COMPUTATIONAL TECHNIQUES FOR FINITE ELEMENT ANALYSIS

R. J. MELOSH

Engineering Science and Mechanics, College of Engineering,
Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061, U.S.A.

SUMMARY

This paper provides a critical review of computational techniques for finite element analysis and projects the set of ideas which could form the basis for advanced software.

The paper envisions a state-of-the-art computer program which is a realization of the better ideas described in the literature. In the course of this development it reviews documented techniques for expediting analysis, controlling accuracy, and automating tedious tasks of numerical analysis in structural mechanics.

It considers computational methods for automating development of mesh data, establishing the sequence of equation treatment; developing element models; predicting equilibrium, resonance, and transient behavior; evaluating secondary unknowns; and describing analysis results. It encompasses "exact" analysis procedures and describes approximate methods successfully used in reducing computer processing costs.

In discussing the collage of ideas, it identifies omissions of valuable computation techniques. These pertain to insuring meeting the user's objectives of relevance of the analysis and automation of computational details.

The paper concludes by describing available but unexploited ideas for an advanced set of finite element analysis computer software. The ideas center around the concept of self-validation of simulations and automatic control of discretization error. Incorporation of these ideas is viewed as the necessary step in obtaining the next order of magnitude increase in application of finite element technology.
1. Introduction

The success of the finite element concept as a basis for analysis of engineering systems is due, in large measure, to the high speed digital computer and its software.

Use of the concept is not new. Its first applications, predicting deflections and vibration of trusses, may be set in the seventeenth century. What is new is extension of the concept to continuous systems. The extension becomes a viable basis for analysis because the computer exists. Since the computer is slave to its programming, the key to execution of a finite element analysis is the collection of ideas encapsulated in the program.

This paper surveys calculation processes and data management techniques that have found use in finite element computer programs. It draws on articles from the engineering literature to describe the scope and thrust of logic which is directing numerical analysis by the finite element approach.

In deducing trends in computational techniques, it proceeds chronologically. The next section establishes the simple skeleton which was the basis for computer logic in implementations around 1960. The third section scans variations and extensions of the last decade. The fourth section assesses these developments, distilling and extrapolating to envision computer techniques that will form the basis for software of the 1970's.

Representative references conclude the paper. These are grouped in accordance with their most relevant content for this paper. Articles of each group are ordered by the year of publication, affording an overview of the chronological evolution of techniques. Many additional references appear in the survey articles.

2. Basic Computational Techniques

To establish a datum for measuring the advances in computational techniques and to identify those ideas which are "basic" because they persist in current software, this section reviews methods of the first analysis implementing programs. These programs became operational about 1960 in the United States aircraft industry.

This early software, by current standards, was limited in scope and placed high demands on the user's understanding. Often the user prepared more items of input than he received as printed output. The interpretation of output involved the tedious task of scanning pages of numbers arranged in relatively poorly labelled columns.

Despite these shortcomings, early codes [143], [144], [145], [146], [150], [151], [152] offered unique and appreciated capabilities. The primary objective of code development was to simulate linear structural systems under static and dynamic loading. The first code [146] encompassed planar membranes. Programs for frame and wing analysis were also developed.

Not only were programs limited in technical scope, but each had a limited number of users and limited problem size. Often the program developer and principal user were the same engineer. Only company proprietary codes existed. Though problem size (number of equations) was limited by small cores (4096 words) and use of drum storage only for problem data, tape storage of intermediate results was adopted early.

The basic computational techniques centered on using sparse matrix arithmetic and storage, exploiting matrix symmetry, and employing the modularity of an element-by-element treatment. These techniques were implemented by storing and manipulating only non-zero
terms of system matrices, generating and storing only triangular matrices for symmetric ones, and generating coefficient matrices and "stresses" an element at a time.

Figure 1 shows the major program steps of finite element analysis. As embodied in early codes, each step involved a group of subroutines; each group performing generation, assembly, transformations, equation solving, and "stress" evaluations in turn.

As suggested by the figure, generation involved reading element input data; generating stiffness, mass, stress, and loading element matrices; and deleting rows and columns of these matrices to imply boundary conditions. Assembly involved separately forming system matrices to represent stiffness, mass, and loading. Transformations of equations to canonical form involved quasi-matrix operations. Equation solving included capabilities to solve simultaneous equations, solve real value eigenproblems and predict transient response. Transformation of results included evaluation of element joint generalized forces, or internal loads per inch of thickness, reading stress matrices from peripheral storage.

Further details on how these functions were performed are summarized in TABLE I. This table lists the groups of computational subroutines, approaches and algorithms of analysis, ideas employed to reduce computer time (by reducing calculations) and storage requirements, and checks made to insure accuracy.

These data show that sparse matrix techniques [91], [100] were the major basis for reducing calculation time and storage space in these programs. Matrix operations - multiplication, addition and solving simultaneous equations were performed manipulating sparse matrices. These matrices were stored in core and on tape, omitting zero coefficients.

Figure 2 [91] exhibits the merit of sparse matrix arithmetic for matrix multiplication. The upper curve establishes the relative calculation time for full matrices multiplied using index registers in a triple DO loop. The lower curve defines the time for matrix multiplication in which coefficient row-column codes are matched to direct coefficient multiplication.

The full matrix curve shows a maximum of a ten percent reduction in computer time due to faster multiplications when zero coefficients are involved. The dashed curve indicates increasing savings in computer time as sparseness increases. Only when matrices become more than 94 percent populated does index register arithmetic have an advantage over code matching.

