ADAPTATION OF A PROGRAM FOR NONLINEAR FINITE ELEMENT ANALYSIS TO THE CDC STAR 100 COMPUTER*

Allan B. Pifko
Grumman Aerospace Corporation

Patricia L. Ogilvie
Grumman Data Systems Corporation

SUMMARY

This paper discusses an effort to convert a nonlinear finite element program to the CDC STAR 100 pipeline computer. The program called DYCAST was developed under contract with NASA Langley Research Center for crash simulation of structures. Initial results with the STAR 100 computer indicate that significant gains in computation time are possible for operations on global arrays. However, for element level computations that do not lend themselves easily to long vector processing, the STAR 100 was slower than comparable scalar computers. On this basis it was concluded that in order for pipeline computers to impact the economic feasibility of large nonlinear analyses it is absolutely essential that algorithms be devised to improve the efficiency of element level computations.

INTRODUCTION

During the last decade, finite element methods, originally developed for linear structural analysis, have been extended to nonlinear problems. These developments have progressed to the point where the available methods are on a firm analytical basis and have been implemented in a number of general purpose computer codes. Since nonlinear analysis with the finite element method is essentially a successive linearization of a nonlinear problem, a given analysis completed in "n" steps can require a level of computation associated with "n" linear finite element solutions. Consequently, the problem facing the analyst today is not so much whether the solution to a given problem can be obtained, but whether the computer cost can be afforded. As analysts we can search for better algorithms based on the currently available computers in order to achieve near optimum computational efficiency while requesting that developers of computer systems constantly produce larger capacity faster machines. The purpose of this paper is to address this problem by describing an ongoing effort to convert a nonlinear finite element program to the CDC STAR 100 computer. This currently operational large fourth generation computer has a number of distinguishing features. From a user's vantage point, the two most important are the capability for vector arithmetic, e.g., pipeline processing, and the use of virtual memory. Vector arithmetic refers to the capability of performing computations on a string of numbers

* This work was supported by NASA Langley Research Center under Contract NAS-1-13148.

(vector) with a single vector instruction. More significantly, the computer central processing unit (CPU) is designed to exploit this type of calculation. For example, experience to date has indicated time savings as high as 35 to 1 over comparable CDC 6600 times for pure vector operations.

The virtual memory capability is a method of data management which gives the illusion that physical memory is larger than it really is. This is accomplished with both computer hardware and operating system software. In principle, the user can organize a program as if all the core necessary is available and the operating system "pages" data in and out of primary core. Thus, the process of overlaying code and using auxiliary storage devices to accommodate large quantities of data, common to programs implemented on third generation serial computers, can be assumed by the hardware and operating system software.

In order to exploit the features of the STAR 100, it is incumbent upon the user to design a solution strategy, if possible, that can utilize these features to the fullest. This can simply mean converting an existing algorithm to vector operations, or it can require devising an entirely new algorithm that can exploit pipeline processing.

An in-depth review of the features of the STAR 100, as well as its anticipated usefulness for finite element analysis, has been given by Noor and Fulton (ref. 1). A study of the feasibility of transferring NASTRAN® to STAR is found in ref. 2.

Work is currently underway by the authors to convert a program for nonlinear transient dynamic analysis to the STAR 100 computer. This program, called DYCAST (DYnamic Crash Analysis of STructures), was developed under contract with NASA Langley Research Center, for crash simulation of structures. It is the purpose of this paper to discuss our user experiences, successes and pitfalls, during the course of this effort. In the remainder of this paper we first outline the theory on which DYCAST is based and identify potential areas where the STAR vector processing will have a significant effect. In order to gain experience in the usage of the STAR pipeline processing system and the associated extended subset of FORTRAN vector syntax, a pilot effort was initially begun to vectorize a program for the eigenvalue/eigenvector extraction of large matrices (ALARM - Automatic LArge Reduction of Matrices to tridiagonal form - ref. 3).

In doing this we not only had the advantage of working with a small program but one that was particularly amenable to vector processing. Results from this study as well as those from DYCAST are presented in the paper.

DYCAST FORMULATION

DYCAST is a finite element program for the nonlinear transient dynamic analysis of structures with particular emphasis on crashworthiness evaluation. It is based on, and represents a continuation of the development of a system of finite element programs for nonlinear static analysis called PLANS (PLastic and Large Deflection ANalysis of Structures - ref. 4). A review of the salient features of DYCAST can be found in ref. 5.

DYCAST implements an updated Lagrangian formulation (refs. 6,7,8) for geometric nonlinearity and an incremental plasticity theory (ref. 9) to represent material nonlinear behavior.

In this section we outline the governing matrix equations used here in order to identify the key computation bound operations on which the STAR 100 will have its greatest impact.

