NASA Contractor Report 4000

Enhancements to the STAGS Computer Code

C. C. Rankin, P. Stehlin,
and F. A. Brogan

CONTRACT NAS1-16723
NOVEMBER 1986

NASA

NASA Contractor Report 4000

Enhancements to the STAGS Computer Code

C. C. Rankin, P. Stehlin,
and F. A. Brogan
Lockheed Missiles & Space Company, Inc.
Palo Alto, California

Prepared for
Langley Research Center
under Contract NAS1-16723

National Aeronautics
and Space Administration

**Scientific and Technical
Information Branch**

1986

TABLE OF CONTENTS

INTRODUCTION

The research conducted during the last three years has greatly enhanced the power of the STAGS family of programs. Members of this family include STAGS-C1 [1] and RRSYS [2].

As a result of improvements implemented during these last three years, it is now possible to address the full collapse of a structural system, up to and beyond critical points where its resistance to the applied loads vanishes or suddenly changes. This also includes the important class of problems where a multiplicity of solutions exists at a given point (bifurcation), and where until now no solution could be obtained along any alternate (secondary) load path with any standard production finite-element code.

Formerly, when rotations for *any part* of a collapsing structure exceeded the small rotation regime ($< 10°$), finite elements in that region of the structure displayed unpredictable behavior that arose out of a violation of the basic assumptions concerning deformations within the individual elements. All elements in STAGS suffered to some degree, with the more economical elements (like the SH410 shell element [3]) locking almost completely as rotations grew. Now, the latest version of STAGS-C1 includes an element-independent corotational formulation for all static nonlinear collapse analyses that virtually eliminates all problems related to large-rotations for small strain problems.

Some of the remaining improvements to STAGS-C1 include a postprocessor that generates secondary solution data (strains, stresses, and resultants) from previously-saved primary data (displacements, velocities, eigenvectors, plastic strains). This postprocessor eliminates the need for saving bulky secondary data that may or may not later be needed for printout or display (e.g. STAPL). A new data translator formats primary solution data created on one type of machine (such as a CDC or CRAY) for simple transfer to another machine (like a VAX/VMS) for local, interactive, and inexpensive postprocessing or solution restart.

These improvements, including those in RRSYS and the Riks algorithm added to STAGS-C1 have transformed the STAGS codes into a unique capability for solving difficult nonlinear collapse problems typical of optimized structural components fabricated with lightweight materials.

This report concentrates on three key categories where most of the work during this contract period was performed.

The three categories are:

1. Development of an element-independent corotational procedure, its introduction into STAGS, and an evaluation of its performance.
2. Development of the Thurston Transformation Processor (TP), its introduction into STAGS-C1, and a demonstration of performance.
3. Specific enhancements to the RRSYS reduced basis program and completion of necessary documentation.

Each of these categories is discussed in this report in a section that summarizes the work. This report is supported by extensive and detailed documentation of theory, program construction, program usage, and test cases to be found in the following referenced sources:

1. Technical reports and manuals.
2. Technical papers published in the literature, of which four are derived from this contract, Refs. [4-7]

A final section summarizes recent miscellaneous improvements to the STAGS-C1 that, taken together, greatly enhance the utility of the program.

SECTION 1—An Element-Independent Large Rotation Algorithm

It is well known that most finite elements either lock (stiffen) or display spurious behavior in the presence of large rotations. It was principally for this reason that the SH411 shell element [3] was originally introduced: the much more economical SH410 element locks when rotations exceed approximately 10°. While the higher order SH411 increases accuracy, it also increases cost substantially. In addition, results may be unpredictable because solution convergence may not be monotonic. Convergence in the presence of large rotations is not guaranteed for some of the more economical shell elements irrespective of the grid size.

The problem arises from using rotations as fundamental freedoms to specify the displacement field of the elements. Rotations are used in shell elements to establish the curvature of the reference surface from which approximations to the bending strains are derived. Traditionally, these rotations are treated as vector quantities, which is true only when they are small. When rotations are large, the final deformed position of the element depends on the *order* in which the rotation components are imposed. That position cannot be uniquely determined with vector quantities. For finite rotations, membrane and bending contributions to the strains are no longer easily combined, and the resulting interpolation difficulties lead to locking or spurious behavior.

Fortunately, for most structures composed of thin sections, strains remain moderate even when the rotations are large. The major contribution to the total rotations arises from *local rigid body* motion that can be accounted for before strains are calculated within the finite elements.

The new large rotation algorithm addresses those cases in which strains remain small but rotations can be arbitrary. The success of the algorithm derives from the fact that the rigid-body contribution for *each* finite element is removed *before* any displacement information is required for the calculation of strain. The rotational freedoms which were formerly treated as vectors have been replaced by triads (three mutually perpendicular unit vectors) rigidly attached to each node, and a unique and efficient means of updating these triads with the increments of increasing displacement has been developed. Because the local rigid body motion is treated independently of the element formulation, *all* elements in STAGS were upgraded at one time; this is a pleasant contrast to the original task specifying that only *one* element (we had suggested the Ahmad SH440) be made corotational.

The element-independent large rotation formulation is detailed in reference [6]. Several test cases are included that demonstrate how well the method works for problems that have caused much difficulty in the past. For this reason the present report will concentrate on some of the specifics of the STAGS-C1 implementation, followed by some spectacular demonstrations of performance in highly nonlinear problems.

Section 1.1—STAGS Implementation and Data Organization

The subprograms that perform the operations described in [6] were written as independent modules that need only a minimum of data. The position of each of these modules follows closely the pattern described in Figs. 1, 2, and 3. Each subprogam requires the construction of a subroutine (or *shell*) that extracts the necessary data and ensures a proper interface between the kernel and existing STAGS software.

The only modifications to STAGS1 were to the input routines (to flag for the non-default non-corotational runs), and to the computation of the initial element reference frame $\mathbf{E}_0$ as shown in Fig. 4. All element frames follow the prescription in [6], with the selection of the local y axis as the 1–4 edge projection to facilitate the description of cones and cylinders (this guarantees that the element system follows the generators for cones and cylinders, simplifying the material description for these cases). For the eight node serendipity (Ahmad) element, the corner nodes define the element frame. It is fortuitous that for beams and triangles, the local element frame already in existence is identical to the choice in [6]. The local element frame $\mathbf{E}_0$ and the local element coordinates X_e for the undeformed system are saved with the element prevariational data for transfer to STAGS2. The initial nodal triads that form the reference for the rotational degrees of freedom are saved as separate nodal quantities and transferred likewise to STAGS2. They turn out to be identical to that transformation required to convert displacements at each node from the branch computational coordinate system to global coordinates.

Considering the benefits gained, the alterations to STAGS2 are relatively straightforward. The reduction to deformational translations $\mathbf{u}^{def}$ specified by the relation

$$\mathbf{u}_e^{def} = \mathbf{E}_k^T(\mathbf{u}_g + X_g) - X_e \tag{1.1}$$

requires the total displacements $\mathbf{u}_g$, the undeformed global coordinates X_g, the local element undeformed coordinates X_e, and the number of nodes in the element that have "full" (as opposed to "deviational") freedoms. The local element frame $\mathbf{E}_k$ is calculated in place and saved for subsequent steps. Local rotations are calculated from the relations (see explanation in [6])

$$\mathbf{D}_k^e = \mathbf{E}_k^T \Delta\mathbf{S}_k \mathbf{E}_0 \tag{1.2}$$

and

$$\mathbf{\Omega} = \frac{2[\mathbf{D}_k^e - (\mathbf{D}_k^e)^T]}{1 + trace\mathbf{D}_k^e} \tag{1.3}$$

Here $\mathbf{\Omega}$ is the skew-symmetric matrix of local rotations needed by the elements. Eqs. (1.2) and (1.3) require the additional presence of the initial nodal triad $\mathbf{S}_0$, the updated element frame $\mathbf{E}_k$ just calculated, and the quantity $\Delta\mathbf{S}_k$ that represents the rotation of the nodal triad as the structure deforms. In STAGS2 a *pseudovector* with the nonsingular $(2\sin\frac{\theta}{2})$ normalization was chosen to represent $\Delta\mathbf{S}_k$; a complete description of the pseudovectors used in STAGS is found in Ref. [6].

The remaining task was to settle on an optimum sequence of updating operations necessary to keep nodal triads and local element reference frames current. A particular sequence is shown in Fig. 1. Initially, the element frame was updated only after a converged solution was obtained. Subsequently we found, however, that a more consistent and accurate solution is obtained when both the nodal triads *and* the element frames are updated during each *iteration* (*inner* loop in Fig. 1), so that both UPSTR and EK are called immediately after new displacements are accumulated. This includes a call to UPSTR and EK (see Fig. 1) after the extrapolated displacements are calculated and *before* iterations begin.

Section 1.2—Verification

A demonstration of the performance of the element-independent large rotation algorithm for several interesting cases is found in Ref. [6]. These include the beam bent by an end moment, the classical elastica problem, the symmetric collapse of an arch, and collapse of a hinged cylindrical shell. In every case, solutions that previously were either incorrect or converged very slowly with respect to grid refinement became accurate to the order of the residual error for very coarse meshes when the new algorithm was turned on.

The SH411 element (the equivalent beam element is B211 [3]) was designed to cure the large rotation problem for cases where the directions of the rotations remain constant. In this case, the rotations *do* behave like vectors. The example shown in Fig. 5 demonstrates graphically that the element fails when rotations become large. The problem involves a deep arch clamped at one end and pinned at the other and loaded at its apex. The arch was modeled with 20 equally-spaced shell elements. One can see from Figs. 5 and 6 that both the SH410 and SH411 fail as soon as rotations become large. As is usually the case, the SH410 locked almost immediately, while the SH411 dropped *below* the converged solution calculated by A. Noor [8]. This example is interesting because even the *corotational* SH410 element locks eventually. One must note, however, that over a space of only seven elements the arch bends into a 3/4 circle. *Local* rotations inside those elements exceed 30°. For such large local rotations, the approximation of a rotation angle for a slope ($\tan\theta$) breaks down. A finer discretization or the use of the higher-order SH411 element provides an accurate solution along the entire load path.

Follower loads are already taken into account by referencing these loads to the *updated* element reference frame, small corrections being provided by the *local* element deflections. Our last example demonstrates the effectiveness of the method for follower loads. We have chosen an infinite cylinder subjected to inward hydrostatic pressure for which the collapse behaviour is known. Our model consists of one quarter of the ring shown in Fig. 7, with symmetry boundaries along all four edges. The mode $n = 2$ was triggered by a 1% point load at 0° and 90°. The response shown in the figure clearly illustrates the difference between the non-corotational and corotational SH410 and SH411 shell elements. Just as in the previous cases, the SH410 element is too stiff, and the underintegrated SH411 element too soft; in contrast, the corotational versions of both elements agree with the expected results. The deflected shape shown in Fig. 7 is plotted to scale, and represents the last point for the corotational SH410 element on the load-deflection plot.

Section 1.3—Imperfection Theory

Imperfections are introduced into STAGS by assuming that the strain energy is a function of the difference between the strain produced by the displacement field ${}^0\mathbf{u}$ of the imperfections alone and that produced by the combined displacement field ${}^0\mathbf{u} + \mathbf{u}$, where $\mathbf{u}$ is any new deflection of the structure as a result of loading:

$$\Delta\epsilon = \epsilon({}^0\mathbf{u} + \mathbf{u}) - \epsilon({}^0\mathbf{u}). \tag{1.5}$$

Green's expression for the strain components ϵ_{ij} is

$$\epsilon_{ij}(\mathbf{u}) = \frac{1}{2}(u_{i,j} + u_{j,i} + \sum_k u_{k,i}u_{k,j}), \tag{1.6}$$

where the standard comma notation is used for derivatives of displacements with respect to the coordinate directions for each of the indicated components (subscripts i and j). By straightforward algebraic manipulation the difference $\Delta\epsilon_{ij} = \epsilon_{ij}(\mathbf{u} + {}^0\mathbf{u}) - \epsilon_{ij}({}^0\mathbf{u})$ reduces to

$$\Delta\epsilon_{ij} = \epsilon_{ij}(\mathbf{u}) + \frac{1}{2}\sum_k({}^0u_{k,i}u_{k,j} + {}^0u_{k,j}u_{k,i}). \tag{1.7}$$

The only new contribution is from the second term.

As is described extensively in [6], strains in the corotational procedure are referred to the convected undeformed configuration, so that the local *deformational* displacements which participate in the strain energy are the difference between the total displacements and displacements arising from a *rigid* rotation of the element frame (see Ref. [6] for details). Thus the strain expression in Eq. 1.6 becomes

$$\epsilon_{ij}(\mathbf{u}) = \frac{1}{2}(u^{def}_{i,j} + u^{def}_{j,i} + \sum_k u^{def}_{k,i}u^{def}_{k,j}), \tag{1.8}$$

where $\mathbf{u}^{def}$ replaces $\mathbf{u}$. The quantity $\Delta\epsilon$ becomes

$$\Delta\epsilon_{ij} = \epsilon_{ij}(\mathbf{u}^{def}) + \frac{1}{2}\sum_k({}^0u^{def}_{k,i}u^{def}_{k,j} + {}^0u^{def}_{k,j}u^{def}_{k,i}). \tag{1.9}$$

The components of strain are now referred to the updated convected local coordinates in a local element frame which includes the imperfections. The expression for the local displacements becomes (Eq. 1.1)

$$\mathbf{u}^{def} = \mathbf{E}^T_k(\mathbf{u}_g + {}^0\mathbf{u}_g + X_g) - \mathbf{E}^T_I({}^0\mathbf{u}_g + X_g), \tag{1.10}$$

where the subscript g indicates that the displacements $\mathbf{u}$, the imperfection displacements ${}^0\mathbf{u}$, and the undeformed nodal coordinates X are all evaluated in the global coordinate system. Notice that the local nodal undeformed coordinates

$$X_I = \mathbf{E}^T_I({}^0\mathbf{u}_g + X_g)$$

include the imperfections in two ways:

(1) *Directly* by adding to the undeformed coordinates (in effect, deformed geometry).

(2) *Indirectly* because $\mathbf{E}^T_I$ is evaluated using the imperfections as displacements, in contrast to the quantity $\mathbf{E}_0$.

X_I are thus the total undeformed coordinates (initial + imperfection) in the local element frame.

