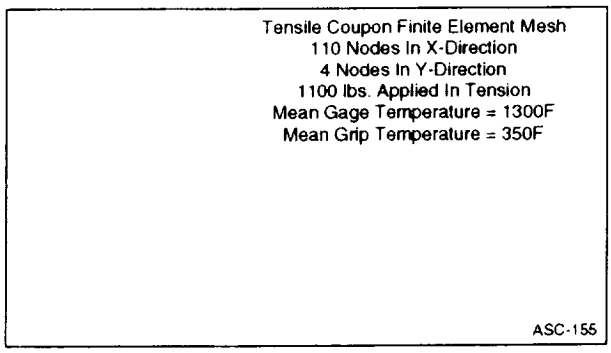


Figure 8-26. Composite tensile coupon model used to demonstrate the preliminary probabilistic analysis capability of PROMISS integrated with HITCAN

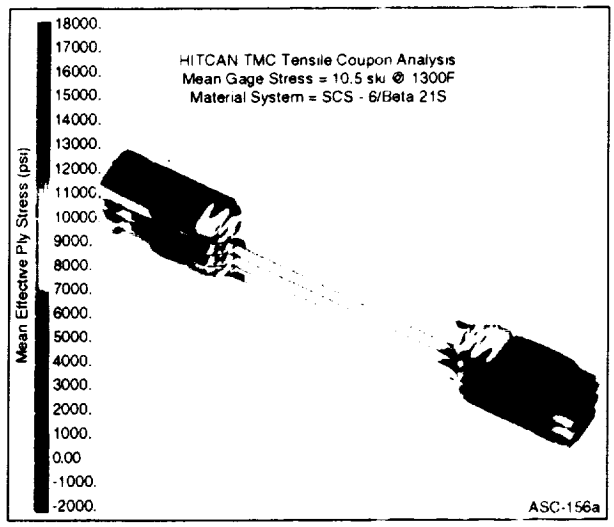


Figure 8-27 . Demonstration HITCAN/PROMISS analysis for a composite tensile test coupon showing the mean effective ply stress for SCS-6/Beta 21S $[\pm 45]_S$ with 650 lbs. applied @ 1300°F

The analysis was performed to provide gage stresses near the known 19.5 ksi elevated temperature yield strength of the SCS-6/Beta 21S [±45]s material system. The applied tensile load was 1100 lbs. @ 1300°F. Unexpectedly, the first attempt at performing the analysis was unsuccessful due to the METCAN analysis determining that a local matrix strengths had been exceeded. The second analysis used an applied load of 650 lbs. @ 1300°F. This time the analysis was successful in providing results.

As can be seen in Figure 8-27, the mean effective ply stress was predicted to exhibit highly concentrated peak stress values at the base of the dogbone fillet radius. Of particular interest is that these stress concentrations coincide with the 45°F orientation of each ply (i.e., the location of the edge stress concentration alternates sides in conjunction with the ply orientation). This type of behavior exemplifies the edge effects produced by dogbone coupons and supports the reasoning used for proscribing straight sided coupon configurations when testing MMCs.

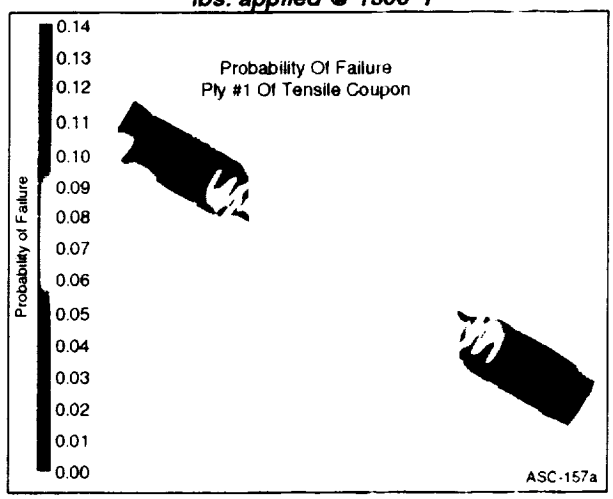


Figure 8-28. Demonstration HITCAN/PROMISS analysis For A Composite Tensile Test Coupon Showing The Probability Of Failure Distribution For Ply #1 Due To Matrix Tensile Stress 11 In Region A for SCS-6/Beta 21S $[\pm 45]_s$ with 650 lbs. Applied @ 1300°F.

The high local stress levels at the fillets helps explain why the first trial analysis did not proceed as expected. As can be seen in Figure 8-28, the probability of tensile matrix failure is predicted to be 13.8 percent at the fillets. It is expected that this value would be higher if the matrix shear microstresses were utilized for the assessment of risk. Unfortunately this effort was limited to on-axis stress. A more accurate risk assessment would be possible if deviatoric stresses were included.