Figure 3 illustrates the savings in storage using sparse matrix techniques. The continuous curve show the relative storage requirements for full matrices. The long-dashed curve defines the storage when each coefficient is labeled by a code. The short-dashed curve represents storage requirements for the matrices when stored in strings of non-zero coefficients. This curve will range between the long-dashed and solid curves as one limit and a straight line joining points A & B as the other - depending on the distribution of non-zero coefficients in the array.

Despite the apparent advantage of string storage, none of the early codes used it exclusively. Generally, coded matrix or string storage was used for storing data on peripheral netics. For in-core data, all three types of storage were used.

One space saving sparse matrix technique introduced was wave front matrix assembly [161]. This technique involves adding element matrices in turn until core storage is full.
# Table 1
INITIATING COMPUTER TECHNIQUES
OF FINITE ELEMENT ANALYSIS

<table>
<thead>
<tr>
<th>Operation</th>
<th>Tasks</th>
<th>Approaches</th>
<th>Algorithms</th>
<th>Time saving</th>
<th>Space saving</th>
<th>Checks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining the problem</td>
<td></td>
<td>User prepared punched</td>
<td>Redundant input data</td>
<td>Modular input</td>
<td>Format</td>
<td></td>
</tr>
<tr>
<td>Generating element matrices</td>
<td></td>
<td>Closed form equations</td>
<td>Interpolating polynomials</td>
<td>Exploit symmetry, Diagonal mass matrix</td>
<td>Elastic coefficient matrix definiteness</td>
<td></td>
</tr>
<tr>
<td>Transforming equations</td>
<td>Assembling coefficients of eqs.</td>
<td>Sparse matrix union</td>
<td>Wave front addition</td>
<td>Triangular matrix assembly</td>
<td>Packed matrix storage</td>
<td>None</td>
</tr>
<tr>
<td>Matrix transformation</td>
<td>Condensation multiplication and transposition</td>
<td>Direct reduction and multiplication, implied transpose</td>
<td>Full and sparse matrix arithmetic</td>
<td>Packed matrix in block storage</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Finding stresses</td>
<td>Evaluate generalized forces</td>
<td>Non-conforming matrices multiply</td>
<td>Reuse element matrices</td>
<td>Modular treatment by element</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Solving system equations</td>
<td>Solving simultaneous equations</td>
<td>Decomposition</td>
<td>Choleski</td>
<td>Skip zero low multiply</td>
<td>Variable hand storage</td>
<td>Positive definiteness</td>
</tr>
<tr>
<td>Obtaining eigenvalues</td>
<td>Matrix iteration</td>
<td>Power method with deflation</td>
<td>Aitken acceleration few mass points</td>
<td>Unpacked matrix storage few mass points</td>
<td>Orthogonality of vectors</td>
<td>Closure</td>
</tr>
<tr>
<td>Transient response</td>
<td>Numerical integration of coupled eqs.</td>
<td>Runge-Kutta, Adams-Moulton</td>
<td>Fixed time step in-core operation</td>
<td>Unpacked storage, Diagonal mass and damping</td>
<td>Repeat with smaller steps</td>
<td></td>
</tr>
<tr>
<td>Nonlinear response</td>
<td>Quasi-linear steps</td>
<td>Choleski</td>
<td>Reuse zero stress stiffness keep problem small</td>
<td>Exploit symmetry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transcribing output</td>
<td>Tabular printout</td>
<td>Coded and full matrices</td>
<td>Blocked data</td>
<td>Modular transfers</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

M 4/
Then, as additional element matrices require additional storage, coefficients with the lowest numbered joints are written on tape to make room. This technique allows assembling large order matrices without backspacing tape or partitioning.

Other sparse matrix computational techniques were introduced in equation solving. [116], [117], [118], [146] Matrix "bandedness" and overlay characteristics were exploited to reduce core storage demands.

Figure 4 illustrates the band concept. Using the displacement approach, non-zero coefficients can couple joint variables no more than one element away from a joint. Thus, as the example shows, the non-zero coefficients will fall within an envelope embracing the main diagonal. The "band" of a matrix is the number of columns, to the right of the diagonal, that are included to span all non-zero coefficients. The envelope includes all non-zero rows coefficients under the restriction that the envelope band cannot decrease by more than 1 for two successive rows.

Assuming the matrix could be advantageously banded, early codes stored the matrix in core a string for a row, to conserve core storage. Programs using the maximum bandwidth of the matrix were prevalent, variable bandwidth programs also were developed.

Since the band concept is consistent with overlaying, early codes used this idea to save core space. Overlaying involves the thought that the envelope for non-zero coefficients is the same for the matrix as for its triangularized form in equation solving. Thus the coefficients of the triangular matrix replace those of the original matrix without additional storage allocation.

Early programs were oriented toward exploiting matrix symmetry. It is characteristic of the finite element approach that structural stiffness and mass matrices are symmetric. Thus, these system matrices were represented by only a triangular set of coefficients.

Solution algorithms were also chosen to exploit symmetry. TABLE II shows this bias for simultaneous equations and for the eigenproblem. The simultaneous equations naturally involved a symmetric matrix. The eigenproblem equations were transformed to involve a symmetric dynamic matrix. The transformation matrix was generated working with the symmetric stiffness (or sometimes mass) matrix. The power method, with deflation, was most popular for spectral analysis, storing the triangular representation of the dynamic matrix in core in variable band form. Transformation of equations of motion to a double-order first order set was performed in core. These calculations were performed using full matrices since few equations of motion were treated (40).

To conserve limited in-core storage space, elements calculations were performed an element at a time. In generating element matrices, only input data which involved the current element was resident. In evaluating generalized forces for each element, previously generated matrices were read from peripheral storage element-by-element.