The governing matrix equation based on the updated Lagrangian formulation with the displacement increment ΔU_{n+1} and acceleration $\ddot{U}_{n+1}$ as unknowns is

$$[K_n^{(0)} + K_n^{(1)}]\ \Delta U_{n+1} = P_{n+1} - F_n - M\ddot{U}_{n+1} \qquad (1)$$

This equation assumes that third order terms in the integral work relation have been neglected in going from the n^{th} configuration to the $n+1^{th}$. These neglected terms are discussed in refs. 6, 8. The matrix term $K_n^{(0)}$ is

$$K_n^{(0)} = \int_{V_n} W_n^t\ D_n^{-1}\ W_n\ d\ V$$

where V_n is the volume in the n^{th} configuration, W_n relates increments in total strain to increments in displacement, and $D_n = E^{-1}\ (I + EC)$ with E the matrix of elastic material properties and C a matrix obtained from the plastic constitutive relations. The matrix term $K_n^{(1)}$ is

$$K_n^{(1)} = \int_{V_n} \Omega_n^t\ \tau_n\ \Omega\ d\ V$$

where Ω_n relates the nonlinear components of the strain displacement relations to displacement increment and τ_n is a matrix of Cauchy stresses with respect to the n^{th} configuration. The matrix term F_n is

$$F_n = \int_{V_n} W_n^t\ \hat{\tau}_n\ d\ V$$

where $\hat{\tau}_n$ is a vector of Cauchy stresses with respect to the n^{th} configuration. One can use either an explicit or implicit time integrator to obtain solutions to eq. (1). Both types are implemented in DYCAST.

Explicit Integrator

DYCAST currently implements a constant time step central difference and a variable step modified Adams predictor-corrector time integrator. In both procedures, eq. (1) is written as

$$M\ddot{U}_{n+1} = P_{n+1} - F_n - \Delta f_{n+1} \qquad (2)$$

where

$$\Delta f_{n+1} = [K_n^{(0)} + K_n^{(1)}]\ \Delta U_{n+1}$$

is the incremental force vector.

This vector is then approximated in terms of previously calculated variables by making use of the discrete time integrator so that if the mass matrix M is unchanging the solution reduces to calculating a right hand side to eq. (2) and then solving for $\ddot{U}_{n+1}$.

For central difference use is made of the recurrence relations

$$\begin{aligned} \Delta U_{n+1} &= 2\Delta U_n - \Delta U_{n-1} + \Delta t^2 \ddot{U}_n \\ \Delta \dot{U}_n &= \frac{\Delta U_{n+1} - \Delta U_{n-1}}{2\Delta t} \end{aligned} \tag{3}$$

in order to obtain Δf_{n+1}.

The modified Adams procedure is based on substituting a predictor solution for ΔU_{n+1}, into eq. (2)

$$\Delta U_{n+1}^P = \Delta U_n^{dev} + \Delta t\ \dot{U}_n + \frac{\Delta t}{2}(\dot{U}_n - \dot{U}_{n-1}) \tag{4}$$

Equation (4) is the Taylor series expansion for U_{n+1} with the backward difference used for the acceleration and ΔU_n^{dev} is the difference between the n^{th} predictor and corrector solutions, $U_n^C - U_n^P$. Once eq. (2) is solved for $\ddot{U}_{n+1}$ the corrector solution is generated based on a forward difference for the third term.

$$\begin{aligned} U_{n+1}^C &= U_n + \Delta t\ \dot{U}_n + \frac{\Delta t}{2}(\dot{U}_{n+1} - \dot{U}_n) \\ \dot{U}_{n+1}^C &= \dot{U}_n + \Delta t\ \ddot{U}_n + \frac{\Delta t}{2}(\ddot{U}_{n+1} - \ddot{U}_n) \end{aligned} \tag{5}$$

An error criterion is used to ensure that the difference between the predictor and corrector solutions satisfies some preset error criterion. In practice, the convergence criterion fails on the difference between the predictor and corrector velocities. This is

$$\begin{aligned} \dot{U}_{n+1}^C - \dot{U}_{n+1}^P &= \frac{\Delta t}{2}((\ddot{U}_{n+1} - \ddot{U}_n) - (\ddot{U}_n - \ddot{U}_{n-1})) \\ &= \frac{\Delta t}{2}(\Delta\ddot{U}_{n+1} - \Delta\ddot{U}_n) \end{aligned} \tag{6}$$

and the error criterion becomes

$$\frac{\Delta t}{2}\left|\frac{(\Delta\ddot{U}_{n+1} - \Delta\ddot{U}_n)}{\dot{U}_{n+1}}\right| < \varepsilon \tag{7}$$

It can be seen from eq. (7) that the error criterion limits the rate of change of acceleration. Whenever the inequality of eq. (7) is violated the time step is halved. Conversely, the time step is doubled whenever the inequality is satisfied within a predetermined lower bound.

Implicit Integrator

A variable time step implicit solution algorithm with inner loop iterations, based on the Newmark-Beta family of integrators, is implemented within DYCAST.

The governing recurrence relations for the Newmark-Beta method are

$$\begin{aligned}\ddot{U}_{n+1} &= \frac{\Delta U_{n+1}}{\beta\Delta t^2} - \frac{\dot{U}_n}{\beta\Delta t} - \left(\frac{1}{2\beta} - 1\right)\ddot{U}_n \\ \Delta\dot{U}_{n+1} &= \Delta t\,\ddot{U}_n + \gamma\Delta t\,\Delta\ddot{U}_{n+1}\end{aligned} \tag{8}$$

Substituting eq. (8) into eq. (1) yields

$$\bar{K}_n \Delta U_{n+1} = P_{n+1} + Q_n^d - F_n \tag{9}$$

where

$$\bar{K}_n = K_n^{(0)} + K_n^{(1)} + \frac{M}{\beta\Delta t^2}$$

$$Q_n^d = M\left(\frac{\dot{U}_n}{\beta\Delta t} + \left(\frac{1}{2\beta} - 1\right)\ddot{U}_n\right)$$

Equation (9) can be solved iteratively within each time step by feeding back the effect of an equilibrium correction term.