The deformational rotations are evaluated using Eqs. 1.2 and 1.3 and converted into a vector of small rotations. The only modifications needed for structures containing imperfections is that all the quantities must be referenced to the *imperfect* state, just as with the translational displacements:

$$\mathbf{D}_k^e = \mathbf{E}_k^T \Delta \mathbf{S}_k \mathbf{E}_I, \tag{1.11}$$

where $\mathbf{E}_I$ and $\mathbf{E}_k$ are the initial and final element frames (both containing the imperfections) and $\Delta \mathbf{S}_k$ is the rotation of the surface coordinates required to take the system from the initial state I (containing the imperfections) to the final state k (containing imperfections *and* displacements). $\mathbf{D}_k^e$ is the matrix of small rotations from which the local rotations are extracted (see Ref [6] for further details).

What remains to be calculated are the ${}^0\mathbf{u}^{def}$ needed for Eq. 1.10. Eqs. 1.1, 1.2, and 1.3 are used as before, except that this time the initial configuration is the perfect geometry, and the displacements are the imperfections expressed as a displacement vector of nodal quantities. Eq. 1.10 becomes

$${}^0\mathbf{u}^{def} = \mathbf{E}_I^T({}^0\mathbf{u}_g + X_g) - \mathbf{E}_0^T X_g, \tag{1.12}$$

where this time the "deformed" element frame $\mathbf{E}_I$ contains the imperfections, and the "undeformed" element frame $\mathbf{E}_0$ does not. The rotations arising from the imperfections are obtained from the formula

$$\mathbf{D}_I^e = \mathbf{E}_I^T \Delta \mathbf{S}_I \mathbf{E}_0 \tag{1.13}$$

containing the imperfection rotations $\Delta \mathbf{S}_I$ (in global coordinates). Once these local deflections and rotations have been extracted, the initial strain quantities ${}^0\mathrm{u}_{k,j}^{def}$ are interpolated to the integration points, stored, and used subsequently in evaluating the strains during the solution process.

It can be verified that the terms in Eq. 1.9 are very small for all but the coarsest grid spacing because most of the imperfection has already been removed by the rigid-body translations and rotations that make up the initial element frame $\mathbf{E}_I$ and initial local element coordinates X_I. Numerous test cases indicate that the solution does not change significantly if, on the one hand, the imperfections are introduced as initial geometry, or, on the other hand, they are introduced into the strains using Eq. 1.7 (1.9 for corotational runs). For noncorotational applications, users are given the choice of either the initial strain approach (Eq. 1.7) or initial geometry. For corotational applications, only the initial geometry option is available; however, if the more refined implementation described in this section is desired, it can be included at some future time.

Section 1.4—Execution

All nonlinear collapse runs with STAGS-C1 (INDIC = 3) are corotational unless the user specifies that he wishes to run the cases the old way. A solution is non-corotational if ICOR (STAGS Input Record **B-1**) is set to unity.

If IWIMP (Input Record **M-5**) is positive, the trigonometric imperfection amplitudes specified on Record **M-6** are added to the initial geometry when the corotational (default) theory is used. In-plane components of the imperfection field are included in this superposition, a modification that applies to *all* classes of STAGS runs. For non-corotational runs, the imperfections are entered into the strains as before. For those users who formerly wrote a WIMP user subroutine (IWIMP = -1), a new subroutine DIMP has been added that *must* be called in place of WIMP for all corotational runs. The user must set IWIMP to -2 and include DIMP, as described in Section 4. The IWIMP=-2 option was provided to allow the user to alter the initial geometry for non-corotational runs as well.

Section 1.5—Suggestions for Future Research

The corotational theory developed under this contract is complete and self-contained as implemented in STAGS-C1. All subprograms are written in standard FORTRAN-77, and are general enough to be portable to other environments with no substantive changes.

One of the fundamental assumptions of the corotational theory is that *each* element is invariant to *infinitesimal* (as opposed to *finite*) rigid rotations. Although the computation of the internal forces remains valid, *infinitesimal* invariance is essential during calculation of the tangent stiffness for modified or full Newton iteration algorithms. For some flat elements, such invariance may be lost when an attempt is made to configure a doubly-curved surface. It turns out that it may be possible to remove the contribution of infinitesimal rigid motion to the tangent stiffness for some elements, thus relieving the old problem of *warping*. We believe that some of the recently observed irregularities (irregular convergence, non-quadratic convergence when using a full Newton iteration) seen with the SH410 corotational element stem directly from these finite element defects.

It has recently been demonstrated [9] that some newly-developed nine-node elements display superior behavior compared to that for a grid of four-node elements with the same number of freedoms. It may be necessary to explore better ways of choosing the local element frame so that the average surface can be more accurately represented than with the current choice of corner nodes. Fortunately, the software is general enough to allow for an arbitrary selection for nodes (such as the midside nodes) to define $\mathbf{E}_0$ and $\mathbf{E}_k$. The best possible frame must be selected: For a given number of freedoms there are only *one fourth* the number of local element frames in the nine-node element model than in the four-node element model with an equivalent grid.

The reader may have noticed that no mention has been made of transient analysis (INDIC=6) in STAGS. Although the corotational theory has *not yet* been implemented for this class of runs, this could be done very simply with the existing corotational software.

SECTION 2—Thurston Transformation Processor (TP)

General purpose finite element codes in use today for nonlinear structural analysis are slow to converge or fail to converge near limit points or bifurcation points. The Thurston Transformation Processor (TP) is an adaptation of Newton's method that improves convergence near critical points where the linear form of Newton's method is ineffective. By critical points, we mean both limit points and bifurcation points.

TP fills an important gap in the nonlinear solution process. Whereas the recently-implemented Riks solution algorithm is of some help in traversing simple limit points, TP enables rapid convergence in the vicinity of both limit and bifurcation points, including those analyses involving multiple bifurcation. When fully developed, TP will give the user the freedom to continue on any alternate solution path, taking full account of the interaction of multiple bifurcation modes in critical neighborhoods of load-deflection space.

The theory underlying TP has recently been presented at the AIAA/ASME/ASCE/AHS 26th Structures, Structural Dynamics, and Material Conference (SDM) [7]. Further detail, including some of the particulars of the STAGS-C1 implementation can be found in Appendix 1.

Section 2.1—Highlights of the Theory

Implementation of the Thurston Transformation Method (TP) is described in Ref. [7]. TP's purpose is to separate the incremental equilibrium equations into two parts. The first part is almost singular, reflecting behaviour of a structure close to a bifurcation or limit point. The second part is non-singular. This separation is done by a sequence of carefully selected operations. First, we apply an equivalence transformation to the incremental equilibrium equations in order to replace some of the original displacement freedoms by the amplitudes of potential bifurcation modes. Subsequently, we subdivide the resulting equations into two smaller equation systems. The first of these is decoupled from the rest of the equations and contains as unknowns only the modal amplitudes and the load factors. It is nearly singular, highly nonlinear and must be solved by special means. It is this set of equations that describes the interaction between different modes in the vicinity of a bifurcation point. The second set of equations describes the coupling between the modal amplitudes, the load factor and the remaining displacement freedoms. This set is numerically well-conditioned and can be treated by standard equation solution techniques, e.g. by true or modified Newton iteration procedures. The Thurston Transformation Method thus gives a comprehensive and consistent description of structural behaviour close to critical points. Not only does it avoid the convergence problems encountered by other methods, but it is the first method permitting an evaluation of modal interaction for general structures in an explicit, efficient and comprehensible manner.

Section 2.2—STAGS Implementation

TP was implemented in STAGS-C1 in stages, as follows:

1. Modifications to the solver to handle multiple right hand sides.
2. Modifications to the eigenvalue package to handle restart (in fact, to be called anywhere desired).
3. Extension of the capabilities of the Z-system routines (described in Appendix 1). This includes a full set of vector operations and control of restart (Section A1.2.3).
4. Construction of the control module BPPCON to process user input data.
5. Introduction of the equivalence transformation and construction of the reduced equation system, taking into account the higher-order terms in the modal amplitudes. The implementation of the equivalence transformation was brought about by the *single* control routine BPP. BPP requires only standard STAGS2 modules and the augmented Z-system routines.
6. Modifications to the output routines for display of the outcome of key TP functions during the progress of a run.

The vector and matrix nomenclature is also found in Appendix 1, followed by principal variable names. A *complete* list of Z-system calls is also included.

Section 2.3—Verification

Two examples found in [7] demonstrate the function of TP in STAGS-C1. The first case in [7] is a shallow spherical cap under hydrostatic pressure, an example of a difficult limit point analysis. Not shown in the figure is an equivalent STAGS-C1 analysis with the Riks continuation parameter algorithm. Although identical results were obtained, the Riks solution required almost 30 times as many load steps as TP, with a tight clustering of points near the limit point (see Fig. 1 in [7]). The second example, fully documented in [7], demonstrates how TP works in a case in which a continuation method such as Riks' cannot work.

The only example we have of a multi-mode solution is shown in Fig. 8 and 9. The solution was obtained with an experimental version of TP not yet completed in STAGS-C1, with the help of a grid search algorithm used to determine modal amplitudes from the reduced equation system containing the higher-order terms. The example consists of a cylindrical panel under axial compression with geometry and boundary conditions shown shown in Figure 8. The finite element model of this structure had 148 computational freedoms.

A bifurcation analysis with linear prestress showed that the first five bifurcation load factors are 0.697, 0.882, 0.915, 0.937 and 0.961, i.e. the lowest eigenvalue is relatively isolated, whereas the others are clustered. An attempt was made to reach a secondary load branch starting from the neighborhood of the second eigenvalue and proceeding in the direction $\mathbf{e}_2$ of the corresponding eigenmode. Two bifurcation modes were used in the equivalence transformation, namely the modes associated with the second and third eigenvalues. The

initial step employed to reach the first shifted point was chosen to be $0.2\mathbf{e}_2$ (see Figure 8, point P_1).

At this point, the Thurston method was exercised without, however, obtaining convergence after a restart in the basic equation system. As a consequence, another specialized step using TP (hereafter referred to as a *Thurston step*) was taken in the direction of $\mathbf{e}_2$ (P_2). The solutions for the modal amplitude increments obtained from the reduced equation system at this point are shown in Figure 9. The solid lines display the loci of all solutions in the plane spanned by the amplitude increments with the load factor P_A being a parameter along the curves. It is interesting to note that there are two classes of solutions represented by the two disjoint solution branches. Hypothetically, the two classes are associated with the pair of secondary load branches in the direction of the bifurcation modes used in the equivalence transformation. Using the solution at point A in Figure 9 convergence was obtained after a restart in the basic equation system, although the converged solution seemed to contain not only one eigenmode, but rather a mixture of the two. Evidently, since the reduced equation system has many solutions one has the choice of selecting other points from which to restart the solution of the basic equation system. Also, one might restart from a point on the second branch of the reduced solutions with possible convergence to another secondary load branch of the structure.

The broken lines in Figure 9 show the loci of the solutions on the reduced solution space after yet another Thurston step (P_3). It is noteworthy that in this case, starting from solution point B, no convergence was obtained in the basic equation system.

It is clear from this example how complicated it is to analyze the modal interaction problem, and how far one was able to proceed with only an experimental version of TP.

Section 2.4—STAGS TP Execution

A series of new input records are needed for the control of TP. Each of these records is digested during the solution phase (STAGS2). Therefore, the total user input to the program comes in two parts:

1. Input to STAGS1. This input block defines the model. It is completely unchanged from an ordinary model definition with the nonlinear static option turned on (INDIC = 3).
2. Input to TP. TP replaces STAGS2 and contains all necessary functions for a static, nonlinear analysis (including corotation and the Riks algorithm).

The new input records **TP-1**, **TP-2**, and **TP-3** contain the same data as the STAGS1 records **A-1**, **C-1**, and **D-1**, respectively. Data in the new records override the STAGS1 input records. This has the advantage that multiple solutions with differing load strategies can be run with the same STAGS1 data. The next three records control the operation of TP as follows:

TP-4 This record is used to select either TP or the Riks algorithm, and when selecting TP, to specify the type of modal analysis desired and which mode to use for the shifted

point. If NPATH is 0, TP will attempt to continue the solution along the current load path, just as would be done in an ordinary STAGS solution. If NPATH is 1, a Thurston step will be attempted, and additional data from this record will be used. NEV specifies the number of bifurcation solutions to calculate. NEQ specifies the number of modes to be used in the Thurston Equivalence Transformation. NSOL describes the particular action of TP, accounting for multiple modes. For NSOL = 1, it is assumed that only one mode is active, and a Thurston step is attempted using this mode for the shifted point. For NSOL = 0, the modal coefficient matrix is saved and the TP run is terminated. For NSOL = -1, the modal *solutions* are read in (Record **TP-6**), and a full multimode Thurston solution step is attempted. IE denotes the mode number to be selected in computing the shifted point.

TP-5 Read **TP-5** and **TP-6** only if NPATH = 2. This record specifies the value of Thurston step for a multimode TP solution. STEP is the independent variable of the solution, replacing load increments (in the case of load control), or continuation parameter increments (for a Riks solution). STEP multiplies the normalized mode IE (from the previous record). PASTP (Record **TP-2**) serves the same function for a TP analysis using a single mode or for the standard load control or Riks analyses. DELTA is some small number to be used in a *numerical* differentiation algorithm for determining the coefficients of the reduced equation system.

TP-6 The data on this record specify the relative values of the modal amplitudes for a multimode TP analysis (NEQ > 1). Load factors PA, PB, and the QFAK are determined from the reduced nonlinear equations specified by the modal amplitude coefficient matrix saved on TAPE31.

The reduced space equations are solved by a separate grid search program using the coefficient data on TAPE31. Currently, however, a multimode TP execution has only been attempted using the experimental version; this option awaits debug and testing in STAGS-C1.

Section 2.5—Current Status and Suggestions for Further Work

At the time of this Report, TP is fully operational in STAGS-C1 for all cases where the specification of one bifurcation or collapse mode suffices to remove the singularity in the tangent stiffness in the vicinity of the critical point. For a multimode case, only the test version has been tried.

A new effort has already begun to fill in the gaps in TP following the theory in Ref. [7]. The nonlinear equations in the modal amplitudes will be augmented with other terms. The links between STAGS-C1, TP, and the grid search algorithm for solving the nonlinear reduced equation system will be checked out and improved. Improved methods for solving the reduced equation system will be investigated. Additional improvements will be made as experience with difficult nonlinear collapse and bifurcation analyses dictates.