Use of these programs [146], [146] provided data vital to the growth of finite element application knowledge that the approach could lead to accurate simulations in short wall clock times and without exorbitant precision or computer costs. Problems with up to about 600 equations were not uncommon. Results were accurate within five per cent for deflections and ten percent on stresses.
3. Advanced Computation Techniques

The last decade has been an extremely fruitful period in development of computational techniques for finite element analysis. The growth of problem sizes to thousands of degrees of freedom evoked new methods for handling input and output. It incited study of manipulation errors. The probing reexamination of data management needs by hundreds of new program developers yielded data on alternatives and established methods of improved efficiency. Demands for extended structural simulation capabilities prompted creation of new computer software using new techniques.

This section will survey the advances, grouping the ideas to correspond with the groups of TABLE I. Though this section represents automatic data generation as the most dramatic advance, the reader will observe the profound increase in the breadth and complexity of finite element analysis implementation.

Problem Definition.

As the number of joints and elements for a a system increased, the amount of input required increased. Thus, it became evident that much of the input preparation work should be automated. Engineers addressed development of algorithms to generate joint and element, data (mesh generation), and sequence joint numbers from data defining boundary geometry and mesh construction policies. These "data generators" constitute one of the most significant computational advances for finite element analysis.

The evolution of mesh generators is a microcosm of the advance of a technology. Figure 5 illustrates this advances viewed chronologically from the finite element literature.
Interpolation algorithms were first documented about 1965. [21], [3]. That of Fig.
5a. [3], is based on partitioning the region by straight lines and arcs of circles. The
occurrence of triangles with small angles (equilateral triangles involve arcs of circles). The
slow grading of the mesh using this algorithm was improved by interpolations
based on isoparametric interpolation functions about 1969 [6], [10], [16], [23] as
suggested by the mesh of Fig. 5b. This mesh was developed by dividing the region into
three trianally shaped subdomains and subdividing by isoparametric interpolation.

More recently, mesh generation based on conformal mapping has been reported [11], [24].
This elegant (and computationally costly) process often yields long-sided triangles as
suggested by the mesh of Fig. 5c. These generators also use subdomains to accommodate
regions of high mesh density. Despite concern over mesh regularity in developing the
algorithm, this method develops capricious meshes.

Remapping the boundary points of an equilateral mesh is another simple but useful
approach [14]. This yields irregularly shaped triangles only near the boundaries as
shown in Fig. 5 d.

The construction of a model of the structural continuum by aggrandizing triangular
regions is an approach which tends to reduce the number of subregions which are
irregularly divided. [13], [15]. Fig. 5e represents this approach. This technique leads
to smoothly graded meshes with well-shaped triangles.

The concurrent publishing of similar ideas is common in technology. Also common is
the lack of knowledge of related work. Developers of finite difference computer software
in 1964 described a simple low-cost process for avoiding unwanted mesh irregularities
[1], [8]. The concept is simply to iteratively reposition joints of the mesh. The basis
chosen was to joint positions as potentials in the La Place equation. Exercising a few
cycles of relaxation smooths the mesh. To avoid having mesh lines fall outside the
boundary for the reentrant corner, the regions must be divided into subdomains as
suggested by the continuous line of Fig. 5f. The approach can treat any element topology
and provides graded meshes by fixing boundary joints. This iterative improvement approach
is superior to the use of interactive graphics for mesh improvement[12].

Though the representations of mesh generators in Fig. 5 are all for triangulated
domains, the more recent generators can treat rectangular meshes.

Given a mesh, two different approaches are available to sequence joint numbers.
Both have the objective of minimizing the maximum bandwidth of stiffness or mass matrices
[21], [15], [4], [7], [20]. Both seek a relative minimum. The first approach [4],
develops a mask of the system matrix and interchanges rows and columns iteratively,
reducing the maximum bandwidth in each step until no more improvement is possible. The
second approach [7], [20] uses vectors to represent joint connectively. It successively
(and rapidly) defines the next joint to take to minimize the growth of bandwidth. Though
these algorithms tend to minimize maximum bandwidth (and hence storage requirements)
they are easily modified to minimize the sum of the row bandwidths squared - the measure
of the number of calculations for triangularizing the matrix.

TABLE III summarizes computational techniques used in implementing mesh generation,
improvement, and joint sequencing algorithms. The most significant time saving devices
are the selection of a relative minimum as the objective an the spanning tree method of
joint sequencing. Direct use of connectivity data with this method is also the most
<table>
<thead>
<tr>
<th>Tasks</th>
<th>Approaches</th>
<th>Algorithms</th>
<th>Times Saving</th>
<th>Space Saving</th>
<th>Checks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generating joint and element data</td>
<td>Distributing joints from user defined mesh constraints</td>
<td>Interpolation Mapping Construction</td>
<td>Slab generation Single type of element topology</td>
<td>Input completeness consistency magnitudes Plot feedback</td>
<td></td>
</tr>
<tr>
<td>Improving mesh grading</td>
<td>Relaxation solution of equations of physics</td>
<td>La Place equation solution</td>
<td>Over-relaxation</td>
<td>One coordinate at a time</td>
<td>Change of joint position in a cycle</td>
</tr>
<tr>
<td>Sequencing degrees of freedom</td>
<td>Interactive graphics</td>
<td>(User generated)</td>
<td>Iterative Interchange</td>
<td>Sub-domain treatment</td>
<td>PLOT feedback</td>
</tr>
<tr>
<td></td>
<td>Direct row and column interchanges</td>
<td></td>
<td>Seek only relative minimum</td>
<td>Bit manipulation of connectivity coefficients</td>
<td>Stop when no more changes useful</td>
</tr>
<tr>
<td></td>
<td>Allocation numbering</td>
<td>Spanning tree</td>
<td>Implicit Interchanges</td>
<td>Use connection data directly</td>
<td>Start with different joint</td>
</tr>
</tbody>
</table>