In this form eq. (9) becomes

$$\bar{K}_n \Delta U_{n+1}^i = P_{n+1} + Q_n^d - F_n + \sum_{j=o}^{i} R_n^j \tag{10}$$

where

$$R_n^j = P_{n+1} - F_{n+1}^j - M\,\ddot{U}_{n+1}^j$$

When i = 0, eq. (10) reduces to eq. (9) since $R_n^0 = 0$.

There are a number of ways to define convergence. Use is made in DYCAST of the following criterion

$$\left\| \frac{\Delta U^{i}_{n+1} - \Delta U^{i-1}_{n}}{U^{i}_{n+1}} \right\| < \varepsilon \tag{11}$$

In addition to eq. (11), an admissibility test is used in order to control the time step. Based on our experience with the modified Adams method, the admissibility test is developed by writing a predictor form of eq. (8) that leads directly to a criterion similar to eq. (7)

$$\gamma \Delta t \left| \frac{\ddot{\Delta U}_{n+1} - \ddot{\Delta U}_{n}}{\dot{U}_{n+1}} \right| < \varepsilon \tag{12}$$

ALARM IMPLEMENTATION

As an initial test of STAR vector processing, a program for the eigenvalue/eigenvector evaluation for large matrices was converted to STAR. This program called ALARM (Automatic LArge Reduction of Matrices to tridiagonal form) is based upon a method which reduces a large matrix to an "equivalent" tridiagonal one of much smaller size. It is based on the work of Ojalvo and Newman (ref. 10) and as such is similar to the eigensolver, FEER, which has been implemented in recent versions of NASTRAN. For full details of the method the reader is referred to ref. 3 for ALARM and ref. 11 for FEER. In the following we outline the computational flow from ref. 3 as a means of identifying the areas of vector processing. The algorithm almost exclusively involves operations on large matrices and as such is particularly suitable for vector processing. It is based on generating a "tall" rectangular transformation matrix, V, the columns of which are orthonormal vectors. This transformation matrix reduces the equation

$$K \phi_i = \omega_i^2 M \phi_i \tag{13}$$

where K and M are the stiffness and mass matrices, respectively, to symmetric tridiagonal form of much smaller size. The eigenvalues/eigenvectors of this equation are then obtained by Sturm sequencing and bisection of the intervals containing ω_i^2.

The sequence of operations excluding the starting procedure and error checking is as follows (ref. 3):

1. Factor the stiffness matrix into upper and lower triangular form $K = LL^t$

2. Solve for successive columns of the reduction vector, V, as follows:

$$v^*_{i+1} = L^{-1} M L^{-t} v_i$$

This is accomplished by first solving for

$$L^t w_i = v_i$$

then performing the matrix multiplication

$$u_i = M w_i$$

and then solving

$$L v^*_{i+1} = u_i$$

All of these operations involve vector processing on vectors of the order of the semibandwidth and bandwidth of K and M. The procedure is completed by

3. $$\bar{v}_{i+1} = v^*_{i+1} - \alpha_i v_i - \beta_{i-1} v_{i-1}$$

where

$$\alpha_i = v_i v^*_{i+1}$$

$$\beta_{i-1} = v_{i-1} v^*_{i+1}$$

The desired column of V is then

$$v_{i+1} = \frac{\bar{v}_{i+1}}{\beta_i}$$

where α_i, β_i are the diagonal and off-diagonal terms, respectively, of the final reduced tridiagonal matrix.

The method outlined above was vectorized using the STAR 100 subset of vector FORTRAN syntax. Results are shown in Table I for CPU time for a number of benchmark problems using an IBM 370/168, CDC STAR 100 with all scalar operations and STAR 100 using vector processing. The times shown for benchmark problems exclude the factoring of the stiffness matrix, and assume a diagonal mass matrix. The table reveals that as the matrix size increases, the relative running time on the STAR 100 in the scalar mode increases relative to the IBM 370/168. However, as the problem size increases, the payoff for vector calculations grows. The final result, that of a 10000 degree of freedom matrix with a semibandwidth of 100 and a running time of 9.91 CPU seconds, indicates almost a 160-fold decrease in time over the IBM 370/168.

This table clearly indicates two points: the efficiency of vector processing for large global arrays and the relative inefficiency of the current version of the STAR 100 for primarily scalar arithmetic.

Figure 1 shows a section of a longitudinally compressed skin stringer element typical of the NASA space shuttle wing cover construction. A linear bifurcation buckling analysis of this structure was performed using one of the PLANS system programs. This program uses a higher order plate element to model the structure and the ALARM program to obtain the eigenvalues and eigenvectors. The resulting model led to matrices with 3158 degrees of freedom and a semi-bandwidth of 301. Running time on an IBM 370/168 in double precision depended on the amount of available core. Table II shows data from ref. 12 and indicates the normalized running time versus increased core size. The increased core size was due to increasing the work area available to the matrix packages. In no case, however, were we able to use more than 150000 words. This corresponds to the available incore capacity of STAR equal to approximately 450000 words for code and data and demonstrates a source of efficiency of STAR due to large incore storage coupled with the virtual core facility. This same problem was run with the STAR version of ALARM. To do this, the necessary matrices were assembled and passed to STAR via tape files.