SECTION 3—RRSYS

The development of RRSYS that began six years ago was spawned by the need for faster and more powerful analysis tools. Despite significant progress in both software and hardware technology, the solution to many problems remains out of reach for the analyst, chiefly due to the significant cost of performing large structural analyses. Since computational effort is strongly influenced by the number of unknowns in the equation system describing the deformation or the motion of a structure, it seemed attractive to explore methods for reducing the number of freedoms characterizing the state of a structure. Another factor critically influencing computing time is the strategy employed for arriving at a solution, particularly in the case of structures with pronounced nonlinear behaviour. The analysis of nonlinear structural phenomena cannot be handled by cookbook-type recipe procedures. In order to arrive at an accurate answer with reasonable computing costs, the analyst needs years of experience that provide intuition and a thorough understanding of the physical processes involved. Hence, there is a strong need for automated problem-adaptive methods that select solution strategies adequate to deal with complex structural behaviour encountered in the course of an analysis.

RRSYS contributes to the solution of the problems of excessive computer time and unreliable strategy. On the one hand, it implements the reduced basis technique by the use of global functions, leading to a significant reduction in the number of unknowns and hence the associated solution times. On the other hand, both the conditions under which the reduced solution space is constructed and the strategy which is used for solving the ensuing equations are automatically adapted to the behaviour of a particular structure. The default solution strategy built into RRSYS usually results in cost savings by a factor of three to five for a typical collapse analysis. RRSYS suggests to the analyst how to choose important parameters affecting the computational effort, thus removing much of initial guesswork involved in finding a solution strategy suitable for a particular case. Options exist for changing the strategy parameters, permitting the analyst further to improve solution efficiency once he has gained insight into the behaviour of a structure.

RRSYS can be used for the analysis of shell structures of arbitrary design and configuration.

Section 3.1—Recent Improvements to RRSYS

During the previous contract period the essential program modules implementing the reduced basis technique and adaptive solution strategy were written and joined together to form the RRSYS program assemblage. Considerable effort was devoted to making the system reliable and error tolerant. Much thought was also given to devising a useful default strategy of the adaptive solution mechanism applicable to a wide range of problems.

The work done on RRSYS during the present contract period consisted partly of improvements made to the system and partly of writing a comprehensive manual [2] describing the function of the system. The manual describes the components of RRSYS. The design of the underlying data structure and methods of access to it are also described.

Improvements to RRSYS include streamlining the code to raise its efficiency. A modular system like RRSYS always carries a certain overhead for data transfer and communication

between modules. Anything that can be done to trim these administrative processes raises computational efficiency. The *restart procedure* was modified and *redundant functions* such as UNMERGE have been eliminated. The module RRLIM for performing collapse analyses in the reduced solution space has been completely rewritten in order to improve the efficiency and accuracy of limit point searches.

Results obtained with RRSYS applied to the collapse of shallow cylindrical shells under combined loading were presented at the PVP conference 1984 in San Antonio, Texas [5].

To aid analysts in using RRSYS, a Systems Reference Manual [2] was written. The manual explains in detail the function and use of each processor. It also describes the input data required by the modules and gives instructions for their use, both as independent processors and as part of the whole RRSYS complex. The output produced by various modules is explained via examples. A comprehensive description is given of the data structure used in RRSYS. This description will facilitate the addition of new modules by programmers other than the original developers.

Section 3.2—Current Status

In summary, RRSYS is fully operational for static collapse analyses. It is applicable to cases with complex structural behaviour because of its automated problem adaptive solution strategy. The use of RRSYS for the investigation of transient response, on the other hand, still requires user interaction. A theory of how to make the solution strategy problem-adaptive does not exist for dynamic analyses of general structural configurations.

SECTION 4—SUMMARY OF IMPROVEMENTS TO THE STAGS FAMILY

Numerous improvements to the STAGS family of codes have been implemented during the period of performance of this Contract, the most important of which are covered in the previous sections. Included here is a summary of additional improvements and a guide to the "1985 Changes to the 1983 STAGS-C1 Manual" (hereafter called "1985 Changes") recently released to COSMIC with the 1985 STAGS-C1 program. The improvements covered here reflect the total STAGS effort, a part of which was supported outside of this contract†

The "1985 Changes" is included here as Appendix 2. This guide comes in three parts as follows:

1. A summary of STAGS improvements.
2. Corrections to entries in specified records of the STAGS1 input block, including user subroutines.
3. Corrections and additions to the STAGS-C1 manual text. These changes reflect the increased capabilities recently introduced.

Section 4.1—POSTP and UNFFMT-FMTUNF

A postprocessor POSTP has been written for the recovery of secondary solutions (i.e., stresses, strains, resultants) with use of the solution vectors that are saved in the SOlution Data file (SOD) produced by a STAGS-C1 execution (STAGS2). The description and use of this postprocessor constitutes section 5.0 in the STAGS Manual [1]. POSTP uses simple input to direct printing of either primary or secondary solution data at specified load or time steps. POSTP also produces new SOD files containing the recovered secondary data for STAPL plot postprocessing. It is therefore never necessary to save bulky secondary data during the initial analysis with STAGS-C1; all such data is efficiently re-created with POSTP.

The pair of data translators UNFFMT-FMTUNF serve to convert SOD primary data from the machine (binary) representation into ASCII form for transport to other machines. This increases flexibility when STAGS, POSTP, and STAPL are resident on more than one type of machine, because the primary analysis can be done on one type of machine, with postprocessing or restarts to be performed on another computer. A detailed description of these translators is found in the Appendix 2.

Section 4.2—Imperfections as Perturbed Geometry

The option of introducing imperfections as modifications to the initial geometry was provided when STAGS became corotational (Section 1.3). At the same time, imperfection

† Some of this work, including the implementation of the Riks solution algorithm, was supported by the David Taylor Naval Ship Research and Development Center (DTNSRDC).

components in the in-plane directions were introduced. The modifications to the input are found in Appendix 2 (page 44).

Section 4.3—Displacement Load Histories

The user has the option of writing a program for extracting displacement and load histories specified as *input* to STAGS2. More on this can be found in Appendix 2 (page 40). Construction of a sample data extraction program using the history data is found in Appendix 3. A connection has been established to a NICE/GAL [10-11] compatible data library and used to create many of the history plots found in this Report and related documents.

REFERENCES

1. B. O. Almroth, F. A. Brogan, and G. M. Stanley, "Structural Analysis of General Shells, Vol. II, User's Instructions for STAGSC-1," Lockheed Report *LMSC-D633873*, Dec. 1982.

2. P. Stehlin, F. A. Brogan, and B. O. Almroth, "RRSYS—System Reference Manual, Second Revision," NASA Contractor Report *CR-178096*, July 1986.

3. B. O. Almroth and F. A. Brogan, "Numerical Procedures for Analysis of Structural Shells," AFWAL-TR-80-3129 (Flight Dynamics Lab., Wright- Patterson AFB, 1981).

4. K. C. Park, "An improved Semi-Implicit Method for Structural Dynamics Analysis," *ASME J. Appl. Mech* **49**, No. 3, pp. 589-594 (Sept. 1982).

5. P. Stehlin and F. A. Brogan, "Analysis of Structural Collapse by the Reduced Basis Technique," pp. 69-83. In *Collapse Analysis of Structures* (eds. L. H. Sobel and K. Thomas), ASME, New York (1984).

6. C. C. Rankin and F. A. Brogan, "An Element-Independent Corotational Procedure for the Treatment of Large Rotations," *ASME J. Pressure Vessel Technology* **108** pp. 165-174 (May, 1986)

7. G. A. Thurston, F. A. Brogan, and P. Stehlin, "Postbuckling Analysis Using a General Purpose Code," *AIAA Journal* **24**, No. 6, pp. 1013-1020 (June, 1986).

8. A. K. Noor and J. M. Peters, "Reduced Basis Technique for Nonlinear Analysis of Structures," *AIAA Journal* **18**, 455-462 (April 1980).

9. G. M. Stanley, "Continuum-Based Shell Elements," Lockheed Report *LMSC-F035839*, August 1985, to appear in the book "Finite Element Methods for Plate and Shell Structures, Vol. 2: Formulations and Algorithms," eds. T. J. R. Huges and E. Hinton (Pineridge Press Int. Lmtd., Swansea, U.K., 1986).

10. C. A. Felippa and G. M. Stanley, "NICE: A Utility Architecture for Computational Mechanics," presented at the 3rd USA-Europe Symposium on Finite Element Methods for Nonlinear Problems, Trondheim, Norway, August 1985, Proceedings to be published by Springer-Verlag.

11. C. A. Felippa, "Database Management in Scientific Computing - II. Data Structures and Program Architecture," *Computers & Structures*, **12**, No. 1, 1980, pp. 131–146

12. W. C. Perry and C. C. Rankin, "110 Mount Element," Lockheed Report LMSC-D062175, revised 1985.

13. John A. DeRuntz, "Implementations of Creep and Hardening Plasticity Theories in the STAGS-C1 Structural Analysis Code," Lockheed Report LMSC-D811752, 1981.

14. T. L. Geers, J. A. Deruntz, G. M. Stanley, W. C. Perry and C. C. Rankin, "Enhanced Analysis Capability in USA-STAGS," Lockheed Report LMSC-D877665, 1983.

FIGURES

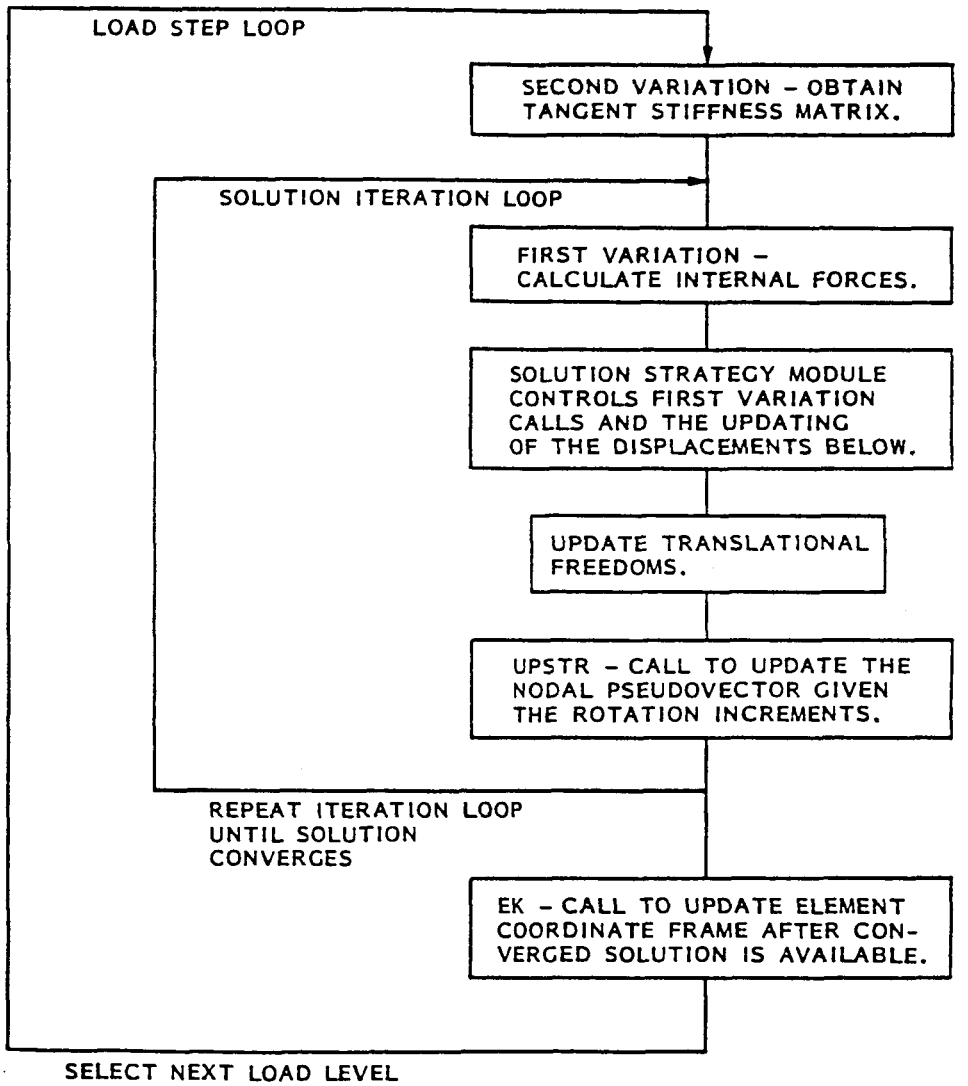


Figure 1. Summary of calculations carried out during the solution phase.

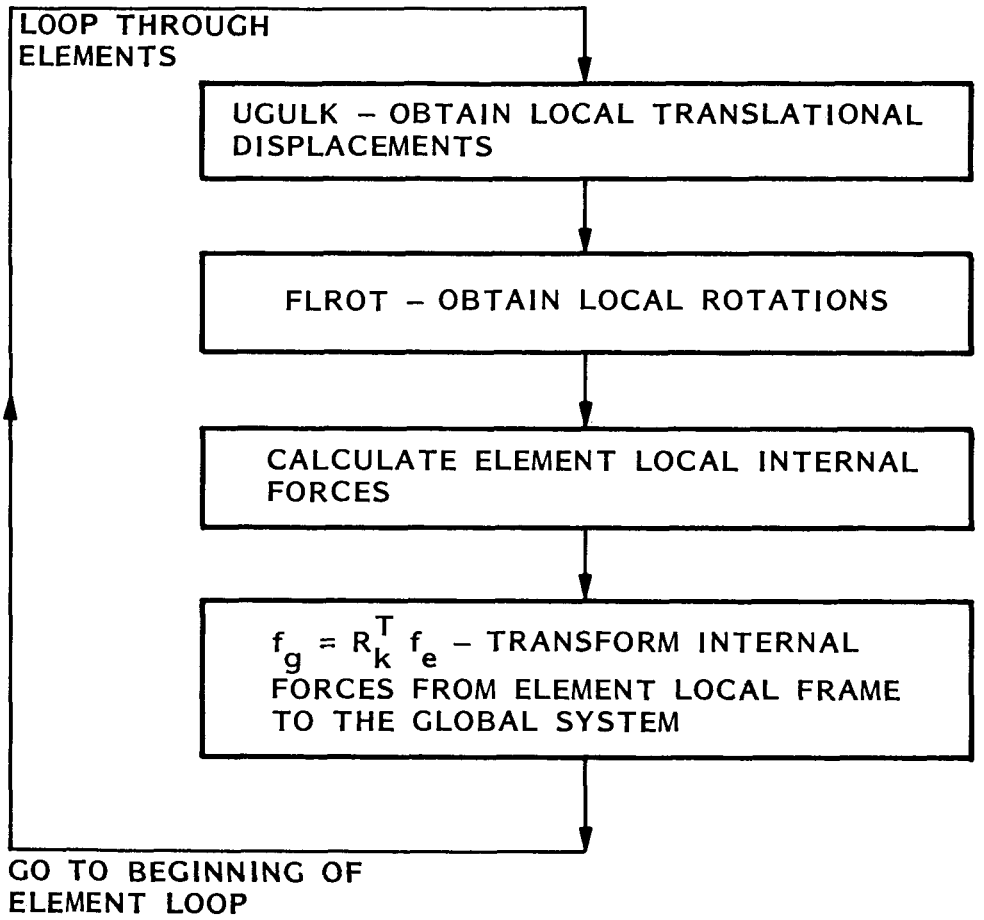


Figure 2. Modifications to the calculation of internal forces.