**TABLE III**
ADVANCED TECHNIQUES: PROBLEM DEFINITION

**TABLE IV**
ADVANCED TECHNIQUES: ELEMENT MATRIX GENERATION

<table>
<thead>
<tr>
<th>Class</th>
<th>Element family*</th>
<th>Computation demands</th>
<th>Computational advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed form equations</td>
<td>Simple [26],[27], [31],[33],[37], [39],[42],[45], [47],[46],[57], [68]</td>
<td>Evaluate formulas by simple operations</td>
<td>Equations for displacements directly by Laplace and Hermite polynomials</td>
</tr>
<tr>
<td></td>
<td>Frequency dependent [42]</td>
<td>Need to know system frequency to develop coefficient</td>
<td>Large elements acceptable since shape function exact</td>
</tr>
<tr>
<td></td>
<td>Elasto-plastic [23],[36],[51], [52],[67]</td>
<td>Requires stress distribution to develop coefficients</td>
<td>Stress distributions uniquely defined</td>
</tr>
<tr>
<td>Series expansion</td>
<td>Fourier series [28],[34],</td>
<td>Evaluate sines and cosines for specified harmonics</td>
<td>Permits separate simplified analysis for each harmonic for a closed shell</td>
</tr>
<tr>
<td></td>
<td>Bessel series [30]</td>
<td>Evaluate Bessel series terms until no significant change in coefficient</td>
<td>Large elements acceptable since shape exact for a cylindrical shell</td>
</tr>
<tr>
<td></td>
<td>Power series [73]</td>
<td>Evaluate power series terms until little change in matrix coefficients</td>
<td>Optimizes element model so fewer elements are required</td>
</tr>
<tr>
<td>Numerical integration</td>
<td>Isoparametrics [39],[38],[62]</td>
<td>Table look-up for integration point locations, weightings and kernel matrix coefficients</td>
<td>Natural coordinates permit accurate fit of geometry and Gauss Integration provides efficient representation of higher order polynomials</td>
</tr>
<tr>
<td></td>
<td>Finite differences [46]</td>
<td>Requires numerical analysis a substructure to evaluate matrix coefficients</td>
<td>A single set of logic can develop matrices for any element shape and number of joints</td>
</tr>
</tbody>
</table>

* References indicated are representative.
significant storage space saving technique.

Another valuable evolutionary development involves extensive input data checks. These cost effective checks encompass checks of input completeness, consistency and number magnitudes. Both numerical checks and plots of input are available to protect against pointless calculations.

Automatic data generators have already drastically reduced the amount of tedious work required to define a finite element problem for computer analysis. They are an integral part of most of the leased programs.

Element Matrix Generation

More than one hundred element models are available as a consequence of development work of the last decade. Generally, each element inventor claims particular calculation saving advantages for his element. Most suggest their element offers greater accuracy for a given number of elements. Since such a claim is generally unprovable, the number of elements can be expected to continue to grow.

From a computational viewpoint, these elements fall into three classes depending on how matrix coefficients are evaluated: closed form, series expansion, and numerical integration. TABLE IV lists these classes and provides representatives of each family of element models. It also indicates the computational demands for generating mature coefficients and some advantages of each family relative to the others.

Of the element families, the Fourier series and isoparametric offer the most promising computational techniques. The Fourier series models may involve response synthesis, analyzing the effect of each harmonic separately [148]. Thus only as many harmonics need be included as represent the loading. A complete segment of a closed shell or solid of revolution may be simulated as precisely as desired with the same relatively small number of equations per harmonic.

Since the isoparametric elements can closely match boundary contours, their use avoids some geometric error. In addition, the economy afforded in matrix generation by preformed kernel element matrices [74] can result in less time for developing element coefficients than for comparable simple elements.

The Gauss integration elements share with mass and damping matrices, the computational advantage of being able to relax the potential energy strictures to reduce calculations and improve results. In Gauss integration elements, experiments show that use of fewer Gauss integration points is consistent with the polynomial order results in reducing stiffness. An error study shows [65] that as long as the number of integration points is greater than a minimum number (a function of the polynomial order), numerical stability will be retained.

Diagonal mass matrices must result in lower frequencies than potential energy "consistent matrices. Similarly introducing approximations to decrease band widths of damping matrices can be expected to yield computational advantages [67].

For any element matrix implying rigid body modes, development of matrices by first developing the elastic submatrix reduces calculations. For a planar beam with four degrees of freedom (normal and angular displacement at each end) this technique engenders developing a 3 x 3 stiffness matrix in global coordinates. The complete 6 x 6 matrix is constructed by adding rigid body modes.

Not only does this approach provide a more adaptable computer code and permit faithful
representation of equilibrium conditions, but thereby also it avoids an unbounded manipulation error [70]. This error is induced by round-off in the generation process.

Figure 6 shows the growth of this error with the number of equations for a cantilevered beam. It indicates that if this error occurs for each beam element for a 27 bit mantissa computer, the error will intrude into the third significant digit of deflections with about 400 equations.

Another useful computational device is to reuse element matrices when structure is replicated. This is made possible by the fact that element matrices are invariant with coordinate translations and rotations. Thus, for example the same triangular membrane element matrix, in local coordinates, can be used for all similar triangular membranes, referenced to their local coordinates.

This idea is particularly useful for simulating three dimensional solids. Then it is convenient to use many elements of the same shape. Then also a large number of calculations are required per element so the payoff is greater. This computation device is an integral part of some programs as well as being available as a special option in others [160], [171].

Advanced computational techniques for element matrix generation arise out of particular features of new elements. In addition, elastic submatrix generation and replication offer advantages for all element classes.