The resulting computer times are summarized in Table III. Evaluation of two eigenvalues and eigenvectors to the desired accuracy took 12.5 CPU seconds on the STAR 100 computer, compared to 269.6 CPU seconds on the IBM 370/168 in double precision using 921 KBYTES of core. This represents a ratio of 21.5 to 1 which is consistent with the data presented in Table I. The computer time to calculate one column of the transformation matrix, step 2 outlined above, was 0.9 CPU second for the STAR 100 and 19 CPU seconds for the IBM 370/168. Also shown in Table III is the time to factor the stiffness matrix. To perform this operation, use was made of a vectorized equation solver written by Dr. J. Lambiotte, Jr. of NASA Langley Research Center. This procedure factors the matrix as $\bar{L}D\bar{L}^t$, where $\bar{L}$ is a lower triangular matrix and D is diagonal. In order to obtain the Cholesky factorization required by ALARM, we performed the additional operation of taking the square root of the diagonal matrix and premultiplied $\bar{L}$ by the result.

The predicted buckling load obtained from this analysis agreed quite favorably with the average of three tests (see fig. 1). Plots of the mode shape predicted by this model are also shown in fig. 1. For a clearer representation, only the portions of the structure that buckled are shown, and the deformed sheet and stringer flanges were plotted as if detached.

DYCAST COMPUTER IMPLEMENTATION

Given that the appropriate theories from structural mechanics have been implemented, the features that distinguish a program for nonlinear analysis from a linear program are: 1) the solution algorithm is of the repetitive type so that calculations performed once for a linear analysis must be repeated for a nonlinear analysis, and 2) field quantities such as displacements, stresses, and strains must be stored for use in succeeding calculations. These considerations force the designer of a nonlinear code to exercise extreme

care in coding key "number crunching" sections since any inefficiencies, while perhaps not being crucial for a linear analysis, are multiplied "n" times in a nonlinear analysis. It is in these areas that the STAR vector processing can potentially make a contribution.

In the following, the flow of the computational bound section of DYCAST is outlined in order to identify areas of potential vector processing. The program is separated into two functional units as shown in fig. 2. A small main program initially determines core allocations and then passes control to a subroutine that reads and processes all input and defines necessary data bases. Since computations are minimal in this routine the only impact of the STAR 100 here is the use of virtual core capability over utilization of temporary scratch files.

After the input phase, control is returned to the calling program which in turn calls a subprogram that controls the main computational loop. This loop, shown in block diagram form in fig. 3, has all the ingredients usually found in a linear finite element program, namely, (1) Matrix assembler, (2) Equation solver, (3) Time integrator, (4) Finite element matrix formation, and (5) Stress and strain calculations.

Items 1 through 3 are global functions that are easily recast into vector form. That is, the equation solver involves dot products on vectors whose length is of the order of the semibandwidth of the matrix, a process that can exploit the STAR vector processing. Implementing the integrator, although not taking a large percentage of the computation time, involves sums of vectors that can easily utilize STAR vector processing. However, items 4 and 5 are carried out on the element level and involve primarily operations with small matrices.

The method devised to implement element level calculations can be an important consideration. This is because it has been our experience that more than half the computer time per time step is taken by the element level calculations (for moderate sized problems) when using an implicit integrator. The relative time can be as much as 3/4 for an explicit integrator. Figure 4 shows the computational flow for a typical element level sequence of calculations for the simplest case of a three node constant strain triangle. For this element, the calculations involve primarily scalar arithmetic along with vector products on vectors of length 3 and 9. These lengths are too small for efficient vector processing because of instruction start-up time for vector operations. Vectors of length 100 begin to substantially exploit the STAR pipeline processing. It is also worth noting that since a finite element model consists of a multitude of elements, a computer system optimized for this type of operation might more naturally be based on parallel rather than pipe-line processing. That is, calculations can be carried out on many elements simultaneously. Reference 13, for example, conceptually describes the implementation of a finite element program on the Illiac IV parallel processing computer.

Practical experience was gained in implementing the items discussed above into a finite element program for nonlinear analysis by the Computer Sciences Corporation under Contract with NASA Langley Research Center. This program developed for plastic analysis alone, is part of PLANS (Plastic and Large deflection ANalysis of Structures - ref. 4). The program chosen treats the plastic analysis problem using the "initial strain" or pseudo load method. Consequently, the problem reduces to the solution of a sequence of linear analyses with a changing pseudo load vector that accounts for plasticity. The stiffness matrix is unchanged in each step so that it can be factored once with subsequent solutions requiring only a forward and backward substitution of triangular matrices. Within this framework the major effort in each step is to solve the set of finite element equations, calculate stresses and strains (impose plastic constitutive equations), and reformulate the vector of pseudo loads. Thus, it has all the steps that the explicit formulation of DYCAST has, eq. (2), with the exception that DYCAST must form a contribution to the pseudo load vector, Δf_{n+1}, that is the incremental internal force vector. Consequently, the effectiveness of the STAR computer on this program should carry over to DYCAST.