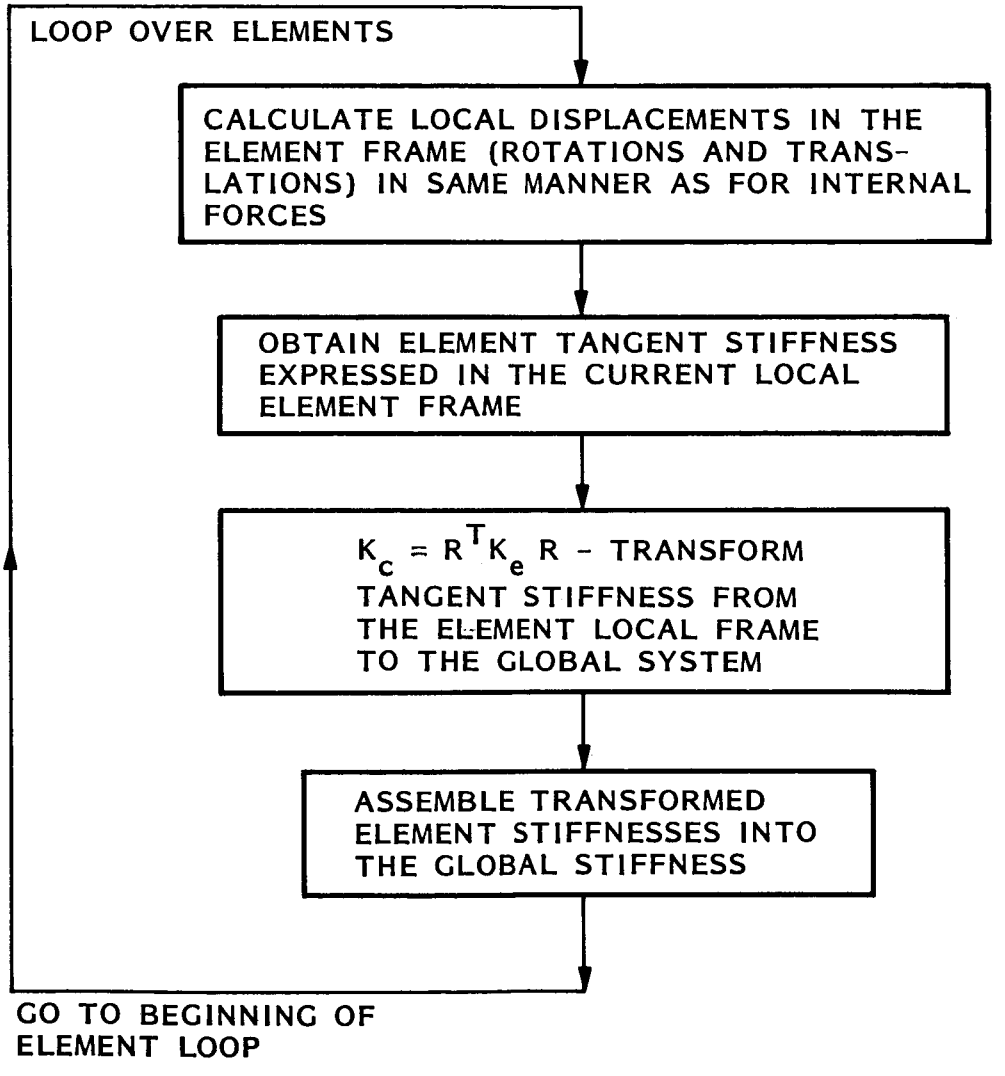
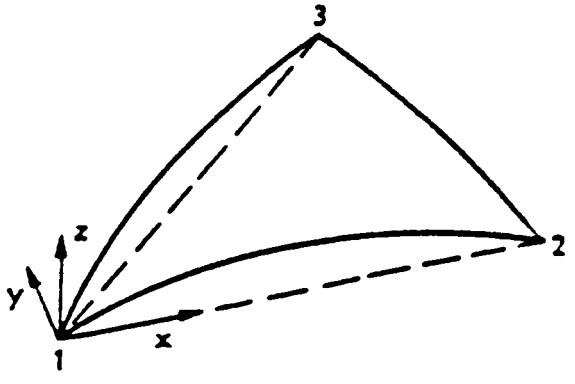
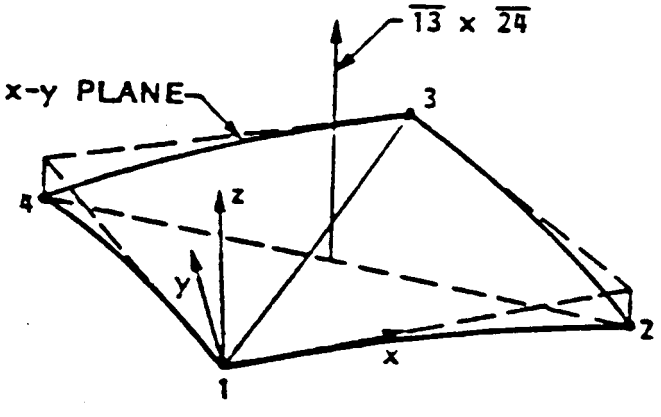


Figure 3. Modifications to the calculation of the tangent stiffness.

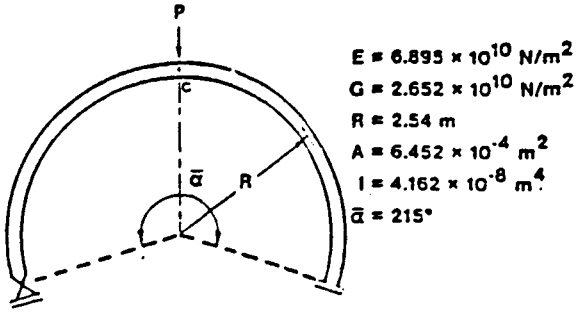


TRIANGULAR ELEMENT



QUADRILATERAL ELEMENT

Figure 4. Definition of the local element frame $\mathbf{E}_k$.



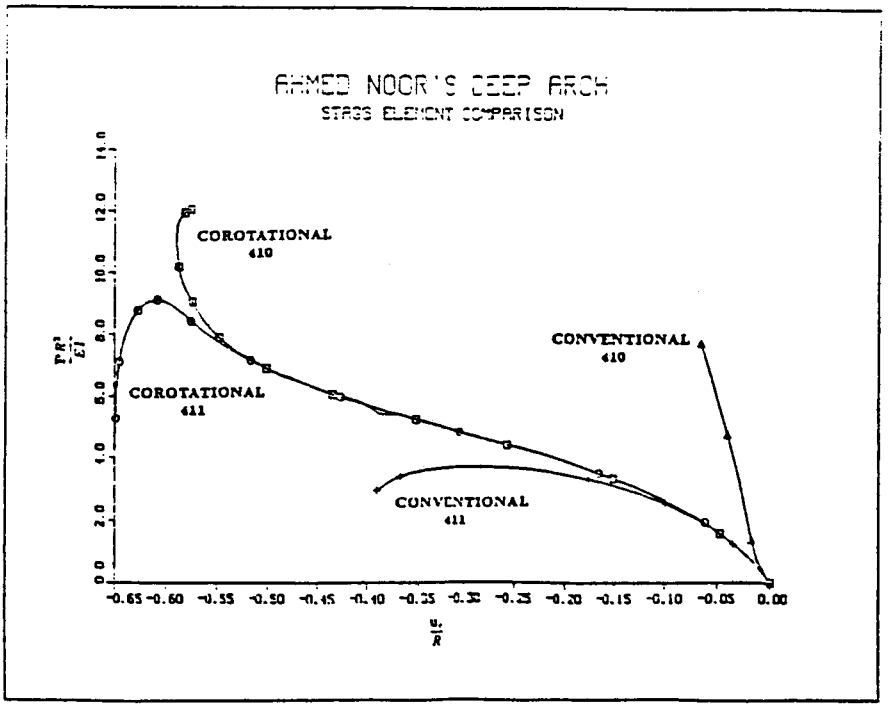
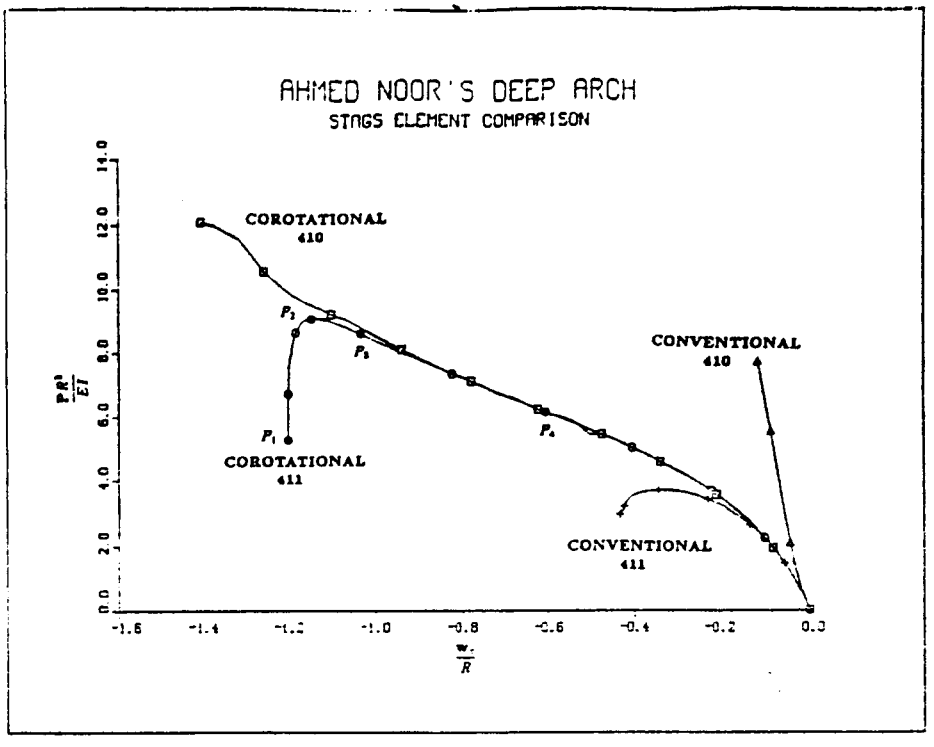
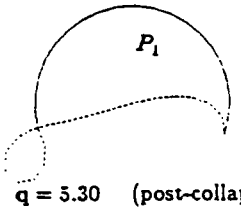
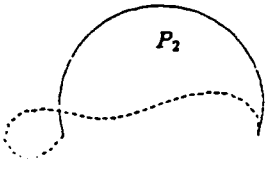


Figure 5. Collapse of a deep arch.



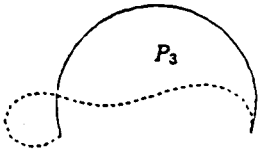


q = 5.30 (post-collapse)

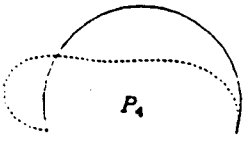


q = 9.09 (limit point)

$$q = \frac{PR^2}{EI}$$

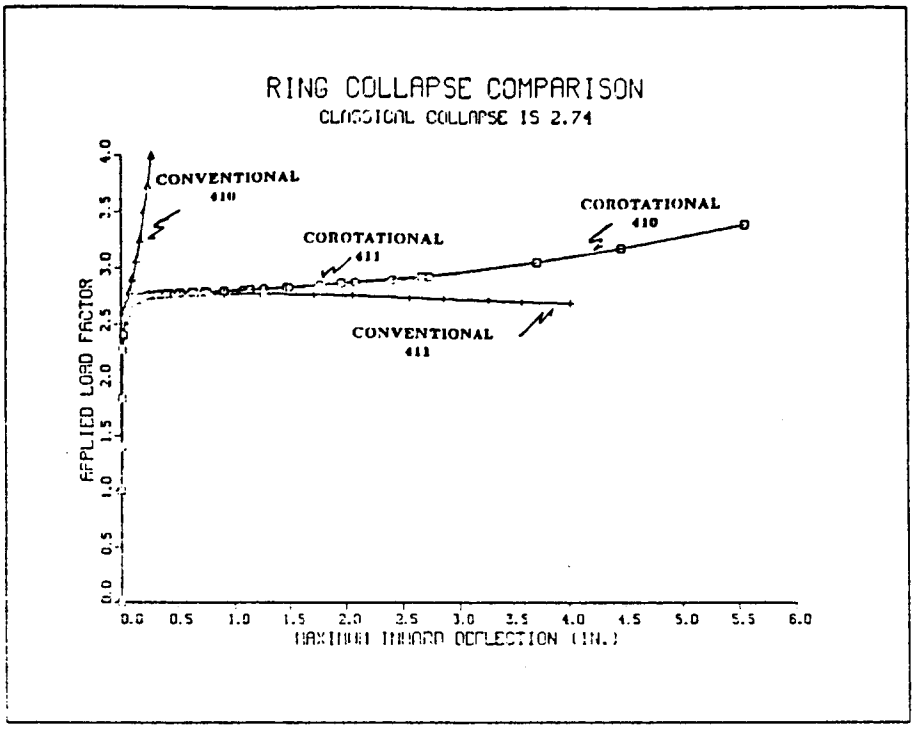


q = 8.44



q = 6.00

Figure 6. Collapse of a deep arch (continued).