8.3 VALIDATION OF AMF AND NESSUS SEQUENTIAL SOLVER

The first four sample problems in the HITCAN Demonstration Manual are presented in this section. Each problem is solved by using both the original NESSUS and the integrated AMF-NESSUS code. The output are compared graphically for each problem.

8.3.1 Demonstration Problem No. 1

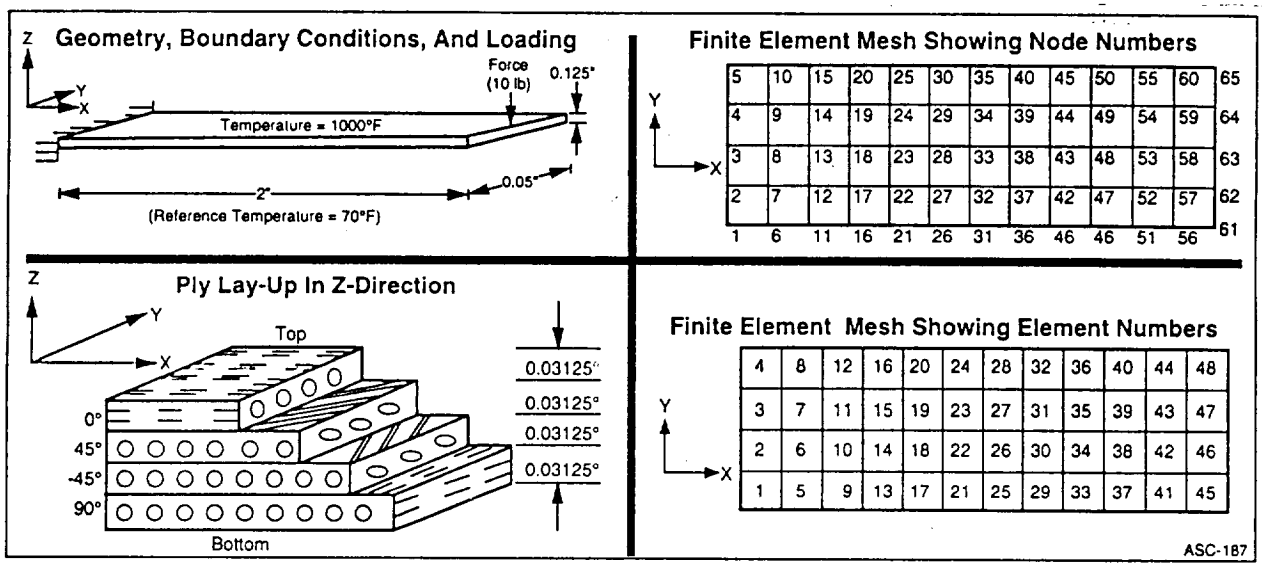


Figure 8-29. Problem No. 1 Cantilever Beam Under Bending and Uniform Temperature Loadings for (SI C/TI-15-3-3-3, 0/±45/90); 0.4 Fiber Volume Ratio.

Problem Type

Static analysis of a solid beam type structure using plate element subjected to thermo-mechanical loading.

Problem Description

A cantilever beam of 2" length and 0.5 x 0.125" cross-section is subjected to a concentrated load of 100 pounds at the center of free end and a uniform temperature increase from 70 to 1000°F. The beam is made of Sic/Ti-15-3-3-3 composite material (Silicon Carbide fiber, Titanium matrix with 15 percent Vanadium, 3 percent Aluminum, 3 percent Chromium, and 3 percent Tin, and interphase with average properties of fiber and matrix). The composite laminate consists of 4 (0/45/-45/90) plies of equal thickness with 0.4 fiber volume ratio. The ply lay-up is such that the 0 degree ply is at the top and the 90 degree at the bottom of the beam. The geometry, boundary conditions, loading, and ply lay-up are shown in the Figure 8-29. The material properties are the same as the HITCAN data bank.

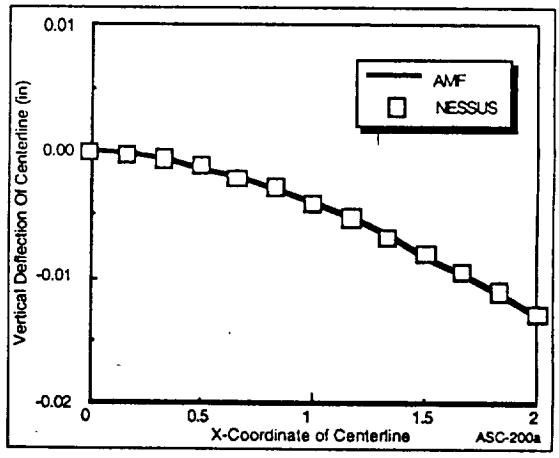


Figure 8-30. Comparison of the Displacement Results

Results

The displacements are calculated using both codes. Figure 8-30 shows the comparison of z-component (the dominating component) of centerline deflection. The results are in excellent agreement.