Transformations

Advanced techniques in performing the operations preparatory to solving the equations provides some important changes in procedures. More substantitive, however, are the collection of ideas for implementing substructuring and rapid reanalysis. Substructuring capabilities were prompted by the operational need to divide large systems into parts that are manageable by one person. Rapid reanalysis methods respond to the need, in design and performance studies, to perform many analyses of closely related systems.

TABLE V summarizes the advanced computational features of transformation operations. The table shows that using sparse matrix operations and exploiting symmetry are the major computer saving techniques.

Assembly of element matrices and multiplications involve few advances. Integration of multiplication and addition and multiplication and transposing into single integrated operations saves computer time by reducing the number of data transmittals. String storage of all array data, as suggested by Fig. 3 curves, is a natural advance. [152],[162]

Substructuring capabilities are a major addition to finite element computer technology. The idea is to treat parts of the system (Substructures) independently as long as possible in the analysis to facilitate piece-wise checkout of problem descriptive data and interpretation of results. [80],[86],[93],[102]

Steps of a substructure analysis include:

1. Developing system matrices of substructures.
2. Decoupling equilibrium equations, for joints not on the interface between substructures, from the rest of the equations of the substructure by row operations. (This operation is often referred to as "condensation").
3. Combining interface equations from all substructures and solving these to determine displacements at the interfaces.
4. Returning to the substructure equations to evaluate displacements at interior joints and element stresses.

In its simplest form, substructuring consists of condensing out the equations for the interior point of a square crossed by two diagonals. In its advanced form it involves the treatment of substructures, each with hundreds of degree of freedom. [83],[106],[158],[161],[163]

Use of linear constraints to reduce the number of degrees of freedom at the interface of substructures is a technique for reducing calculations in the analysis of multi-equation models [171]. This has application to dynamics problems as well in reducing the number of dynamic degrees of freedom. [81],[82]. Though computationally attractive, disciplined approximations are not well known [97].

Advances appear in procedures for treating displacement boundary conditions. Imposition of prescribed displacements as part of the equation solving process eliminates some data transmittals. [128] Performing a congruent transformation on a substructure matrix permits imposing linear constraints with relatively little data transmittals and fewer calculations matrix multiplications with the whole system matrix.

The literature also describes several new techniques for stress evaluation. Regeneration of stress coefficients is now known to require less calculation time than writing and reading stress matrices from peripheral storage for several elements. Sequencing elements to reduce search time is available to speed stress evaluation [133].

Attacks on the troublesome problem of obtaining accurate stresses yielded other techniques. Curve fitting now is expected to be less satisfactory than use of conjugate functions for determining stress [84], [87], [96], [105]. Experiments in sensing the potentially unbounded errors in evaluating stresses illustrated that despite failure to satisfy joint equilibrium near supports, accurate peak stresses can be obtained. Thus, the value of error measures in reducing computer time by saving "apparently" inaccurate analyses was shown. [134]

Computer software also contains rapid exact and approximate algorithms for multiple-configuration analyses. The basis for exact analysis is to superimpose changes on the original solution and thereby save calculations and storage space. [90],[92],[95],[99],[101]. In procedures in which the changes can be represented by few vectors, calculation savings may be orders of magnitude less than first-time analysis. The recombination-decomposition algorithm is useful when many changes are made. It reduces calculations when less than half of the last equations are changed. A combination of these two procedures insures that reanalysis calculations, no matter how many changes are made can require only about one third the number of first time calculations.

Approximate multiple configuration analysis methods are usually iterative. [88],[95],[104]. They aspire to reduced calculations by starting the iteration with the old solution as a guess.

Advances in transformations reflect the maturing and broadening of the base for finite element data processing. Vertical integration of operations is an important ingredient in the advance. Substructuring and rapid reanalysis extensions, however, provide new analysis scope.
Equation Solving

Ten years of active study of ways of implementing solution of simulation equations has yielded many advances. Separate examinations of alternative solution algorithms has provided a variety of data processing techniques for each solution objective. Experience in solving equations for a variety of computers and applications has brought forth useful methods for improving processing efficiency. Theoretical and experimental error analysis has identified sources of critical manipulation error.

TABLE VI summarizes these advances. It divides each equation solving task into different data processing approaches. It lists some of the algorithms for implementing finite element analysis and cites their data processing features.

Most emphasis is on the task of solving linear simultaneous equations. Results of progress in both direct and iterative approaches to solution strengthened technology for solving large (many more coefficients than can be contained in core) sets of equations. Many ideas are available for implementing direct methods [116], [118], [121], [128], [130], [133]. The most significant involves reducing calculations by recognizing zeros included within band envelope. In figure 4, these are zeros like that in column 4, rows 1, 2, 5, and 6.

Wavefront data processing [128], [130], saves both calculations and computer storage space over variable band processing. The idea is to allocate storage space only when required by non-zero row coefficients in the decomposition process. Thus, for the problem of Fig. 4, when rows for joint 1 are in core, core space is only required for joints 1, 2, 5, and 6. When joint 2 is treated, the decomposition for the equations of joint 1 are complete. Thus the wavefront includes only joints 2, 3, 5, 6, and 7.

Experience shows wavefront processing reduces equation solving computer time 30-60 percent below times for bandwidth processing. It is known that the number of calculations for wavefront processing cannot exceed those of band. Smaller storage space requirements also permit solving sets of equations with bigger wavefronts without spilling core. To exploit the full potential of wavefront treatment changes to optimum joint sequencers make them address minimizing wavefronts.