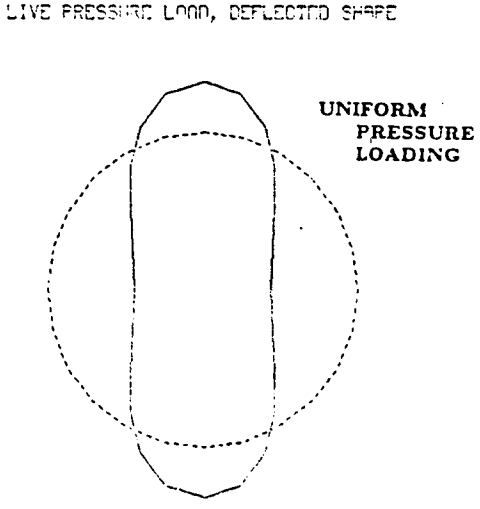


Figure 7. Response of ring under uniform pressure loading.

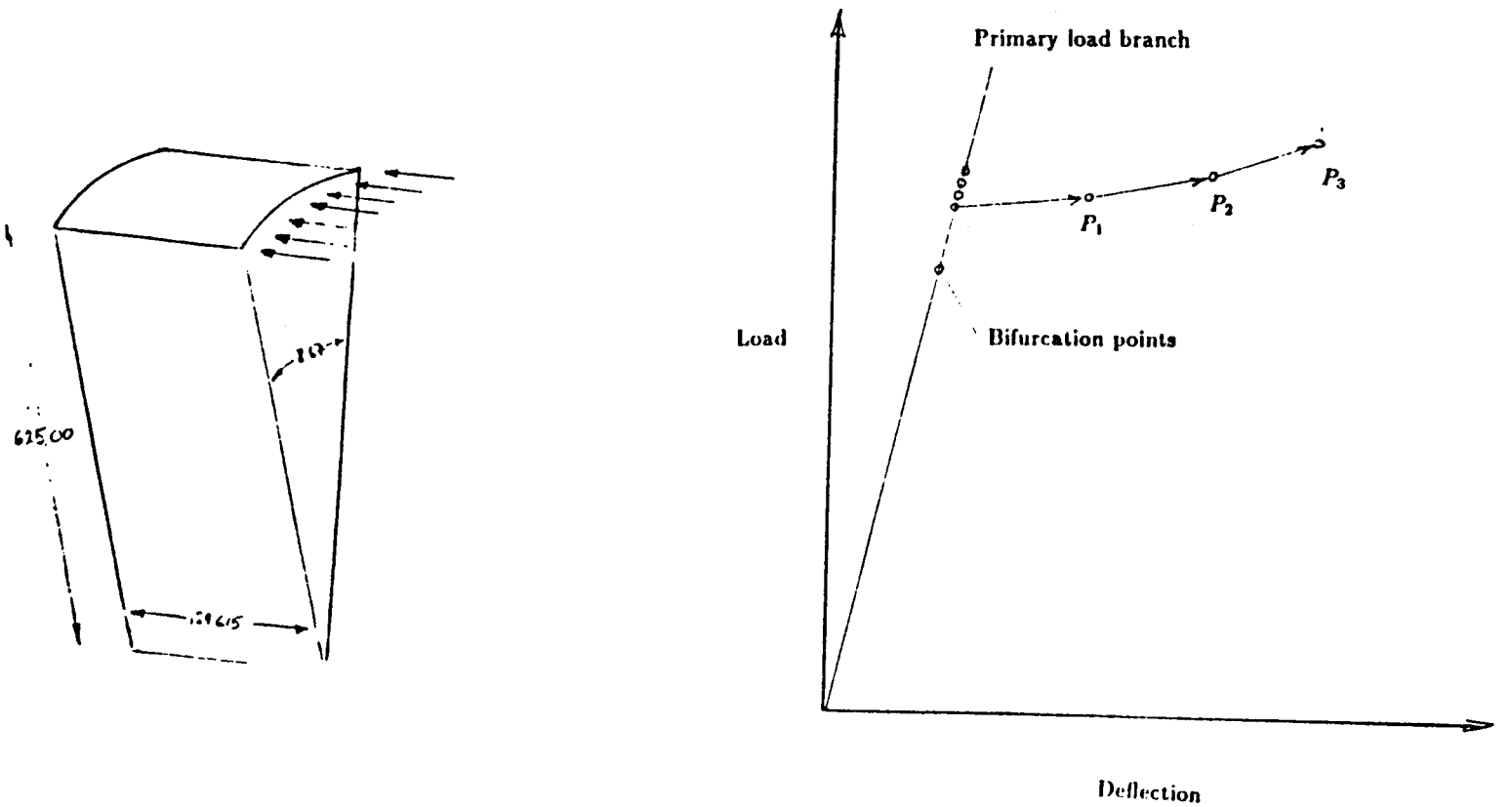


Figure 8. Geometry, boundary conditions, and Thurston steps taken in sample problem

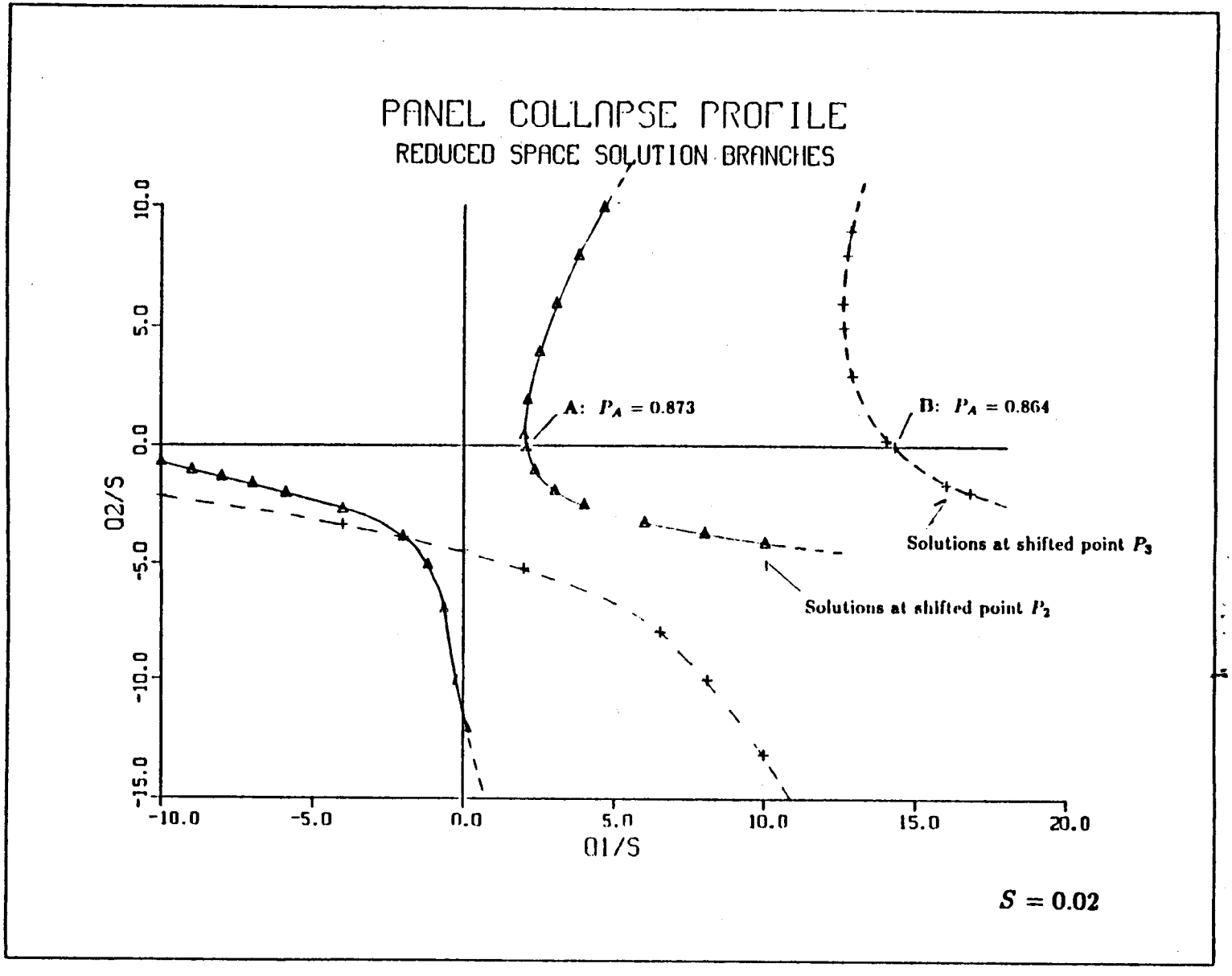


Figure 9. Solution branches in reduced solution space.

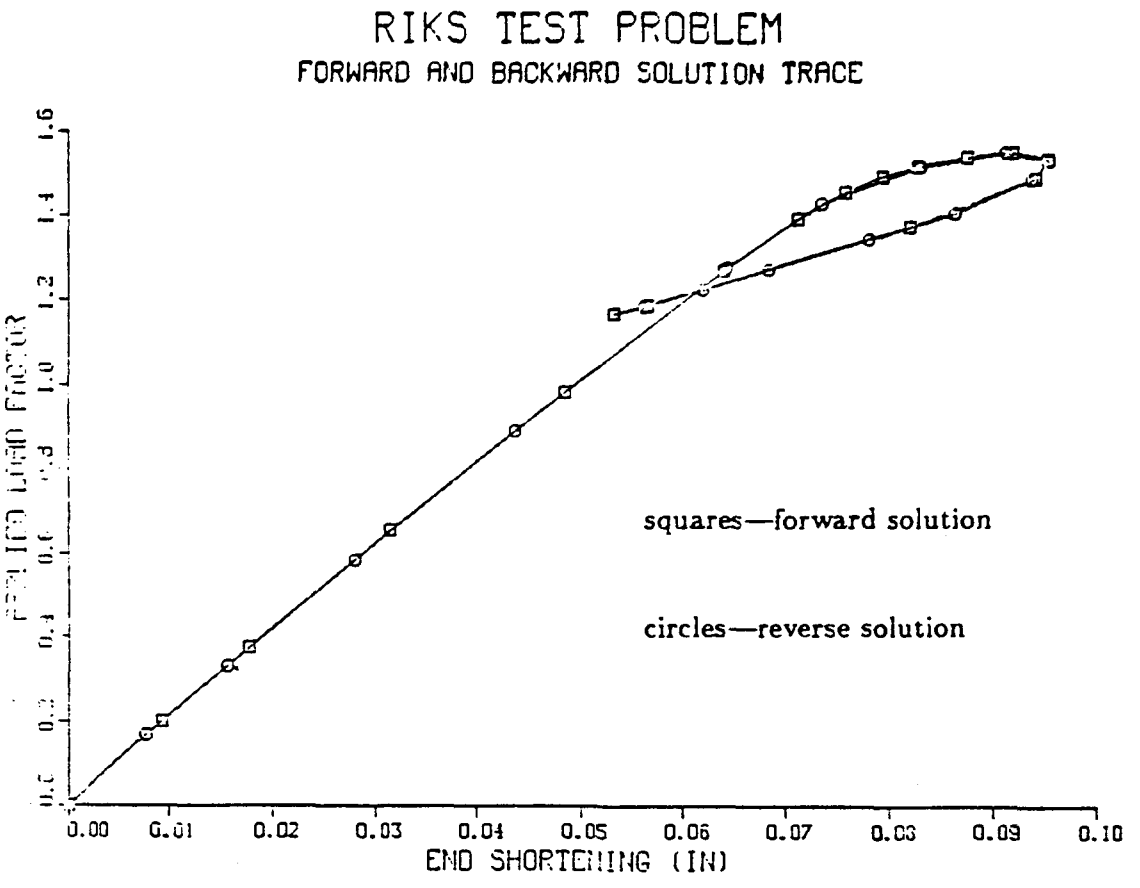


Figure 10. Forward and reverse solution of Riks test case.

APPENDIX 1

THURSTON TRANSFORMATION

PROCESSOR TP

APPENDIX 1—THURSTON TRANSFORMATION PROCESSOR (TP)

Section A1.1—Bifurcation Point Procedure

Original equation system:

$$\bar{\mathbf{F}} = \mathbf{K}_0 \bar{\mathbf{w}} \tag{A1.1}$$

Permute in anticipation of equivalence transformation:

$$\mathbf{F} = \mathbf{K}\mathbf{w} \tag{A1.2}$$

Introduce transformation:

$$\mathbf{w} = \mathbf{T}\boldsymbol{\mu} \tag{A1.3}$$

Transformed equation system:

$$\mathbf{T}^T\mathbf{F} = \mathbf{T}^T\mathbf{K}\mathbf{w} = \mathbf{T}^T\mathbf{K}\mathbf{T}\boldsymbol{\mu} \tag{A1.3.1}$$

Introducing:

$$\boldsymbol{\phi} = \mathbf{T}^T\mathbf{F} \tag{A1.4}$$

$$\mathbf{S} = \mathbf{T}^T\mathbf{K}\mathbf{T}$$

leads to:

$$\boldsymbol{\phi} = \mathbf{S}\boldsymbol{\mu} \tag{A1.5}$$

From the particular choice of the transformation matrix $\mathbf{T}$, it follows that the transformed stiffness matrix has the structure:

$$\mathbf{S} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{bmatrix} \tag{A1.6}$$

where the matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ can be evaluated explicitly:

$$\mathbf{A} = \mathbf{L}^T\mathbf{K}\mathbf{L}$$

$$\mathbf{B} = \mathbf{L}^T\mathbf{K}\mathbf{R} \tag{A1.7}$$

$$\mathbf{C} = \mathbf{R}^T\mathbf{K}\mathbf{R}$$

where $\mathbf{L}$ and $\mathbf{R}$ follow below:

$$\mathbf{T} = \begin{bmatrix} \mathbf{I}_{n_m,n_m} & \mathbf{R}_{n_m,m} \\ \mathbf{0}_{m,n_m} & \mathbf{R}_{m,m} \end{bmatrix} \tag{A1.8.1}$$

and

$$\mathbf{L} = \begin{bmatrix} \mathbf{I}_{n_m,n_m} \\ \\ \mathbf{0}_{m,n_m} \end{bmatrix}, \qquad \mathbf{R} = \begin{bmatrix} \mathbf{R}_{n_m,m} \\ \\ \mathbf{R}_{m,m} \end{bmatrix} \tag{A1.8.2}$$

Here $\mathbf{R}$ is a rectangular matrix of m eigenvectors and $n_m = n - m$.