8.3.2 Demonstration Problem No. 2

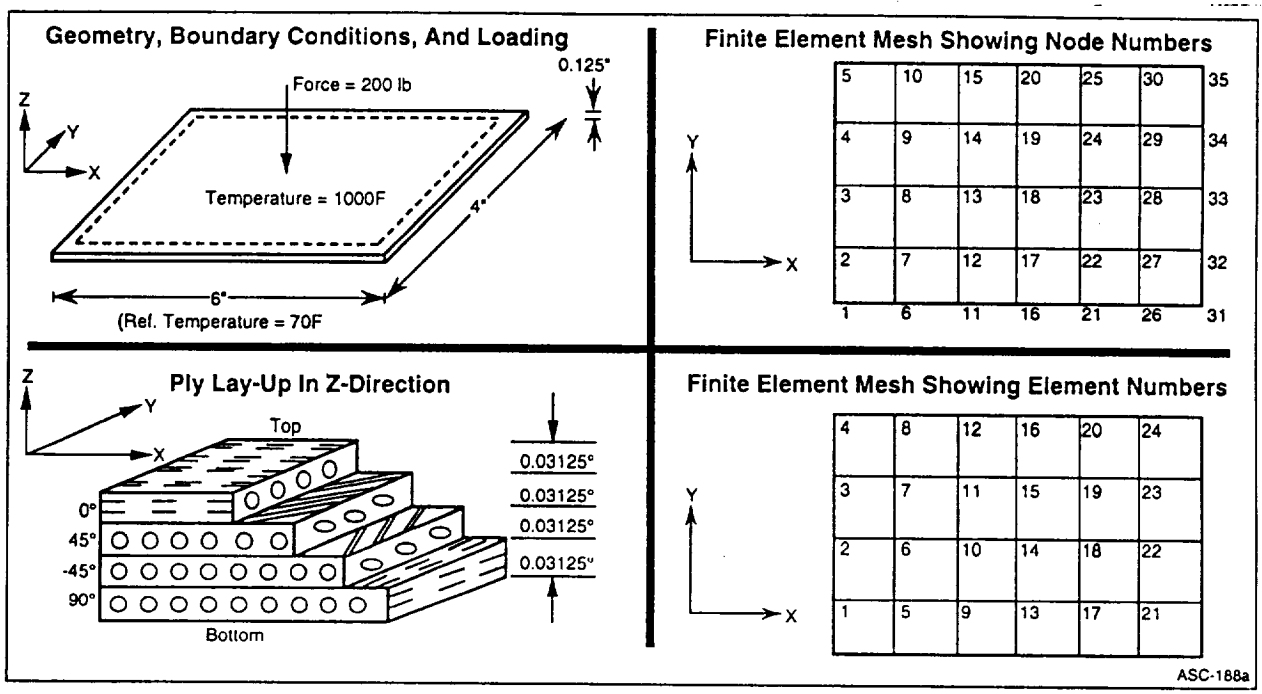


Figure 8-31. Problem No. 2 Simply Supported Plated Under Bending and Uniform Temperature Loadings for (SI C/TI-15-3-3-3, 0/±45/90); 0.4 Fiber Volume Ratio.

Problem Type

Static analysis of a solid plate type structure using plate element subjected to thermo-mechanical loading.

Problem Description

A 6" long, 4" wide, and 0.5 x 0.125" thick plate with all 4 edges simply supported is subjected to a concentrated load of 200 pounds at the top center and a uniform temperature increase from 70 to 1000°F. The plate is made of Sic/Ti-15-3-3-3 composite material (Silicon Carbide fiber, Titanium matrix with 15 percent Vanadium, 3 percent Aluminum, 3 percent Chromium, and 3 percent Tin, and interphase with average properties of fiber and matrix). The composite laminate consists of 4 (0/45/-45/90) plies of equal thickness with 0.4 fiber volume ratio. The ply lay-up is such that the 0 degree ply is at the top and the 90 degree at the bottom of the plate. The geometry, boundary conditions, loading, and ply lay-up are shown in the Figure 8-31. The material properties are the same as the HITCAN data bank.