Error analyses reveal numerical singularity error as the most significant source of manipulation error [108], [109], [112], [134], [137] Figure 6 shows how this error limits problem size for a prismatic cantilevered beam with equal stiffness elements. The boundary defines when solution results consist only of round-off errors.

The curve of Fig. 6 shows that given enough bits of precision, numerical singularity need not limit analysis accuracy. The error study, however, determines that the error source can be eliminated by optimizing the sequence of arithmetic. This is essentially effectuated by sequencing the joints starting with the most flexible part of the system.

Vertical integration of the equation solving algorithm represents another advance in finite element implementations. This idea involves integrating the steps of solution to reduce transmissions to and from peripheral storage units.

TABLE VII shows how vertical integration reduces the number of data transmissions from nine to six. The integration reduces data transmissions without any penalty in the number of calculations. However, integrating decomposition and forward substitution tasks reduces the band envelope size that can be manipulated without spilling core. Thus it is best suited to the single vector loading case.
### TABLE V

**ADVANCED TECHNIQUES: TRANSFORMATIONS**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Approaches</th>
<th>Time-saving</th>
<th>Space-saving</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix-By-Vector Multiplying</td>
<td>Contiguous matrix multiply</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Non-contiguous matrix multiply</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Decomposition</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast matrix multiply</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sparse matrix multiply</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sparse matrix multiply</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### TABLE VI

**ADVANCED TECHNIQUES: EQUATION SOLVING**

<table>
<thead>
<tr>
<th>Task</th>
<th>Approaches</th>
<th>Time-saving</th>
<th>Space-saving</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solving Linear Equations</td>
<td>Direct methods</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gauss-Seidel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gaussian</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Partial Gaussian</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Non-linear</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solving Nonlinear Equations</td>
<td>Newton-Raphson</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Quasi-Newton</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stochastic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calculation of Parameters</td>
<td>Non-iterative</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Iterative</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### TABLE VII

**WEAK TRANSMISSIONS**

<table>
<thead>
<tr>
<th>Weakness Type</th>
<th>Count</th>
<th>Y 1</th>
<th>Y 2</th>
<th>Y 3</th>
<th>Y 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic errors</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Assembly</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Decompression</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Normal Substitution</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Local Substitution</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>9</td>
<td>7</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

*Y* = one transmission, comparable to all the data for a system matrix, to or from a peripheral storage unit.

**Questions for back testing and reading are works in the operation.
With the introduction of computer hardware which buffers input and output, concern
for the number of data transmissions has become less critical. Thus, explorations of
iterative solution methods have yielded knowledge on their implementation features [110],
[119], [121], [131].

There is no convincing evidence that iterative techniques are superior to direct
methods. Figure 7, for example, permits comparing a number of iterative methods with Gauss
elimination for a particular problem. This suggests that much better iterative methods must
be developed if this approach is to be a viable alternative for solving linear equations.

Recognizing this, investigators directed their attention toward hybrid algorithms.
The intent is to reduce the number of iterations by treating the equations in partitioned
form [126]. Partition sizes are fixed by core storage capacity. The approach has proven
effective in reducing iterations.

The activity in advancing nonlinear analysis computer implementation has involved gain-
ing experience with available algorithms in the structural simulation context. Studies have
encompassed both quasi-linear and gradient methods.

Experience indicates the effective load and Newton Raphson methods are the more
efficient quasi-linear methods [27], [123], [138], [142], [155], [168], [172]. In the
effective load method the tangent stiffness is loaded with a vector representing the real
load and the product of stiffness changes, which depend on the stress and displacement state,
and the estimate of the displacement state. If the change of displacement is sufficiently
small, successively better estimates of the displacement state are attained. The approach
efficiency derives from the ability to perform iterations with a single decomposition.
Since the process may converge to the wrong answer, the advantage only accrues when
accurate error controls on load step size are used. This error control usually consists
of reducing the analysis with different smaller step sizes until no significant changes
occur.

Larger load steps are acceptable when the stiffness coefficients are corrected to
represent nonlinear terms to a first order as in the Newton Raphson method and its
variations. In this method, decomposition of the stiffness matrix is performed in each
iteration.

Direct minimization of the total potential energy provides an alternate way of
solving the nonlinear equations [111], [113], [115], [122], [125]. Experience indicates
that iterative methods with quadratic convergence, such as conjugate gradient [113], are
more efficient than pure gradient methods. The quadratic convergence rate more than
compensates for additional calculations in each cycle to represent the curvature. The
conjugate gradient approach, when scaling is used to improve resolution [125], provides an
algorithm with the important computational advantage that no square (or triangular) matrix
representation of the system need be stored.

Quasi-linear solution methods are the more popular approach to solving non-linear
equations. However, the decade studies suggest that as the number of degrees of freedom
and the importance of nonlinear terms in the equations increase, the popularity of the
direct minimization approach will increase.

Identifying two classes of spectral analysis for structures has resulted in improved
computational efficiency. One class consists of eigen problems which require evaluation
of only the eigenvectors and values of a few of the lowest frequency modes. This class includes prediction of the critical small deflection buckling load as well as the other class consists of eigenproblems involving evaluation of all spectral data.

A review of the literature suggests that inverse iteration (with shifts) is attractive for the first class of problems and Givens Householder for the second. [140], [141], [154], [162]. Inverse iteration offers the computational advantage of retaining matrix sparsity, symmetry and bandedness regardless of how many eigenvalues are evaluated. In addition, it can use the same decomposition logic as used for solving linear simultaneous equations. Givens Householder implementations have demonstrated the ability to get accurate spectral data for systems up to 1500 order. Besides being twenty times more efficient than Jacobi, it has survived assaults by proponents of the QR algorithm and algorithms based on Sturm sequence properties for the second problem class.