Results provided to us by R. Fudurich and D. Dunlop of Computer Sciences Corporation are shown in Table IV and summarize the results obtained to date. These results indicate a substantial speedup in the solution algorithm, up to 13:1 for initial solutions, and up to 190:1 for subsequent solutions that require only forward and back substitution. However, in the area of stress and strain recovery, where operations are presently primarily scalar, the STAR computer was appreciably slower than the CDC CYBER 175. As mentioned previously these element level calculations are performed on small matrices for which we anticipate that the vector capability would not lead to significant savings. Based on the results shown in Table IV it becomes apparent that it is essential to restructure the program in this area in order to exploit the STAR 100 pipeline capability.

Work is currently underway to convert the DYCAST program to the STAR 100 computer. To date, subroutines for matrix assembly and equation solution that use the STAR virtual core and vector processing capability have been implemented into DYCAST. The effect of vector processing was demonstrated on a finite element model containing 332 elements (179 triangular membrane, 33 axial force stringer, 120 beam), 413 degrees of freedom, and 136 semibandwidth. The computer times for one time step for this problem using both the implicit and explicit integrators on an IBM 370/168 and CDC STAR 100 are shown in Table V. As anticipated, any savings due to vectorizing the solution algorithm is offset due to the slower computation time on STAR for scalar operation. Vector processing, although on small vectors, is currently being introduced into the element level routines.

CONCLUSIONS AND RECOMMENDATIONS

This paper discusses our successes and pitfalls in converting a finite element program for nonlinear finite element analysis to the CDC STAR 100 computer. Our experience may be summarized by the following: On global functions, the vector processing, coupled with the virtual memory capability,

led to decisively improved computation times over the available scalar computers. However, on element level computations that were not vectorized, and indeed do not lend themselves easily to long vector processing, the STAR 100 was decisively slower than comparable scalar computers. We do expect some slight improvement when these routines are converted to vector code. Consequently, as the number of successive linearization increments increased in a nonlinear analysis, the gains made on processing global functions were offset by the element level calculations. On this basis it can be concluded that in order for pipeline computers to impact the economic feasibility of large nonlinear analyses it is absolutely essential that algorithms be devised to improve the computational efficiency of element level computations. Recommendations to accomplish this are discussed below.

In problems involving plasticity alone it is usual that a contained region of plastic flow exists. For this situation the pseudo load method can be used and element level stress recovery can be simply limited to this contained region. This can be done using the pseudo load method since these effective plastic loads are nonzero only in the contained region and element stiffness matrices are not reformed in each increment. This notion can be expanded even further (ref. 14) by employing a substructuring technique to eliminate the unknowns in the assumed elastic region. Since these calculations are exclusively on global arrays, the substructuring operations can effectively make use of vector processing. The remaining incremental calculations can then be carried out using the reduced set of equations.

Because of the effectiveness of vector processing on global arrays, any method that operates on these primarily large matrices during an incremental nonlinear solution will be computationally efficient on the STAR 100. Methods were previously developed (ref. 9) that eliminate the displacements in the governing matrix equation so that the remaining incremental equations are global arrays of stress or strain increments. For example, the total strain increment can be written as shown in ref. 9 as follows:

$$\Delta e^{T} = A\ \Delta P + J\ \Delta \varepsilon \tag{14}$$

where $\Delta\varepsilon$ is the incremental plastic strain, ΔP is an incremental load factor, and A and J are matrices that depend on the number of strain recovery points in the finite element model. Once eq. (14) is formulated, it is solved incrementally as part of the nonlinear analysis thereby replacing the usual element level calculations. Consequently, it may be worthwhile to re-examine this method for implementation on the STAR 100.

The previous comments are also pertinent to the formulation of linear dynamic analysis because the coefficient matrices do not change when either the explicit or implicit methods are employed. Consequently, the method reduces to calculating the right hand side vector and then solving for either accelerations or displacements. The right hand side vector for both methods can be recast in terms of global matrices by assembling and storing the total stiffness and mass matrices and then performing the matrix multiplications indicated in eqs. (2) and (9). Stress and strain recovery may still be a limiting factor, but these need not be computed in every time step.

The methods described above cannot be used in DYCAST because the stiffness matrix changes in each step due to the problem nonlinearities. One is therefore constrained to perform these element level calculations in every time step. Consequently these calculations must be reformulated so that they can effectively use vector processing. It may be possible to do this by partitioning the structure so that calculations are carried out on assemblages of elements, i.e., super elements. This technique will be pursued in future developments of DYCAST.

REFERENCES

1. Noor, A.K.; and Fulton, R.E.: Impact of the CDC STAR 100 Computer on Finite Element Systems, J. Structural Div. ASCE, vol. 101, 1975, pp. 731-750.

2. Study of the Modifications Needed for Efficient Operation of NASTRAN on the Control Data Corporation STAR 100 Computer. Aerospace Division of Control Data Corp., NASA CR-132644.

3. Ojalvo, I.U.: ALARM -- A Highly Efficient Eigenvalue Extraction Routine for Very Large Matrixes. The Shock and Vibration Digest, vol. 7, no. 12, Dec. 1975.

4. Pifko, A.; Levine, H.S.; and Armen, H. Jr.: PLANS - A Finite Element Program for Nonlinear Analysis of Structures, vol. I - Theoretical Manual. NASA CR-2568, August 1974.