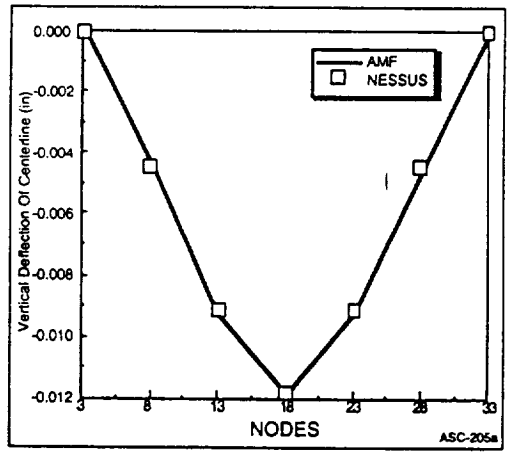


Figure 8-32. Comparison of the Displacement Results

Results

The displacements are calculate using both codes. Figure 8-32 shows the component of z-output (the dominating component) of centerline deflection. The results are in excellent agreement.

8.3.3 Demonstration Problem No. 3

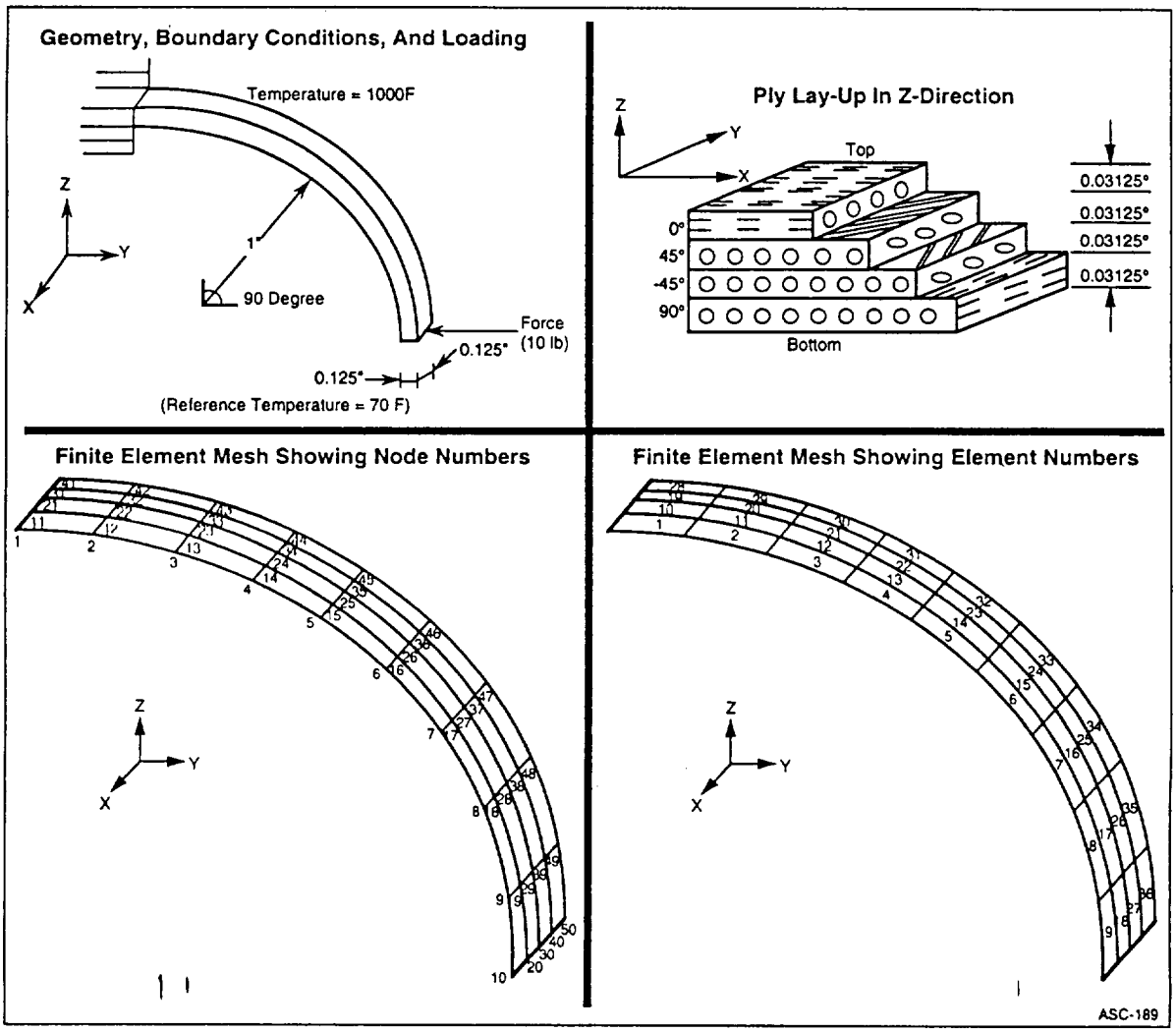


Figure 8-33. Problem No. 3 Cantilever Ring Under Bending and Uniform Temperature Loadings for (SI C/TI-15-3-3-3, 0/±45/90); 0.4 Fiber Volume Ratio.