In preparation of the partitioning process, the conventions are introduced:

$$\phi = \begin{bmatrix} \mathbf{f} \\ \\ \mathbf{p} \end{bmatrix}, \quad \mu = \begin{bmatrix} \mathbf{v} \\ \\ \mathbf{q} \end{bmatrix} \tag{A1.9}$$

Using equations A1.6 and A1.9, the equation system Eq. A1.5 can be written as follows:

$$\begin{aligned} \mathbf{f} &= \mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{q} \\ \mathbf{p} &= \mathbf{B}^T\mathbf{v} + \mathbf{C}\mathbf{q} \end{aligned} \tag{A1.10}$$

By elimination:

$$\begin{aligned} \mathbf{v} &= \mathbf{A}^{-1}(\mathbf{f} - \mathbf{B}\mathbf{q}) \\ \mathbf{p} &= \mathbf{B}^T\mathbf{A}^{-1}(\mathbf{f} - \mathbf{B}\mathbf{q}) + \mathbf{C}\mathbf{q} \\ &= \mathbf{B}^T\mathbf{A}^{-1}\mathbf{f} + (\mathbf{C} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B})\mathbf{q} \end{aligned}$$

From this follows the solution for both $\mathbf{q}$ and $\mathbf{v}$:

$$\mathbf{p} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{f} = (\mathbf{C} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B})\mathbf{q} \tag{A1.11.1}$$

$$\mathbf{v} = \mathbf{A}^{-1}(\mathbf{f} - \mathbf{B}\mathbf{q}) \tag{A1.11.2}$$

Steps in finding a solution:

1.	*Permute* right hand side:	yields $\mathbf{F}$
2.	*Project* right hand side:	yields ϕ
3.	*Decompose* right hand side:	yields $\mathbf{f}$ and $\mathbf{p}$
4.	*Solve* Eq. A1.11.1:	yields $\mathbf{q}$
5.	*Compute* $\mathbf{v}$ from Eq. A1.11.2	yields $\mathbf{v}$
6.	*Assemble* to form μ, Eq. A1.9	yields μ
7.	*Transform* with Eq. A1.3	yields $\mathbf{w}$
8.	*Perform* reverse permutation	yields $\bar{\mathbf{w}}$

Section A1.1.1—Solution Close to a Bifurcation Point

Equilibrium equation close to a bifurcation point may be written in the following form:

$$\mathbf{P} = \mathbf{F} = \mathbf{F}_0 + \mathbf{K}\delta\mathbf{w} + \mathbf{H}(\delta\mathbf{w}) \qquad (A1.12)$$

where:

- $\mathbf{P}$ is the external force vector.
- $\mathbf{F}$ is the vector of internal forces (equals first variation of strain energy).
- $\mathbf{H}$ is a vector containing all higher order terms (ie. the effects which are not included in the linear terms expressed by $\mathbf{K}\delta\mathbf{w}$).

It is assumed that the freedoms in Eq. A1.12 are already reordered in anticipation of the equivalence transformation.

In Eq. A1.12, $\mathbf{F}_0$ is the first variation at the pole of the expansion and $\delta\mathbf{w}$ are displacement increments with respect to that pole.

Note that $\mathbf{H}$ can be computed numerically from Eq. A1.12 by considering the first variation at a number of points close to the pole.

Step 1: *Apply coordinate transformation, change basis vectors as proposed by Gaylen Thurston with his equivalence transformation. Then break down the equation system into two separate equation systems.*

$$\begin{aligned} \mathbf{T}^T\mathbf{P} &= \mathbf{T}^T\mathbf{F} \\ &= \mathbf{T}^T\mathbf{F}_0 + \mathbf{T}^T\mathbf{K}\delta\mathbf{w} + \mathbf{T}^T\mathbf{H} \\ &= \mathbf{T}^T\mathbf{F}_0 + \mathbf{T}^T\mathbf{K}\mathbf{T}\delta\boldsymbol{\mu} + \mathbf{T}^T\mathbf{H} \end{aligned} \qquad (A1.14)$$

Using previous terminology:

$$\boldsymbol{\phi} = \mathbf{T}^T\mathbf{F} \qquad (A1.15.1)$$

$$\mathbf{S} = \mathbf{T}^T\mathbf{K}\mathbf{T} \qquad (A1.15.2)$$

and introducing:

$$\boldsymbol{\pi} = \mathbf{T}^T\mathbf{P} \qquad (A1.15.3)$$

$$\mathbf{h} = \mathbf{T}^T\mathbf{H}(\delta\mathbf{w}) \qquad (A1.15.4)$$

Eq. A1.14 transforms into:

$$\boldsymbol{\pi} = \boldsymbol{\phi}_0 + \mathbf{S}\delta\boldsymbol{\mu} + \mathbf{h} \qquad (A1.16)$$

To prepare for the partitioning process, the definitions given by Eq. A1.9 are introduced:

$$\boldsymbol{\phi} = \begin{bmatrix} \mathbf{f} \\ \\ \mathbf{p} \end{bmatrix}, \qquad \boldsymbol{\mu} = \begin{bmatrix} \mathbf{v} \\ \\ \mathbf{q} \end{bmatrix}$$

Defining in addition:

$$\pi = \begin{bmatrix} \pi_u \\ \\ \pi_l \end{bmatrix}, \qquad \mathrm{h} = \begin{bmatrix} \mathbf{h}_u \\ \\ \mathbf{h}_l \end{bmatrix} \tag{A1.17}$$

Eq. A1.16 can be rewritten as follows in the form of two separate equation systems:

$$\begin{aligned} \pi_u &= \mathbf{f}_0 + \mathbf{A}\delta\mathbf{v} + \mathbf{B}\delta\mathbf{q} + \mathbf{h}_u \\ \pi_l &= \mathbf{p}_0 + \mathbf{B}^T\delta\mathbf{v} + \mathbf{C}\delta\mathbf{q} + \mathbf{h}_l \end{aligned} \tag{A1.18}$$

Since $\mathbf{H}$ and thus $\mathbf{h}$ are functions of all freedoms involved, these systems are coupled. However, in the vicinity of a bifurcation point it may be assumed that at least $\mathbf{h}_u$ is a function of the modal variables only. This permits $\delta\mathbf{v}$ to be expressed uniquely by the modal variables (Eq. A1.18). By partitioning, $\delta\mathbf{v}$ can therefore be eliminated from the ensuing equation system. Hence, we end up with a very small system in the modal variables only, from which $\delta\mathbf{q}$ can be determined by any available means.

Basic assumptions:

$$\mathbf{h} = \mathbf{h}(\delta q_i) \tag{A1.19}$$

By this assumption from Eq A1.18.1

$$\delta\mathbf{v} = \mathbf{A}^{-1}(\pi_u - \mathbf{f}_0 - \mathbf{B}\delta\mathbf{q} - \mathbf{h}_u) \tag{A1.20}$$

where the expression of the right side is now a function of $\delta\mathbf{q}$ only. While $\mathbf{h}_u$ is not explicitly known, it can at least be numerically computed and then be approximated by an analytical expression.

Introducing Eq. A1.20 into Eq. A1.18 one obtains an equation system for the δq_i only:

$$\pi_l = \mathbf{p}_0 + \mathbf{B}^T\mathbf{A}^{-1}(\pi_u - \mathbf{f}_0 - \mathbf{B}\delta\mathbf{q} - \mathbf{h}_u) + \mathbf{C}\delta\mathbf{q} + \mathbf{h}_l \tag{A1.21}$$

Moving all terms which are potentially functions of the δq_i to one side, constant terms to the other, Eq. A1.21 can be written in the form:

$$\pi_l - \mathbf{p}_0 - \mathbf{B}^T\mathbf{A}^{-1}(\pi_u - \mathbf{f}_0) = \mathbf{C}\delta\mathbf{q} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\delta\mathbf{q} + \mathbf{h}_l - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{h}_u$$

As slightly rewritten for convenience:

$$\pi_l - \mathbf{p}_0 - \mathbf{B}^T\mathbf{A}^{-1}(\pi_u - \mathbf{f}_0) = (\mathbf{C} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B})\delta\mathbf{q} + \mathbf{h}_l - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{h}_u \tag{A1.22}$$

This is the reduced equation system which must be solved. To make a solution possible, the term

$$\mathbf{h}_l - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{h}_u \tag{A1.23}$$

is expanded into a polynomial of δq_i. Since the loads are assumed to be linearly dependent on the load factor P_A, the left hand side can be written in the form:

$$\mathbf{Lhs} = P_A \mathbf{l}_A + \mathbf{l}_{base} \qquad (A1.24)$$

Eq. A1.22 can then be solved for any convenient combination of unknowns.

Step 2. *Solve reduced equations.*

Step 3. *Back substitution to fine* $\delta\mathbf{v}$, $\delta\boldsymbol{\mu}$

Step 4. *Transform and permute to find* $\delta\mathbf{w}$, $\delta\bar{\mathbf{w}}$

Section A1.2—TP Computer Implementation

The procedure described in section A1.1 has been implemented for the STAGS programs in subroutine TP. Because of the inherent complexity of this procedure, it has been considered essential to develop the program in a concise and clear manner which would also facilitate future changes. Accordingly, the matrix and vector operations are all executed by calls to a set of routines, called the Z-system. These routines have been designed so that the arguments contain precisely the minimum information which is required to specify some step in the procedure. The use of the Z-system first requires the establishment of vector and matrix files and their identifying symbols. This initialization step is performed in subroutine INTP. The symbol definition list (analogous to the nomenclature section of a journal article) is given in section A1.2.1 and A1.2.2. A list of the Z-system operations for quick reference is provided in section A1.2.3. More detail concerning each routine is given in a comment section in the subroutine itself.

Section A1.2.1—Description of matrix and vector files

File	*Description*
A	Assembled/factored stiffness matrix (n_m, n_m)
B	$= \mathbf{KE}$; (n_m, m) matrix
BTAI	$\mathbf{B}^T\mathbf{A}^{-1}$; (m, n_m) matrix
DR	$= \mathbf{K}\delta\mathbf{x}$; vector (linear part of first variation)
DX	Displacement increment
E	Set of m eigenvectors
FORCA	External force vector (system A)
FORCB	External force vector (system B)
HU	Set of vectors of higher order residual terms
K	Element stiffness matrix file
R	Residual force vector
R0	Residual vector (first variation) for $\mathbf{x}_0$
SKY	Sky-line vector (semi-bandwidth)
X	Displacement vector
X0	Displacement vector for iteration $(\mathbf{x}_0)$
X1	Converged solution for last step (ISTEP-1)
XX	Set of increments for perturbed vectors

Section A1.2.2—Description of variables used.

Variable	*Description*
DNORM	Euclidean norm of displacement vector
EPSD	Convergence criteria for displacement error
EPSL	Convergence criteria for residual error
ISEC	Maximum number of seconds permitted in execution
ISTEP	Load step number
HL	$= \mathbf{h}_l - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{h}_u$
IE	Number of eigenvector used for step from bifurcation point or from other solution (Normally the largest (component is chosen)
M	Number of eigenvectors used
N	Number of freedoms of system vector
NPOLY	Number of perturbed solutions for polynomial
PA	Initial load factor (system A)
PB	Initial load factor (system B)
QFAK	Modal amplitudes solved for in reduced equation
RNORM	Euclidean norm of residual force vector
STEP	Length of step from X1 along eigenvector

Section A1.2.3 Description of Z-system operations on vectors and matrices.

Subroutine	*Operation*
ZADD(V1,S1,V2,S2,V3,S3)	[V3]=S1*[V1]+S2*[V2]+S3*[V3]
ZASS(SKY,K1,K2,S,A)	[A]=Assembled matrices [K1]+S*[K2]
ZDOT(V1,V2,S)	S=[V1]·[V2]
ZFACT(A,V)	[A]= Factored form of [A]-Diag[V]
ZFIX(V1,V2)	If $[V1]_i$=0, then $[V2]_i$=0. i=1,N
ZGET(V,N,ID,SS)	SS(I)=$[V]_{ID(I)}$ I=1,N
ZMAX(V,S,I)	S=$[V]_I$, where |S| is max elt in V
ZMOVE(V1,V2)	[V2]=[V1]
ZMULT(V1,V2,V3)	$[V3]_i$=$[V1]_i$*$[V2]_i$
ZMXVEC(K,V1,V2,N)	[V2(I)]=[K]*[V1(I)] I=1,N
ZPRINT(V,ISTEP,PA,PB)	Print [V] in displacement vector form
ZPUT(V,N,ID,SS)	$[V]_{ID(I)}$=SS(I) I=1,N
ZRAND(V,I)	[V]= Random vector in range: I=0 between {0.,1.} I=-1 between {−1.,1.}
ZSET(V,S)	$[V]_i$=S i=1,N
ZSMUL(F,S)	$[F]_i$=S*$[F]_i$ i=1,N

Subroutine	Operation
ZSOLVE(A,N,V)	[A]*[V(I)]=[V(I)] I=1,N
ZVAR1(V1,ILIN,IPLAST,ISTEP, PA,PB,V2)	[V2]= First variation of [V1] ILIN IPLAST 0 - Linear No plasticity 1 - Non-linear Plasticity
ZVAR2(ILIN,ISTAB,V,IPLAST, ISTEP,PA,PB,K,SKY)	[K]= Second variation of [V] [SKY]= Skyline vector for [K] ILIN IPLAST 0 - Linear No plasticity 1 - Non-linear Plasticity ISTAB=0 - [K]=Total stiffn. matrix ISTAB=1 - [k]=Stability matrix

---

Scalars used in Z-system calls are always in the highest precision used in the program.

Symbols used in the summary of Z-system functions are:

S	Scalar
SS	Scalar array
I	Integer
ID	Integer array
[A]	Assembled stiffness matrix
[F]	Any type of file
[K]	Element stiffness file
[SKY]	Skyline vector (semi-bandwidth in integers)
[V]	Vector file (or set of vector files)

APPENDIX 2

1985 CHANGES TO

1983 STAGS-C1 MANUAL

APPENDIX 2

1985 Changes to 1983 STAGS-C1 Manual

PREFACE

The 1985 STAGSC-1 release contains many new features that extend the power of the program significantly. The most important improvements are as follows:

1. **Large Rotations.** STAGS nonlinear collapse analyses are no longer restricted to moderate rotations. All collapse runs (INDIC=3, Record B-1) default to the large rotation corotational formulation unless specifically overridden by the user. Presently, bifurcation, vibration, and transient analyses are not affected.