5. Winter, R.; Pifko, A.; and Armen, H. Jr.: Crash Simulation of Skin-Frame Structures Using a Finite Element Code. Presented at Soc. of Automotive Engineers, Business Aircraft Meeting, Wichita, Kansas, Mar. 29 - Apr. 1, 1977, Paper no. 770484.

6. Armen, H.; Levine, H.; Pifko, A.; and Levy, A.: Nonlinear Analysis of Structures. NASA CR-2351, March 1974.

7. Hofmeister, L.; Greenbaum, G.; and Evensen, D.: Large Strain Elasto-Plastic Finite Element Analysis. AIAA J., vol. 9, no. 7, 1971, pp. 1248.

8. Bathe, K.; Ramm, E.; and Wilson, E.L.: Finite Element Formulations for Large Deformation Dynamic Analysis. Inter. J. for Numerical Methods in Engineering, vol. 9, 1975, pp. 353-386.

9. Armen, H. Jr.; Pifko, A.; and Levine, H.: Finite Element Analysis of Structures in the Plastic Range. NASA CR-1649, February 1971.

10. Ojalvo, I.U.; and Newman, M.: Vibration Modes of Large Structures by Automatic Matrix Reduction Method, AIAA J., vol. 8, no. 7, July 1970, pp. 1234-1239.

11. Newman, M.; and Pipano, M.: Fast Model Extraction in NASTRAN via the FEER Computer Program. NASA TMX-2893, Sept. 1973, pp. 485-506.

12. Crouzet-Pascal, J.: PLANS - Current and Potential Capabilities for Finite Element Analysis of Intersecting Thin Shell Structures. Grumman Research Dept. Memorandum RM-635, June 1977.

13. Field, E.I.; Johnson, S.E.; and Stralberg,: Software Development Utilizing Parallel Processing. Structural Mechanics Computer Programs, Surveys, Assessments, and Availability, Univ. Press of Virginia, Charlottesville, VA, 1974.

14. Armen, H., Jr.; and Levy, A.: Substructuring, Restart, and Variable Constraints in a Three-Dimensional Finite Element Program for Fracture Analysis. Grumman Aerospace Corp. Report RE-553, April 1978.

TABLE I. - COMPUTER TIME (CPU SECONDS EXCLUSIVE OF FACTORING STIFFNESS MATRIX) FOR EIGENVALUE/EIGENVECTOR EXTRACTION FOR VARIOUS SIZE MATRICES

Matrix Size	Maximum Semi-bandwidth	Solution Time IBM 370/168 (single precision)	Solution Time STAR 100 Scalar	Solution Time STAR 100 Vector	STAR Scalar / STAR Vector	IBM 370 / STAR Vector	IBM 370 / STAR Scalar
10	1	3.44	1.30	0.61	2.13	5.64	2.65
100	10	4.47	5.64	1.05	6.42	4.26	0.79
1000	100	26.91	76.21	1.76	45.06	15.29	0.353
4000	100	175.08		4.86		36.02	
4000	400	225.52		7.18		31.4	
10000	100	1532.3[+]		9.91		154.6	

[+]Time to form tridiagonal matrix

TABLE II. - PERCENT DECREASE IN CPU TIME ON IBM 370/168 VERSUS AVAILABLE CORE SIZE FOR BUCKLING ANALYSIS OF SKIN-STRINGER PROBLEM

Problem Size = 3158 degrees of freedom

Semibandwidth = 301

Core Utilized KBYTES	Normalized CPU Time
964	1.00
1356	0.667
1744	0.564

TABLE III. - COMPARISON OF CPU TIMES IN SECONDS FOR BUCKLING ANALYSIS OF SKIN-STRINGER PROBLEM

	Step 1 Factor K (CPU Time)	Step 2 Eigenvalue Eval. (CPU Time)	Total (CPU Time)
IBM 370/168	342.	269.6	611.6
CDC STAR 100	57.7	12.5	70.2
Ratio 370/STAR	5.9	21.5	8.7

TABLE IV. - SUMMARY OF CDC STAR 100 VERSUS CDC CYBER 175 COMPUTING TIMES FOR PLANS PROGRAM

Problem	No. of Elements	DOF	Semi-Bandwidth	Initial+ Solution	Initial* Total	Total Subsequent Solutions	Total S Recover Pseudo
1	25	138	42	0.2126/ 1.514	3.885/ 3.864	5.229/ 470.265	530.07 109.78
2	66	300	60	0.8142/ 5.390	8.8926/ 8.305	1.4579/ 277.679	191.90 34.71
3	560	1492	93	6.3/ 82.96	45.26/ 100.7	0.312/ N.A.	35.04 N.A.
4	710	4366	170	58.1/ 546.9	---	---	---

+Refers to initial factoring and forward and backward solution. All times given STAR time/CYBER-175 time.

*All initial preprocessing, element formation, critical load stresses and strains algorithm vectorized.

Problem 4 Compared versus IBM 370/168; Total times depend on number of increme range.