Problem Type

Static analysis of a solid plate type structure using plate element subjected to thermo-mechanical loading.

Problem Description

A cantilever quarter (90 degree segment) ring of 1" radius and 0.125 x subjected to a concentrated load of 200 pounds at the top center and a uniform temperature increase from 70 to 1000°F. The plate is made of Sic/Ti-15-3-3-3 composite material (Silicon Carbide fiber, Titanium matrix with 15 percent Vanadium, 3 percent Aluminum, 3 percent Chromium, and 3 percent Tin, and interphase with average properties of fiber and matrix). The composite laminate consists of 4 (0/45/-45/90) plies of equal thickness with 0.4 fiber volume ratio. The ply lay-up is such that the 0 degree ply is at the top and the 90 degree at the bottom of the plate. The geometry, boundary conditions, loading, and ply lay-up are shown in the Figure 33. The material properties are the same as the HITCAN data bank.

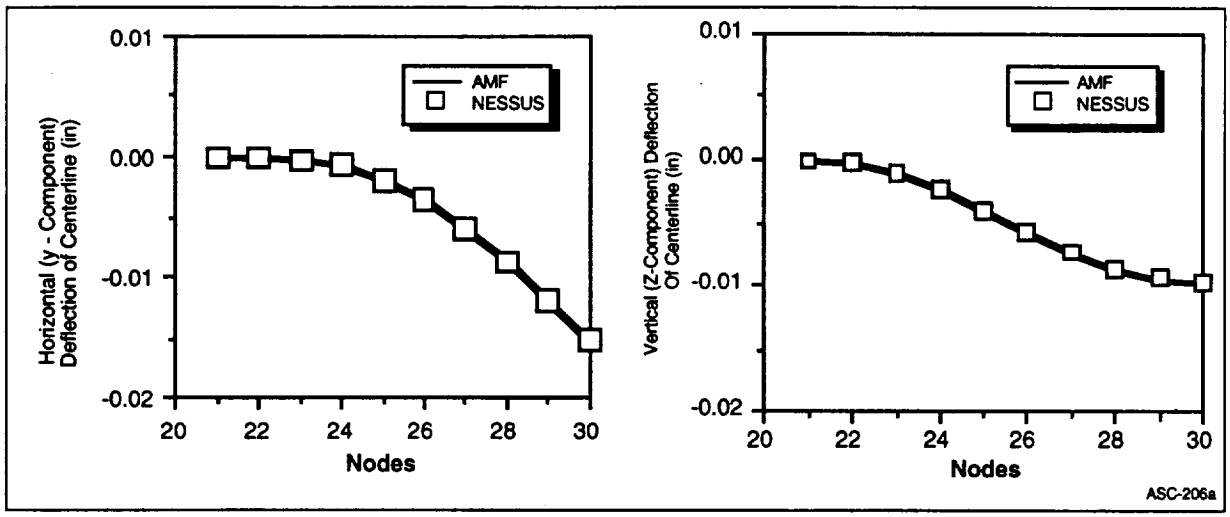


Figure 8-34. Comparison of the Displacement Results

Results

The deplacements are calculated using both codes. Figure 8-34 shows the comparison of y- and z-components of centerline deflection. The results are in excellent agreement