2. **Negative Load Increments.** Users of the STAGS nonlinear collapse options have much greater control of the load incrementation procedure than was previously possible. The solution can be forced to stop at user-specified load factors contained in a list for all nonlinear collapse and bifurcation runs. There are two types of negative load increment runs:

   a. *Riks*, or path-length incrementation, where the independent load variable is a measure of the arc length along the current load path, and where the load step is calculated as a dependent variable. (The Riks capability was added to STAGS-C1 in 1983.)

   b. All other run types, where loads are specific functions under strict user control.

   For runs of the first type (Riks), the user can restart an analysis at any point, and he can reverse the direction of the run if desired. The INDIC =4 option has been extended to include Riks runs of all types, as explained later in this Appendix. For non-Riks runs under load control, the user can also restart at load level and reverse the direction at any time.

3. **Nonlinear Mount Elements.** Mount elements have been added to STAGS. As the name implies, mounts have been designed to simulate points of attachment either between units of a given structure, or attachments to ground. Mounts are generalized nonlinear springs with rigid links, the force of which can be any piecewise linear function of relative spring displacement or velocity. Thus dampers or rubber mounts can be simulated with the new 110 STAGS mount element. A concise description of the theory and input is included in Ref. [12].

4. **Creep and Isotropic Hardening Plasticity.** A power-law creep algorithm has been added to STAGS. Isotropic strain hardening plasticity has been enhanced,

including the option of a tangent stiffness for more rapid convergence. Further documentation can be found in Ref. [13].

5. **Explicit Central Difference Time Integration.** The explicit time integration option (IMPL=1, Record E-2) has been generalized by the addition of the STINT-CD central difference integrator. An important new feature is the ability to let the program select the largest possible stable time increment as a function of run parameters. A short documentation of the STINT-CD option is included in Ref. [14].

6. **Timoshenko Beam Element.** A modification to STAGS beam elements (B210) has been made to account for transverse shear deformations. The theory and input modifications are explained in the same document as for STINT-CD (Ref. [14])

7. **Displacement and Load Histories.** The user has the option of writing a program for extracting displacement and load histories specified as *input* to STAGS2. The data are stored on an auxiliary Fortran unit, and entry points are provided for the user to extract the values and process the results. This capability is available only for Riks nonlinear collapse runs and STINT-CD explicit transient time integrations for CDC and VAX-VMS machines.

8. **Imperfections as Perturbed Geometry.** For large rotation runs, trigonometric imperfections are now added to the input geometry, in effect generating a new starting model containing the desired imperfections. For those who desire to calculate imperfections with a user-written subroutine, a new subroutine DIMP has been added for perturbing the initial geometry. The DIMP subroutine, to be described subsequently, replaces WIMP for large rotation runs. DIMP is available for all classes of runs.

9. **Conversion of Solution Data File.** The solution data file (often called SOD, always accessed as Fortran unit 22) cannot be transported from one type of computer to another because, for reasons of efficiency, it is unformatted (binary). A translator has been created that converts the SOD file into a formatted ASCII "FSD" file that can be copied from one type of machine to another. A back translator reconverts the formatted data into the original SOD data structure for use in POSTP, STAPL, or a STAGS restart. Thus it is now easy to generate SOD data with STAGS on a huge "number cruncher" machine (such as CDC or CRAY), convert the data to the FSD format, copy the ASCII over to a satellite interactively-oriented machine such as a VAX for reconversion to the SOD format for postprocessing and plots. Use of the translator is described in a subsequent section.

10. **Numerous Bug and Documentation Fixes.** Much of what follows reflects careful study and collective experience using the STAGS manual. The corrections

that are described are designed to tide the user over until a completely new release of the manual is available (by late 1986).

The corrections follow in three groups, beginning with a record-by-record account of important changes to the STAGS1 input and user-written subroutines. Following this table, there is a list of the most relevant changes to the manual text, indexed by the affected page number of the STAGS-C1 User's Manual (Ref. [1]); this list covers the most essential sources of confusion. The last part consists of sections pertaining to the new features mentioned above, including the required input to STAGS.

CHANGES TO INPUT AND USER ROUTINES

Page no.	*Record*	*Entry*	*Variable*	*Description*
3.7	B-1	10	ICOR	0—Corotational procedure is used 1—Non-corotational procedure is used
3.7.1	B-2	7	NIMPFS	*Must* be 0 if IWIMP$\neq$0 (Record M-5, p3.71)
3.7.1	B-3	5	NTMT	Number of tables for mount elements (nonlinear springs)†
3-10	C-1			Negative load increments have been added‡
3-11	C-1	4	STLD(2)	For a creep analysis
3-11	C-1	5	STEP(2)	(ICREEP=1, Record I-1), load
3-11	C-1	6	FACM(2)	system B data is replaced by creep time data: initial time, time increment, and final creep time, respectively*
3-11	C-1	9	IXSTF	Cannot be used in conjunction with subroutine UCONST
3-17	D-2	1	NCLUST	One cluster per run: *set* NCLUST to *unity.*
3-19	E-1	6	BETA	Damping factor β must be zero for explicit integration
3-20	E-1	8	THOLD	Not used for explicit integration
3-26	F-2	4	NSKEW	Do not use (set NSKEW=0)

†*Refer to* "110 Mount Element," *Ref. [12].*

‡*Refer to section entitled* Negative Load Increments.

**See Ref. [13]*

Page no.	*Record*	*Entry*	*Variable*	*Description*
3-29	G-2	2	IR1	If IR1=0, then IR2 *must* be 0
3-29	G-2	3	IC1	If IC1=0, then IC2 *must* be 0
3-30.2	H-1	2	NSPRI	Number of mount elements†
3-31	I-1	2	NESP	NESP cannot be greater than 10
3-31	I-1	5	ICREEP	0—no creep 1—creep
3-34	I-3			If ICREEP>0 (Record I-1), go to I-3A NTMT=number of mounts (Record B-3) If NTMT>0, go to I-4a
3-34.1	I-3A	1	ACO	Creep constant A,
3-34.1	I-3A	2	BCO	creep constant B,
3-34.1	I-3A	3	EXPM	creep exponent m, and
3-34.1	I-3A	4	EXPN	creep exponent n for the relation $\bar{\epsilon}^c = A(\bar{\sigma}/B)^m \bar{t}^n$.‡
I-4	Mount	Record		
3-34.1	I-4a			Mount element table
3-34.1	I-4b			information
3-34.1	I-4c			I-4a through I-4d are
3-34.2	I-4d			*new* records†
3-58	M-1	2	IGLOBE	4—Location of unit origin (x_g, y_g, z_g) (one translation record and one rotation record) 5—Location of corner point #1 (one translation record and one rotation record)

†*Refer to* "110 Mount Element," *Ref. [12].*

‡*See Ref. [13]*

Page no.	*Record*	*Entry*	*Variable*	*Description*
3-60	M-2A			IGLOBE=4 or 5, go to M4-D
3-65	M-2A			For ISHELL=11, the formula should be $X^e = \arctan(\frac{R_\epsilon}{R_{yz}} \tan X)$
3.71	M-4D			Special Shell Unit Record (*new*) (global position coordinates)
3.71	M-4D	1	XG	x_g
3.71	M-4D	2	YG	y_g
3.71	M-4D	3	ZG	z_g
3.71	M-4E			Special Shell Unit Record (*new*) (Eulerian rotation)
3.71	M-4E	1	XGDOT	$\dot{x}_g$
3.71	M-4E	2	YGDOT	$\dot{y}_g$
3.71	M-4E	3	ZGDOT	$\dot{z}_g$
3.71	M-5	2	IWIMP	*Must* be zero if NIMPFS (record B-2) is non-zero.
3.71	M-5	2	IWIMP	Set to -2 if perturbed initial geometry is desired, and subroutine DIMP must be used.
3.71	M-5	6	IPLAS	0—elastic behavior only. 1—plasticity included. 2—deformation theory.
3-73	M-6			w_0 is the imperfection amplitude
3-73	M-6	6	ID	1—u imperfection 2—v imperfection 3 or 0—w imperfection
3-84	O-1A	5	IPLAS	0—elastic behavior only. 1—plasticity included. 2—deformation theory.
3-87	O-2A	6	IPLAS	0—elastic behavior only. 1—plasticity included. 2—deformation theory.

Page no.	*Record*	*Entry*	*Variable*	*Description*
3-87.2	O-3			Skew stiffeners do not work.
3-97	Q-3	2	LT	The statement: (Irrelevant for initial conditions.....) should be *replaced* by: (For initial conditions, i.e., velocities and displacements at time T=0, set LT to −1).
3-109	T-1			*Discard* and *replace* by Mount Element Record†
3-111	T-2	10	IPLAS	0—elastic behavior only. 1—plasticity included. 2—deformation theory.
3-112	T-3	9	IPLAS	0—elastic behavior only. 1—plasticity included. 2—deformation theory.
3-113	T-4	10	IPLAS	0—elastic behavior only. 1—plasticity included. 2—deformation theory.
3-113	T-4	12	IPENL	0—*Enforce* penalty 1—*Relax* penalty
5-3	PL-3	1	KPLOT	2—(Solution contours) plot *one* unit at a time *only*
5-3	PL-3	3	IDISC0	Option -1 refers to KPLOT = 1, *not* 2.
5-3	PL-3	2	IUNIT	*Delete* the word *"shell"* wherever it occurs.
5-4	PL-3	7	MOSY	*Replace* Meaningful only for: *with* For: *Add* (iv) otherwise, set MOSY=0.

†*Refer to* "110 Mount Element," *Ref. [14].*

Page no.	*Routine*	*Entry*	*Variable*	*Description*
4-7	UGRID			*Remove* the statement: `IF (IUNIT.NE.1) RETURN`
4-33	UCONST	3	IX(I)	*Remove* first description of IX: the second description supersedes it.
4-37	UPRESS			*Replace* `PRESS=.....` *with* `PRESS=(FLO(1)-FLO(3)*` `COS(YR))*FLO(2)*PA`
4-42	QUAD	11	INTEG	The variable INTEG *should* occupy position 11, *not* 6.
4-42	QUAD	12	IPENL	IPENL *should* occupy position 12, *not* 7. The order of the variables in QUAD is: N1, N2, N3, N4, KQUAD, IWALL, ZETA, ECZ, ILIN, IPLAS, INTEG, IPENL. IPENL has same meaning as record T-4.

Note also that all references to SPRING elements (p6-13 and p6-20) are superseded by the mount element documentation [14].

CHANGES TO MANUAL TEXT

Page 3-12 First sentence on page should contain reference to C-1, not C-2 (in parentheses).

Page 3-15 Delete "as presently formulated, it is likely to cause difficulties with live loading", near bottom of page.

Page 3-22 Please refer to Ref. [13] for more documentation on the new enhanced automatic explicit time integrator.

Page 3-28 The following text replaces the description of the G-2 record in the manual:

This record is included only if partial compatibility between displacements at specified nodes is defined, NPATS > 0 (B-2)

If G-2 records are used to connect shell units, IGLOBE (M-1) must be 0, 3, 4, or 5.

A shell unit displacement component is identified by unit number, row, column, and component (direction) numbers. A displacement component in the element unit is identified by node number (IR1), a zero for IC1, and a value for the direction (ID1). Each G-2 record provides 8 items defining a pair of displacement components.

Since no transformations are preformed, the nodes referred to should not lie on a boundary line involved in a shell unit connection (G-1) unless the two shell units have the same orientation with respect to the global coordinates. *If a row or column is set equal to zero, the compatibility condition applies to an entire column or row, respectively.*

In contrast to shell unit connections (G-1, NINTS > 0 Record B-2), or boundary conditions (P records), the partial constraints are not restricted to boundary lines.

Page 3.71 A new user subroutine DIMP has been added which is flagged by setting IWIMP to -2. DIMP *must* replace WIMP for corotational collapse analyses. If IWIMP > 0, then the correct perturbed initial geometry is automatically introduced with the trigonometric imperfections when using corotational theory. Failure to use DIMP instead of WIMP will void imperfections for corotational collapse runs.

Page 3.77 The logic at the bottom of the page for determining the next record to be read should be replaced by the following:

```
NNX = -1, go to N-4
NNX > 0, go to N-2
NNX = 0, NNY = -1, go to N-7
         NNX = 0, NNY > 0, go to N-5
```

```
                    KELT = 0, go to N-9
                    KELT > 0, IRREG = 1, go to N-8
                              IRREG = 0 *
```

*Follow instruction at end of N-9B.

Page 3-82.4 Numbers in column headed by KELT should end in 2, not 3.

Page 3-89 All references to Record C-1 should be changed to Record F-1.

Page3-106 In the fourth paragraph, IUS = 0 should be replaced by IUS > 0.

Page 4-16 Add Subroutine Dimp, as follows:

```
      SUBROUTINE DIMP (IUNIT,X,Y,VV)
C
C     DIMP requires the user provide output in the array VV.
C     Imperfections in VV are added to the initial geometry as follows:
C
C     VV(1) -- U, to be added to X coordinate (in-plane)
C     VV(2) -- V, to be added to Y coordinate (in-plane)
C     VV(3) -- W, to be added to the radial (transverse) direction.
C
C     INPUTS
C
C     IUNIT -- Shell Unit
C         X -- Axial shell coordinate
C         Y -- Hoop shell coordinate (degrees)
C
C     OUTPUT
C
C        VV -- Array of three displacements to be added to initial
C              geometry.  All displacements must be in units of length.
C
      DIMENSION VV(3)
C
C     Example User Input for a trigonometric representation of
C     perturbed geometry for a cylinder of length 15, radius 25.
C     User programming replaces example:
C
      SPAN=15.
      PI=3.14159
      RADIUS=25.
      WMAX=.01
      WX=WMAX*(PI/SPAN)*COS(PI*X/SPAN)*SIN(PI*Y/180.)
      WY=WMAX*SIN(PI*X/SPAN)*(PI/RADIUS)*COS(PI*Y/180.)
      VV(1)=WX**2*SPAN/2.
      VV(2)=WY**2*RADIUS/2.
      VV(3)=WMAX*SIN(PI*X/SPAN)*SIN(PI*Y/180.)
```

```
C
C     End Example
C
      RETURN
      END
```

Page 4-45 Input parameter"K" should read ISYS and have same meaning as in Record Q-2. ISYS must *never* be changed in user subroutines. X and Y are *arrays* containing the coordinates of all nodes (row and column intersections) in the shell unit IUNIT; users must set all pertinent loads for IUNIT at one time.