Truncation techniques and dual matrix iteration are two advances which provide storage space saving potential for the eigenproblem. The modal synthesis technique [136] involves deleting high frequency generalized coordinates on the substructure basis. Dual matrix iteration implies transformation to canonical form as part of the iterative process [139]. If both the mass and stiffness matrices are non-diagonal, this technique can reduce calculations for class one problems.

Advances in predicting transient response center on studies of available methods to establish ones which are computationally advantageous. [107], [124], [132], [135]. "Exact" integration of uncoupled equations, numerical integration of coupled equations using finite difference operators in the time coordinate and minimization of the potential energy are the three distinct approaches.

Exact integration involves transforming the equations of motion to modal coordinates to result in a set of uncoupled equations. The forcing functions are then synthesized from simple exactly integrable functions. (e.g. circular functions, polynomials) Results are converted back to the original basis for interpretation. This approach provides the highest computational efficiency for the integration operation since each equation is integrated independently and without iteration. It involves the fewest calculations compared with other approaches, when the calculations extracting resonant modes and frequencies will be amortized over many time steps.

Solution of coupled differential equations is a classical applied mathematics problem for which many algorithms are available for solution. Study of these algorithms in the last decade suggest those of TABLE 5. The principle computational advance is based on performing numerical integration for a constant time step. Then the unconditionally stable operations of implicit methods can be used without the need to triangularize for each time step.

One technique for reducing calculations for transient analysis is to use curve fitting and extrapolation of responses with relatively large but stable time steps. Figure 8 illustrates the idea for a one degree of freedom system. The continuous curve is the exact solution. The two dashed curves are the solutions for steps of about 1/6 and 1/5 of the shortest system period. These curves exhibit the lower frequency response characteristic of approximate solutions. Exponential extrapolation at constant interpolated displacement produces the results of the dotted curve in less than half the computer time to get comparable results with a smaller time step.
Recent work discusses solving the equations of motion using Hamilton's integral principle. [43] From a computational viewpoint, the advantage this may provide is a reduction in the number of time steps to cover a given time period.

Transcribing Output

Significant advances in transcribing output reside in new hardware and software yielding graphical displays. Software producing printer plots, precise pen plots, electrostatic plotting, and photograph of CRT plots on appropriate hardware now provides X-Y, contour, and stereographic plots of input geometry and/or predicted behavior. [182] - [194].

The techniques used are generally not peculiar to finite element simulation and will not be described here. It is only noted that since the module for plotting is usually a line segment, the association of response with elements of the system, as in the finite element approach, makes generation of plot logic easy.

Though most plots are the result of mechanizing traditional plotting, explorations of the potentialities of color plots have introduced new computational techniques. [189], [196]. Use of color tones to describe stress or displacement intensities, and to add depth discrimination to orthographic projections in snapshot and movie picture modes promises to eliminate much of the previous clutter on graphs and thereby simplify data interpretation.

Computer Cost Saving Techniques

TABLE VIII lists, in the order of importance, the more significant computer cost saving techniques associated with finite element analyses. Use of all these techniques reduces computer costs about twenty times over not using them. The list includes only techniques which do not effect accuracy.

It's surprising that two of the more valuable techniques are not more widely exploited. One of these involves solving smaller problems and extrapolating to estimate results of the infinitely fine mesh. This technique is important because it reduces computer costs for all phases of data processing. The second neglected technique involves developing and exploiting regular meshes. Data generators are available to construct regular meshes [13], and the transfer method to simplify their analysis [77], [78], [79]. These only await integration into more programs to realize significant cost savings.

The table omits the important techniques of automatic data generation and graphical output. These, of course, increase computer costs. Savings in engineering labor and schedule time justify these techniques.

4. Projected Techniques

Finite element computer implementation technology will continue its rapid growth. It will continue because of the desire of engineers to use this numerical analysis approach for a broadening class of problems, because of the inherent interest of developers of improving the program breed, and because of the changing computer environment. Most of all, it will continue because the potential applications for finite element codes are at least ten times those currently encompassed.

To identify new techniques, this section will review the demands implied by recent research. It will extrapolate development trends to suggest techniques that may be used to satisfy the new requirements.
### TABLE VIII
PRINCIPLE FINITE ELEMENT ANALYSIS
TECHNIQUE FOR REDUCING COMPUTER COSTS

<table>
<thead>
<tr>
<th>Priority</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sparse matrix arithmetic</td>
</tr>
<tr>
<td>2</td>
<td>Solving small problems and extrapolating</td>
</tr>
<tr>
<td>3</td>
<td>Regularization of discretization</td>
</tr>
<tr>
<td>4</td>
<td>Exploiting symmetry in generation and storage</td>
</tr>
<tr>
<td>5</td>
<td>Wavefront processing in assembly and solution</td>
</tr>
<tr>
<td>6</td>
<td>Vertical integration of operations</td>
</tr>
<tr>
<td>7</td>
<td>Avoiding iteration and extrapolating if iteration used</td>
</tr>
<tr>
<td>8</td>
<td>Selecting best-suited algorithms for each task</td>
</tr>
<tr>
<td>9</td>
<td>Storing matrix data in strings</td>
</tr>
</tbody>
</table>