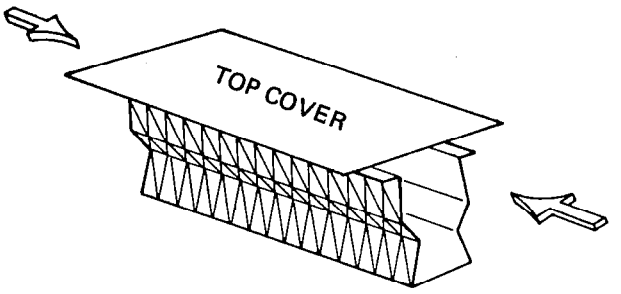
TABLE V. - COMPARISON OF CDC STAR 100 VERSUS
IBM 370/168 COMPUTING TIMES FOR
ONE TIME STEP USING DYCAST
413 DOF, 136 SEMIBANDWIDTH, 332 ELEMENTS

	Explicit	Implicit (one iteration)
Matrix Assembly	0.0/0.0	0.32/4.03
Solution	0.03/0.65	2.05/6.77
Time Integrator	0.02/0.02	0.04/0.10
Stress/Strain Recovery and Element Formation	8.86/4.08	16.90/6.87
Total	8.91/4.75	19.31/17.77

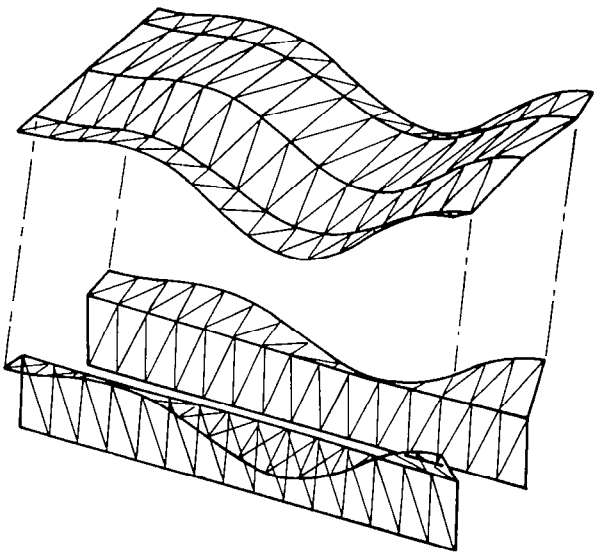
All times given in CPU seconds and STAR time/IBM 370/168.

BUCKLING LOAD, N (lb)
BENST 81500 (18320)
TESTS(3) 88850 (19970) AVERAGE

SKIN THICKNESS = 0.18 cm (0.070 in.)
LENGTH = 21.5 cm (8.5 in.)



a) FINITE ELEMENT MODEL OF STRINGER



b) COMPUTER PHOTO OF BUCKLING MODE SHAPE IN SKIN AND ATTACHED STRINGER FLANGES (ONLY PORTIONS OF STRINGER THAT BUCKLED)

Figure 1 Analysis of Longitudinally Compressed Skin-Stringer Element Typical of Shuttle Wing Cover Construction: Comparison of Computed Buckling Load with Test Results