8.3.4 Demonstration Problem No. 4

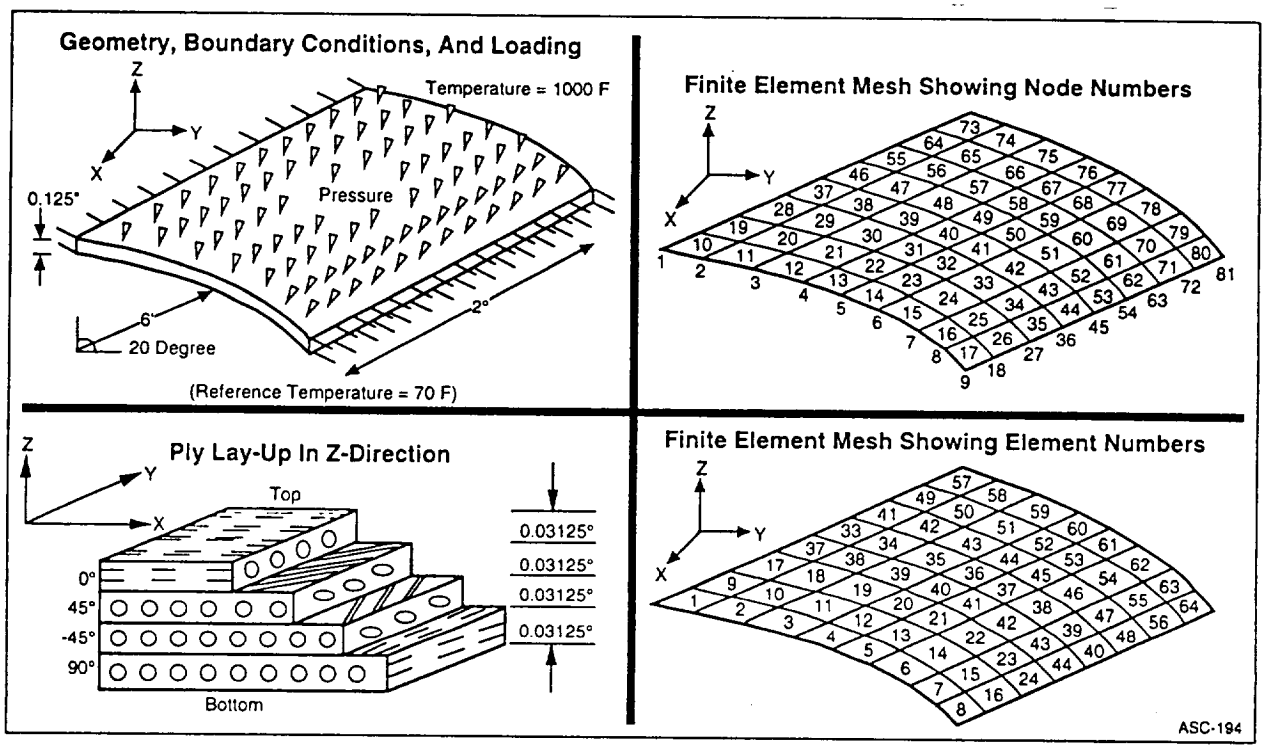


Figure 8-35. Problem No. 4 Fixed-Free Curved Panel Under Bending and Uniform Temperature Loadings for (SI C/TI-15-3-3-3, 0/±45/90); 0.4 Fiber Volume Ratio.

Problem Type

Static analysis of a solid curved panel type structure using plate element subjected to thermo-mechanical loading.

Problem Description

A curved panel (20 degree segment) of 6" radius, 2" width, and 0.125" thickness with both straight edges clamped and both curved edges free, is subjected to a concentrated load of 200 pounds at the top center and a uniform temperature increase from 70 to 1000°F. The plate is made of Sic/Ti-15-3-3-3 composite material (Silicon Carbide fiber, Titanium matrix with 15 percent Vanadium, 3 percent Aluminum, 3 percent Chromium, and 3 percent Tin, and interphase with average properties of fiber and matrix). The composite laminate consists of 4 (0/45/-45/90) plies of equal thickness with 0.4 fiber volume ratio. The ply lay-up is such that the 0 degree ply is at the top and the 90 degree at the bottom of the curved panel. The geometry, boundary conditions, loading, and ply lay-up are shown in Figure 8-35. The material properties are the same as the HITCAN data bank.

Result

The deplacements are calculated using both codes. Figure 8-36 shows the comparison of y- and z-components of centerline deflection. The results are in excellent agreement