### TABLE IX
PROJECTED TECHNIQUES: NEEDS AND RESPONSES

<table>
<thead>
<tr>
<th>Needs</th>
<th>Applications</th>
<th>Candidate Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. More automated input</td>
<td>ALL</td>
<td>Optimum meshing Automatic element selection</td>
</tr>
<tr>
<td>b. Rapid solution of equations with scattered coupling</td>
<td>Analysis of three-dimensional systems in all disciplines</td>
<td>Hybrid iteration New iteration algorithms</td>
</tr>
<tr>
<td>c. More transportable computer programs</td>
<td>New disciplines for finite elements</td>
<td>Modularized element generation routines</td>
</tr>
<tr>
<td>d. More efficient solution of large sets of equations</td>
<td>Aerodynamics Nuclear diffusion</td>
<td>Vertical integration of processing, Regularization of discretization Boundary solutions, New computer hardware</td>
</tr>
<tr>
<td>e. Treat large variations in mesh spacing</td>
<td>Fracture mechanics</td>
<td>St. Venant principle exploitation, Optimum arithmetic</td>
</tr>
<tr>
<td>f. More rapid nonlinear equation solving</td>
<td>Bioengineering Vehicle crash</td>
<td>Use appropriate approximations, New algorithms, New computer hardware</td>
</tr>
<tr>
<td>g. Improved accuracy control</td>
<td>ALL</td>
<td>Available sensors and corrective techniques Self-validation</td>
</tr>
<tr>
<td>h. More intelligible output</td>
<td>ALL</td>
<td>Colored graphical displays</td>
</tr>
</tbody>
</table>
The paragraphs that follow describe the demands and expected responding computer techniques:

a. **More automated input.** Though data generators have improved rapidly, there is still room for improvement. Recent research, [22], [25], sets up bases for constructing near optimum meshes rather than just nice looking ones. Technology for optimum element selection can be constructed on the same basis. Thus, logic which eliminates the need for the analyst to define discretization beyond geometric description and response or load points of interest is on the horizon.

b. **Efficient solution algorithms for solving equations without enveloped coefficients.** The large envelopes of three-dimensional continua have already reinstigated study of iterative solution algorithms to avoid the spill problem [33]. Continued research will investigate a number of alternative algorithms both of the hybrid and pace types, in an attempt to more fully realize the potential of iterative processes. Major improvements will be prompted by broad accessibility of parallel processing computers.

c. **More transportable computer programs.** Computer programs of the near future will draw the current concepts of "general purpose," "Modularity," and "transportability." Recent research demonstrates the suitability of the finite element steps of analysis of structures for problems in heat conduction, fluid flow, electromagnetic wave propagation, aerodynamics, coupled electro-fluid flow, plasma stability, nuclear diffusion and wave motion in solids and fluids [41], [49], [50], [53], [54], [55], [56], [59], [64], [66], [69], [72], [74], [75]. General purpose will come to mean across disciplines. Modules will be stripped of structural approximations so they will be useful for all disciplines. Separate element generation subroutines will involve the interpolation (shape) functions and each of the differential or integral equation interface of interest. Transportability will mean across computers and user disciplines.

d. **More efficient solution of large sets of equations.** The persistent pressure for using more degrees of freedom in modeling a system will be increased when those who customarily deal with more complex systems of equations, such as nuclear engineers and weather forecasters, add their voices to structural engineers. Since the problem has been persistent, no dramatic improvements can be expected until new computer hardware arrives. Look for new vertically integrated solution software and greater emphasis on exploiting the savings associated with regularization of the discretizations, and bounding capabilities to become more widely available [31].

e. **Ability to treat large variations in mesh size.** Though those working in fracture mechanics may initiate the thrust toward this ability, it will be an increasingly acute need for structural engineers. The obvious, and expensive solution of multiple-precision arithmetic will probably not be resorted to until other avenues have been explored. St. Venant's principle and optimum arithmetic [129] may be the basis for attaining the ability at relatively low cost penalties.

f. **More rapid nonlinear equation solving techniques.** Bioengineering and crash safety applications incite this demand. Current trends suggest the development of special purpose computer programs to fill the need. As more and more applications are made, groups of analysis approximations will become imbedded in these codes. These approximations offer the greatest potential for reducing the many calculations of nonlinear analysis. The nonlinear transient equation solving process however provides a fertile ground for research.
A number of papers can be expected in this area. The recently published search algorithm of Jacobson-Oksman illustrates one of several unexplored nonlinear solution processes improvements in existing solution processes will follow the basic studies of error growth in transient equation solving [135], [142]. The pipeline computers will, by reducing cost per calculation, also effect more economical nonlinear solution.

g. Improved accuracy control. Investigators have already suggested ways in which unbounded manipulation errors can be sensed and controlled. [70], [109], [112], [137], [40]. These ideas include kernel element generation, sensing of the stress evaluation error, discriminating singularity tests, and avoiding numerical singularity by optimizing arithmetic.

Program developers will also focus attention on program validation tests. These tests will address the problem of protecting the user from limitations of the simulation. For example, existing linear analysis computer programs are capable of predicting deflections of light-year magnitudes. Though the user can recognize this inconsistency between the analysis basis and the problem results, codes do not include the simple validation checks to insure infinitesimal deflections. As the engineer solves larger problems spanning more disciplines, he will stimulate creation of self-validating programs.

h. More intelligible output. Economically competitive hardware is available for plotting in color and toning pictures. [196]. Software concepts are established [189], [196]. Therefore, the next few years will bring colored graphical output of finite element simulations to production computer facilities and users.

TABLE IX summarizes these projected computer techniques for finite element analysis. The needs are all relevant to structural applications. The list suggests however, that most of the improvements in the next years will be driven by non-structural applications.
A. Problem Definition Techniques


8. Element Generation Techniques


C. Manipulation Techniques


D. Solution Procedures


E. Codes and Their Application


F. Graphical Output Capabilities


Joint Displacement Variables

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

envelope (upper boundary)

Joint Force Variables

8 9 10 11 12 13 14 15 16 17 18 19

bandwidth (of row 8)

maximum bandwidth

Figure 4. Banded Form of System Matrices

Figure 5. Triangulated Meshes