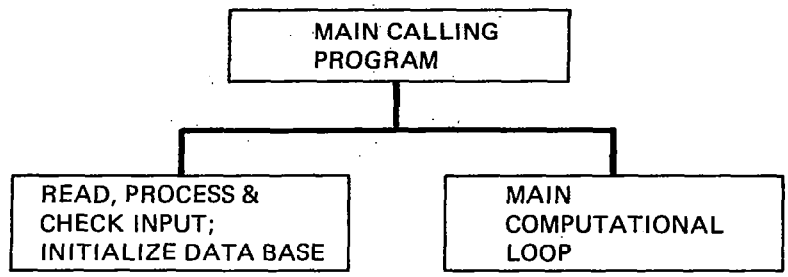


Figure 2 Global Structure of DYCAST

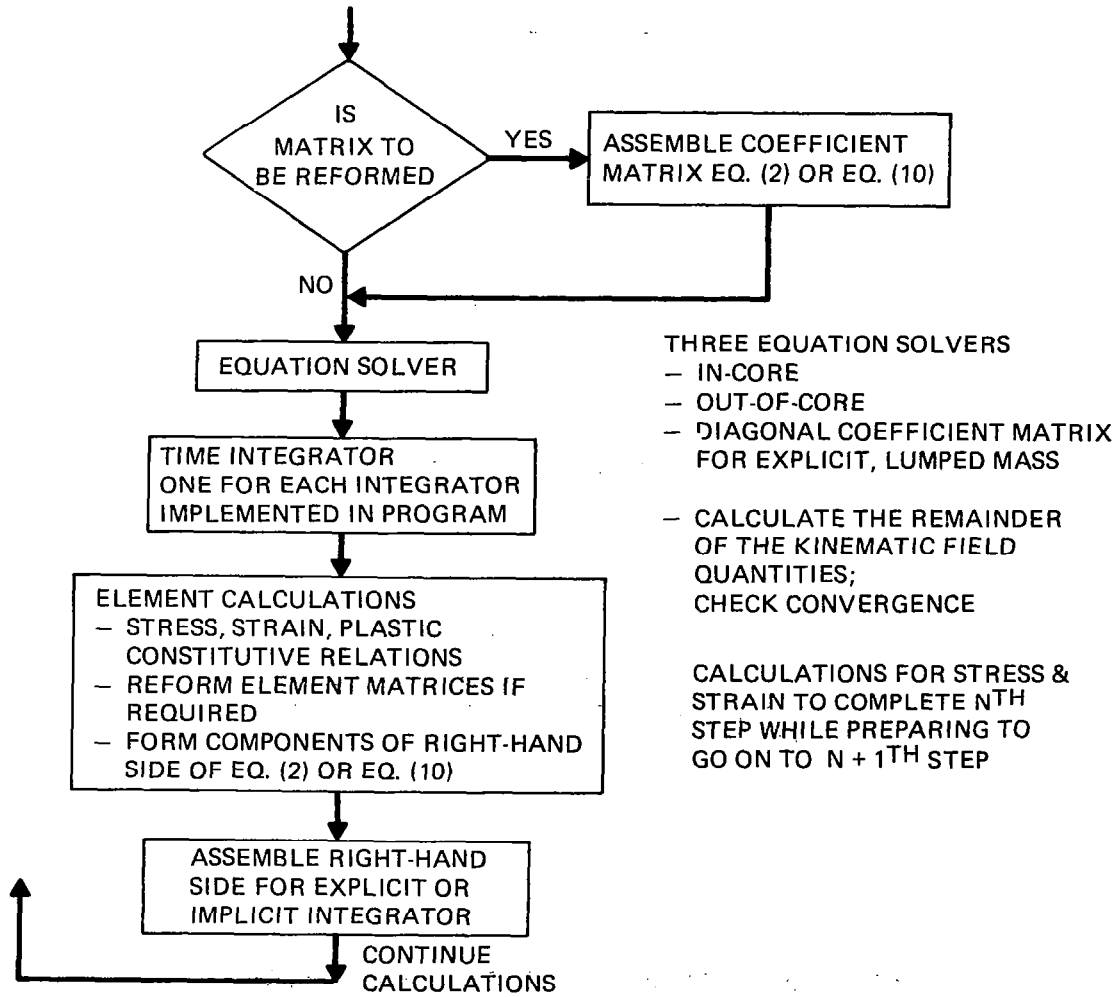


Figure 3 Computional Flow of Major Loop of DYCAST

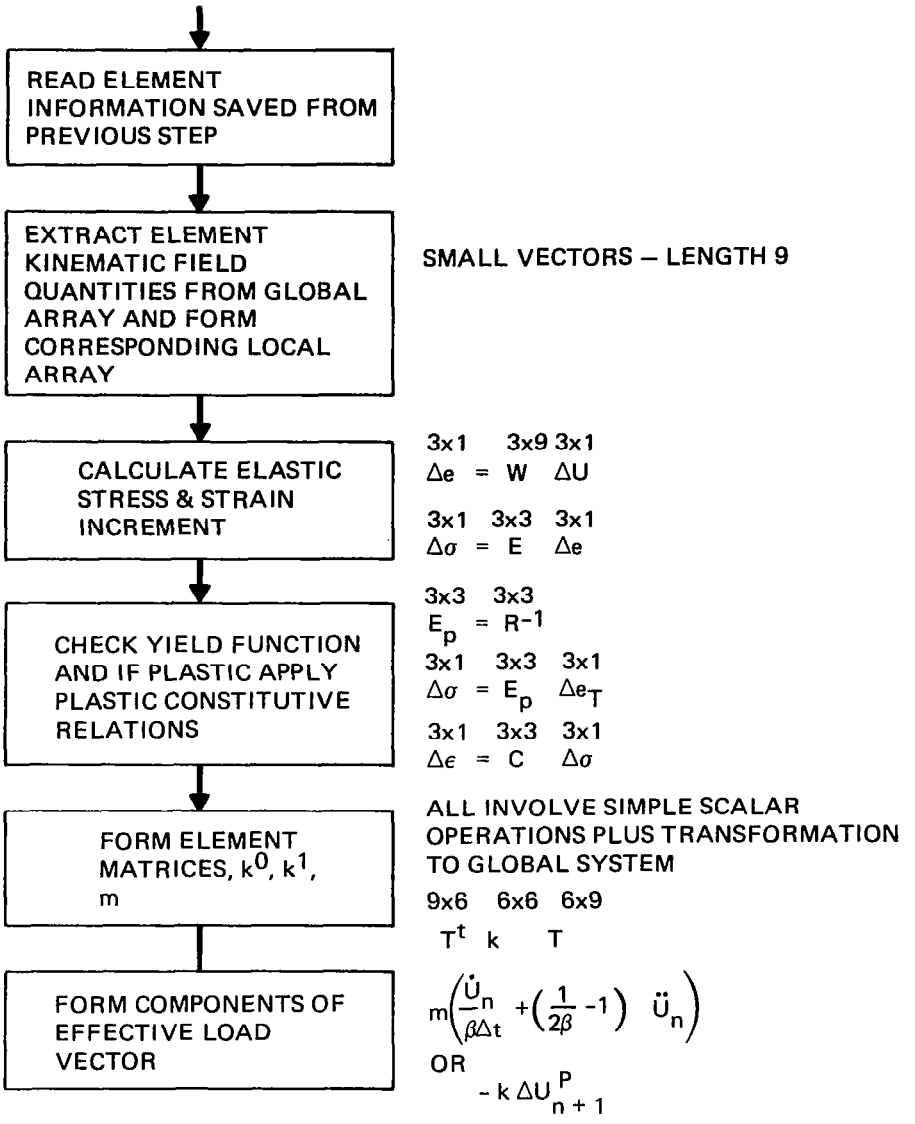


Figure 4 Computational Flow of Element Level Subroutine