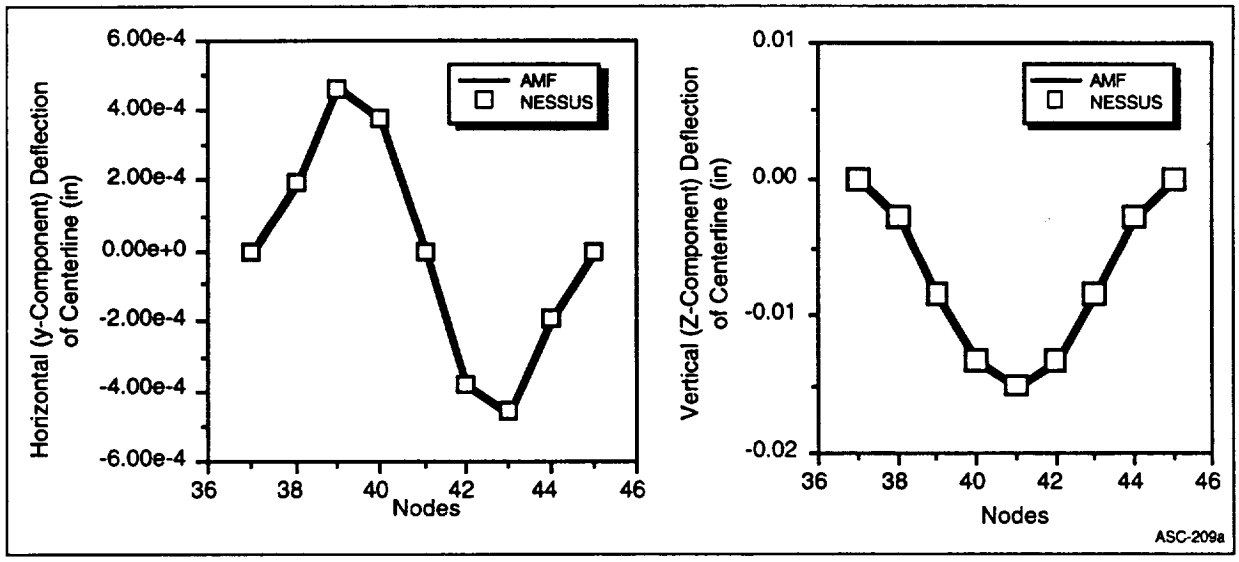


Figure 8-36. Comparison of the Displacement Results

9.0 Summary And Conclusion

The primary objectives of this effort were to: 1) adapt and optimize existing key structural composite analytical software for parallel processing on existing and next generation massively parallel computing hardware; 2) integrate the resulting parallel software with various types of computer architectures and user friendly graphical interfaces to achieve portability and increase efficiency as needed to make a viable commercial parallel processing simulation tool for probabilistic analysis of high temperature composite structure; 3) develop an executive controller window architecture that will best suit the I/O requirements of the user as well the GENOA solver modules; 4) demonstrate the key advantages of the newly developed GENOA software system; and 5) seek industry partnership for commercialized application.

In achieving these objectives, a detailed study was conducted to acquire and demonstrate the various aspects of integration methodology associated with the parallel software development effort. Section 3 focuses on the individual technical areas investigated to provide enabling computational capabilities for performing: 1) finite element-based structural analysis utilizing composite micromechanics; 2) probabilistic simulations of material strength and structural reliability; 3) structural domain decomposition and parallelization of sequential algorithms. and 4) multidisciplinary design optimization to reduce the total CPU time.

Section 4 describes the innovative methods of integrating the codes reviewed in Section 3 and details several algorithms used to translate data from various commercial CAD/CAM software programs.

Sections 5 through 7 describe the key enabling technologies that were integrated for implementation and optimization of parallel code performance on massively parallel hardware. These sections assess the programmatic strategies necessary for successful development of a commercially viable parallelized software package.

In Section 8, several example problems demonstrate the development efforts described in Sections 3 through 7.

Appendix A provides a thorough market analysis evaluating the demand for the GENOA software development and information for marketing.

Our work with the HITCAN program provided some very promising results. After reviewing the methodologies and computational techniques embodied by this code, several conclusions were made regarding its potential integration as a baseline analytical capability. First, the innovative architecture of the program provides a unique opportunity to examine the highly coupled behavior of both the structure and constituent composite materials response. Through its nonlinear iterative procedures and extended levels of stress resolution, numerous metallurgical and structural uncertainties were incorporated for global probabilistic structural response simulation. Secondly, the computationally intense nature of the HITCAN program offers an excellent scenario for demonstrating the advantages of using the domain decomposition and a real time dynamic load balancing system for parallel implementation. Although most of the efforts with this code indicate that it is the best suited for GENOA application, there were some modifications required to increase commercial viability and competitiveness with other finite element programs. These modifications allowed: 1) development of a mixed element capability to better model more complex structures and components; 2) implementation of a modular architecture and dynamic memory allocation needed for large problems , and easy integration of codes such as CEMCAN; 3) implementation of new user interfaces to improve geometry modeling speed, data base and I/O management; and upgrading of data visualization to improve the overall marketability of the code.

Most of our efforts for probabilistic integration concentrated on examining methods of evaluating the effect of uncertainties at the constituent material level. State-of-the-art capability was assured for GENOA by choosing simulation techniques and strength degradation methods like those implemented by programs such as the Integrated Probabilistic Assessment of Composite Strength (IPACS) at NASA/LeRC. In addition, the integration of PROMISS with HITCAN provides a demonstration of the ease of incorporation of state-of-the-art probabilistic programs into GENOA.