Page 4-55 The description of GROUP E should contain "NSMRS>0" instead of "NSMRS=0". The same applies to the NSMRS=0, p. 4-58; replace by NSMRS>0.

Pp 4-57,4-58 Dimensions in the common blocks /WALL1/ and /WALL2/ *must* be increased from 30 to 50.

Page 4-64 In example subroutine, increase dimensions from 30 to 50.

Page 4-66 Use WIMP only for analyses containing small rotations, and with the corotation in STAGS turned off (by setting ICOR to 1, Record B-1). DIMP replaces WIMP for all other cases, and also can be used for noncorotational runs. Flag for DIMP (perturbation of initial geometry) by setting IWIMP to -2, Record M-5. (Note that at present, corotation is automatically turned off for bifurcation, vibration, and transient analyses.)

NEGATIVE LOAD INCREMENTS

STAGS now has the added ability to back out of a nonlinear solution to locate bifurcation points, or to calculate residual deflection for nonlinear material problems, adding greatly to the overall flexibility the user has when attacking difficult nonlinear problems. The input in STAGS1 has been modified as follows:

Record **C-1**

STLD(1) Starting load step. Can be positive or negative.

STEP(1) Signed load increment. If it is positive, the increment will be *added* to STLD; if negative, the absolute value of STEP will be *subtracted* from STLD. In an initial Riks run (NSTRAT=-1, Record D-1), the solution will be executed under load control for the first load step, and the path length parameter DETA will be adjusted according to the solution behavior for subsequent steps. The absolute value of STEP is added or subtracted from the initial

load value as with load control. For restart runs, if load control is used, STEP will be used the same way as for initial runs. For Riks runs, however, STEP will be an initial estimate for the path length parameter, giving the user greater control of his restart. Values of DETA used in previous iterations are always printed with the iteration output. If a negative value of STEP is chosen, the Riks procedure is automatically instructed to *go in the reverse direction.*

FACM(1) The signed final load value at which the analysis will be terminated. For load control runs, termination will be handled according to the sign of STEP: if STEP is *greater* than zero, the solution is terminated when the load *exceeds* FACM; if *less* than zero, the solution is terminated when the load is *less* than FACM. For Riks runs, the termination criterion is determined by the sign of the difference FACM−STLD: if it is *greater* than zero, the solution stops when the load *exceeds* FACM; if it is *less* than zero, the solution stops when the load is *less* than FACM.

In a Riks run, the parameters for the B load are interpreted in the same manner as the A load, except that when the B load reaches its final value, only its incrementation is terminated; the run continues until the criteria for the A system are satisfied. All other parameters on the **C-1** record are unchanged.

Note that a Riks restart is always begun with the user providing a positive STEP *unless* he wants to *reverse* the path at that point. Even if the solution appears to be reversing, or if a new restart is desired during a reverse, a *positive* input value for STEP is required.

For INDIC=4, both the RIKS and load control strategies work in either forward or reverse direction; it is important, however, that the user choose his list of load factors for which solution points are required in the proper order. For example, in an ordinary load control case, the load values are to be selected in ascending order. In a *reverse* load control case, the load list will be in descending order. For Riks runs, the list must be chosen as an *expected* sequence of load factors predicted to occur along the load path. Such a choice may become complicated for strongly nonlinear Riks cases, as can be seen from the load-deflection trace in Figure 10.

STAGS SOLUTION DATA TRANSLATOR

Now that high speed networks connect machines from different vendors, it is often desirable to perform the bulk of the STAGS "number crunching" on huge, batch-oriented computers such as CRAY or CDC, leaving postprocessing of all kinds to smaller, more accessible machines such as a VAX or a workstation. The principal impediment for STAGS has been that the solution data file, often referred to as "SOD" (Fortran unit 22) is written in machine binary and cannot be easily transported across vendor lines. This is why the

program UNFFMT has been written. UNFFMT automatically converts all displacements, eigenvectors, corotational data, plastic strain histories, and velocities to a Fortran formatted file with a very simple and transparent data structure. In contrast to the binary file, this ASCII file is easily transported with resident machine utilities across the network (or by tape) to the interactive destination machine. There, the companion program FMTUNF reconverts the data back into the original SOD format for use in postprocessing, graphics, or a new STAGS restart. All secondary data (strains, stresses, resultants and the like) are efficiently re-created with a STAGS1 and POSTP execution. UNFFMT and FMTUNF never need any input data; the only requirement is access to the SOD file by fortran unit 22. The formatted "FSD" data is produced *and* read on unit 23. Specifics for a given machine are contained in the documentation with the STAGS-C1 source program tapes.

APPENDIX 3

INSTRUCTIONS FOR WRITING A DISPLACEMENT HISTORY DATA RETRIEVAL SUBROUTINE

History plots of selected displacements can be generated from data placed on Fortran files STAGS.PLT (VAX-VMS), or STGPLT (CDC), with very simple data provided on unit FOR075 (VAX-VMS), or the primary input unit (CDC), as follows:

RECORD 1 -- CASE TITLE

RECORD 2 -- NDSPS - Number of displacements, up to ten, for which histories are desired.

RECORD 3 to NDSPS+2 (NDSPS records):

IUNIT,IDU,IROW,ICOL

where

IUNIT --- Shell unit no.
IDU --- Freedom type (1 thru 6 for u,v,w,Ru,Rv,Rw)
IROW --- Row number for node in question
ICOL --- Column number for node in question

For collapse or explicit integration runs, STAGS will save data with the titles XXXX.LOAD, XXXX.DIS1,...,XXXX.DS10, where XXXX represents the first four letters taken from the case title (Record 1). Data structure is explained below.

A description follows of all the operations that involve the creation and retrieval of history data. Users need only be concerned with the REOPEN and READ operations to be performed on data already saved by STAGS.

SUMMARY OF OPERATIONS WITH HISTORY DATA

PDATA has four entry points for the separate functions needed to create PLOT data sets. These functions are:

(1) Open DATA library and initialize all the data blocks needed.

(2) Write fixed-length records on to a given data set with a given index.

(3) Close out library and flush any buffers (after last write)

(4) Reopen library in preparation for data retrieval

(5) Read previously-stored record by data set name back into a user workspace.

First function--OPEN data sets (STAGS does this automatically).

Entry PDATA must be called only once during an execution. The user must provide the name of a new external file on which the library will reside. He must provide a single name which describes in some way the run underway which will compose the first part of all data set names, four characters or less. He must pass a character array (four characters or less per data set) containing unique names to form the second part of the data set names desired. The program will then open the library, write headers for each data set, and store the contents from a real user array. Subsequent references for writing data to the data sets is done by specifying the index to the table, which is the same as the index to the array of names passed.

Data is retrieved by specifying the two data set names and the user array to receive the data.

INPUTS --

LIB - integer referring to the library. Should be one greater than the number of libraries already in use. This number is almost always set to unity.

VNAME - Name of VAX (machine dependent) file where data is to reside.

FNAME - First part of data set name common to all data sets.

NMS - Number of data sets desired.

NAMES - Character array (4 character words) of length at least NMS containing a unique list of names forming the second part of the data set name. data set names will look like

FNAME.DNAMES(1),... where of course the contents of FNAME and DNAMES are used.

NSZE - Max length of a record expected for largest write operation. This variable is used only for error checking. Set to size of DIMENSION of array being written.

The output is a set of FORTRAN data set sequence numbers stored in ITBL. (Placed in common to insure data is preserved during run)

Second function--WRITE data sets (STAGS does this for you).

Entry WDATA is called each time a record is to be written on a given data set. The number of words per record must be the same for all records, and now a max of 500 words and 10 records are allowed. The user must provide a single precision array of real data which are to be written on to a specified data set. Upon exit these numbers will be stored in the data set.

INPUTS:

INDX - Data set index. This is the index of the data set name in the list DNAMES above.

NPTS - Number of data points to be written.

DATA - Array of data points.

Third function--CLOSE out library (STAGS does this)

The single statement CALL CDATA closes out library after last write operation. Must be called only once.

Fourth function--REOPEN data file (user must do this subsequent to STAGS run)

After file has been closed and data needs to be read back, a file is reopened by a call

CALL REOPEN(LIB,VNAME)

where VNAME is the same as in the call to PDATA (OPEN), and
LIB is an associated unit number (usually set to unity).
(For CDC, STAGS has chosen VNAME to be STGPLT, local file, CDC,
or STAGS.PLT, VAX-VMS permanent file.)

Fifth function--READ data set by data set name.

Previously-stored data is brought back by referencing each set
of data by the two names as specified in the call to PDATA.
The proper number of user data are then returned to the user
workspace, provided the data set has been written. If the data
cannot be found, an error flag is set for user disposition.
The entry point name is RDATA:

```
CALL RDATA (N1,N2,DATA,ISTAT)
```

INPUTS:

N1 -- (Character) First part of data set name (4 characters or less)

N2 -- (Character) Second part of data set name (4 chrs. or less)

DATA -- User array to receive data.

OUTPUT:

DATA -- Contents of user array DATA are replaced with contents of data set {N1,N2}.

ISTAT -- Status variable: if ISTAT>0, data set was found, and ISTAT words (single precision) were returned into DATA. If ISTAT = 0, data set name was found, but no data was written. If ISTAT<0, no data set with the name pair {N1,N2} was found.

For the STAGS histories here, the first part of the name is taken from the case name, and the second name is LOAD, DIS1, DIS2, etc. An example: Suppose case name is PRESSURE VESSEL TEST, and the user wants the third displacement. The call to RDATA will then be CALL RDATA ('PRES','DIS3',ARRAY,N) with N values of data being sent (single precision real) to ARRAY. It is the responsibility of the user to process the data in ARRAY for plotting.

EXAMPLE RUN (VAX-VMS)

Command file:

```
!
!  (regular stags2 command procedure)
!
$ASSIGN CASE.IN2 FOR075
$RUN STAGS2.EXE
```

Contents of file CASE.IN2

```
PRESSURE VESSEL TEST
4
1 1 1 1  $ U-displacement
1 5 1 3  $ W-displacement
1 5 2 3  $ Adjacent W displacement
1 5 3 4  $ Ru at column 4
```

Example subroutine

```
      Subroutine GETDATA
      DIMENSION DDATA(30),LOADS(30)
      REAL DDATA,LOADS
      CALL REOPEN(1,'STAGS.PLT')
      CALL RDATA('PRES','DIS1',DDATA,N)
      CALL RDATA('PRES','LOAD',LOADS,N)
      IF(N.LE.0) CALL ERROR
C
C     CALL TO SOME USER PLOT ROUTINE, CALLED "PLOT" HERE
C
      CALL PLOT('First load case',DDATA,'Displacements (in)',
     &LOAD,'Load Factor, p/E',N)
C
C     Repeat above sequence for other displacements, etc.
C
      .
      .
      RETURN
      END
```

EXAMPLE RUN (CDC)

Runstream:

```
Comment.
Comment. Regular STAGS runstream up to STAGS2. card image.
Comment.
DEFINE,STGPLT=PLOTDATA. (Any name will do.)
STAGS2.
**EOR**  (End-of-record card image)
This is the input to STAGS1
.
.
. and so on...
**EOR**
PRESSURE VESSEL TEST
4
1 1 1 1  $ U-displacement
1 5 1 3  $ W-displacement
1 5 2 3  $ Adjacent W displacement
1 5 3 4  $ Ru at column 4
**EOF**  (End-of-file card image)


SAMPLE SUBROUTINE EXECUTION

FTN5,B=LGO.
LDSET,PRESET=ZERO,LIB=LIB2/LIBU. (STAGS2 and STAGSU libraries)
NOGO,RUNIT
ATTACH,STGPLT=PLOTDATA.
RUNIT.
**EOR**
      PROGRAM MAIN (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
C     any user operations, including plot initialization,etc.
C
      CALL GETDATA
      STOP
      END
      Subroutine GETDATA
      DIMENSION DDATA(30),LOADS(30)
      REAL DDATA,LOADS
      CALL REOPEN(1,'STGPLT')
      CALL RDATA('PRES','DIS1',DDATA,N)
      CALL RDATA('PRES','LOAD',LOADS,N)
      IF(N.LE.0) CALL ERROR
C
C     CALL TO SOME USER PLOT ROUTINE, CALLED "PLOT" HERE
C
      CALL PLOT('First load case',DDATA,'Displacements (in)',
     &LOAD,'Load Factor, p/E',N)
C
```

```
C     Repeat above sequence for other displacements, etc.
C
      .
      .
      RETURN
      END
**EOR**
any inputs to user program
**EOF**
```

Standard Bibliographic Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-4000		

4. Title and Subtitle	5. Report Date
Enhancements to the STAGS Computer Code	November 1986
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
C. C. Rankin, P. Stehlin, and F. A. Brogan	LMSC-D060755
9. Performing Organization Name and Address	10. Work Unit No.
Lockheed Missiles & Space Company, Inc. Palo Alto Research Laboratory 3251 Hanover Street Palo Alto, CA 94304	11. Contract or Grant No. NAS1-16723
12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
National Aeronautics and Space Administration Washington, DC 20546	Contractor Report
	14. Sponsoring Agency Code 505-63-31-01

15. Supplementary Notes

Langley Technical Monitor: Gaylen A. Thurston
Final Report

16. Abstract

The research conducted during the last 3 years has greatly enhanced the power of the STAGS family of programs. Members of this family include STAGS-C1 [1] and RRSYS [2].

As a result of improvements implemented during these last 3 years, it is now possible to address the full collapse of a structural system, up to and beyond critical points where its resistance to the applied loads vanishes or suddenly changes. This also includes the important class of problems where a multiplicity of solutions exists at a given point (bifurcation), and where until now no solution could be obtained along any alternate (secondary) load path with any standard production finite-element code.

17. Key Words (Suggested by Authors(s))	18. Distribution Statement
Shell structures Postbuckling Collapse Bifurcation Corotational formulation	Unclassified - Unlimited Subject Category 39

19. Security Classif.(of this report)	20. Security Classif.(of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	64	A04

For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1986