Aside from some minor porting difficulties which were easily surmounted, the integrated version of the PROMISS code performed extraordinarily well in several demonstration problems presented in Section 8. The use of the code required a minimum number of assumptions to be made for the distributions and standard deviations of the primitive variables...

In Sections 5 and 6, several innovative methods are presented for parallel implementation. techniques These methods focus on the application of the parallelism to finite element analysis, material property calculation, and probabilistic micromechanics.

In order to port HITCAN, a finite element-based program, to a parallel processing computer, it was necessary to establish the independent tasks that would be performed by each processor and the way data was partitioned and managed by the ensemble of processors. Parallelism was achieved by partitioning the mesh into subdomains each of which was assigned to a separate processor. When a problem domain is decomposed into as many balanced subdomains as the number of processors, it is possible to approach the theoretical speedup. It was determined that the degree of interaction between subdomains depends strongly on the number of interface nodes. On local memory computers, the interface nodes give rise to interprocessor communication, while on shared memory architectures they result in memory contention. Therefore, an essential step in achieving high levels of parallelism is to use an effective integrer optimization partitioning scheme that is capable in real time of balancing subdomains in such a way as to keep the number of interface nodes at a minimum. Successful demonstrations of an integrer optimization partitioning scheme were performed with HITCAN in Section 8.

A cascading processor assignment was developed and demonstrated to enhance the synchronization process for micro mechanics calculations with METCAN. This cascading processor assignment concept allows optimization of processor assignment for groups of processors that are dedicated to various levels of the analysis.

Developing a code specifically for problem solving on a parallel computer requires the selection and understanding of the type of multi-processor architecture to be implemented. After considering the various types of architecture (MIMD , SIMD, and distributed workstations) a modular architecture was developed for GENOA that is optimizes portability to specific hardware or combination of platforms.

After completing the market analysis in Appendix A, it was evident that a new awareness regarding the introduction and utilization of new materials is closely coupled with the development of high speed computers and software. Advances in one will require concurrent efforts in the others. It is also becoming increasingly evident that a new revolution is in progress triggered by the onset of advanced composites with high performance characteristics. GENOA is now available to fill the need for the advanced analytical capability required for realization of the performance benefits provided by these new materials. GENOA's advanced analytical capability, achieved by the development of a parallel PSM code specifically designed for the analysis of advanced composites, is an enabling commercial ready technology needed by government and industry.

Appendix A - Business Plan

Appendix A will be submitted as a separate document.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1996	Final Contractor Report

4. TITLE AND SUBTITLE

Concurrent Probabilistic Simulation of High Temperature Composite Structural Response

5. FUNDING NUMBERS

WU–505–63–5B
C–NAS3–26997 SBIR

6. AUTHOR(S)

Frank Abdi

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Alpha STAR Research Corporation
5200 W. Century Blvd., Suite 340
Los Angeles, California 90045

8. PERFORMING ORGANIZATION REPORT NUMBER

E–10323

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR–198502

11. SUPPLEMENTARY NOTES

Project Manager, Christos C. Chamis, Structures Division, NASA Lewis Research Center, organization code 5200, (216) 433–3252.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Released as Publicly Available July 1996
Subject Category ~~34~~ 24

This publication is available from the NASA Center for AeroSpace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

A computational structural/material analysis and design tool which would meet industry's future demand for expedience and reduced cost is presented. This unique software "GENOA" is dedicated to parallel and high speed analysis to perform probabilistic evaluation of high temperature composite response of aerospace systems. The development is based on detailed integration and modification of diverse fields of specialized analysis techniques and mathematical models to combine their latest innovative capabilities into a commercially viable software package. The technique is specifically designed to exploit the availability of processors to perform computationally intense probabilistic analysis assessing uncertainties in structural reliability analysis and composite micromechanics. The primary objectives which were achieved in performing the development were: 1) Utilization of the power of parallel processing and static/dynamic load balancing optimization to make the complex simulation of structure, material and processing of high temperature composite affordable; 2) computational integration and synchronization of probabilistic mathematics, structural/material mechanics and parallel computing; 3) implementation of an innovative multi-level domain decomposition technique to identify the inherent parallelism, and increasing convergence rates through high- and low-level processor assignment; 4) creating the framework for Portable Paralleled architecture for the machine independent Multi Instruction Multi Data, (MIMD), Single Instruction Multi Data (SIMD), hybrid and distributed workstation type of computers. 5) Market evaluation. The results of Phase II effort provides a good basis for continuation and warrants Phase III government, and industry partnership.

14. SUBJECT TERMS

Computer programs; Composite materials; Probability theory; Multi-level domain-decomposition; Multi-instruction distributed workstations

15. NUMBER OF PAGES

155

16. PRICE CODE

A08

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	