STATIC AEROELASTIC ANALYSIS OF FLEXIBLE WINGS VIA
NASTRAN PART I(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING K JONES
UNCLASSIFIED DEC 82 AFIT/GAE/AA/82D-16-PT-1
STATIC AEROELASTIC ANALYSIS
OF
FLEXIBLE WINGS
VIA NASTRAN

THESIS

AFIT/GAE/AA/82D-16

Kim Jones
1Lt USAr
STATIC AEROELASTIC ANALYSIS
OF
FLEXIBLE WINGS
VIA NASTRAN

THESIS

AFIT/GAE/AA/82D-16
Kim Jones
1Lt USAF

Approved for public release; distribution unlimited
STATIC AEROELASTIC ANALYSIS
OF
FLEXIBLE WINGS
VIA NASTRAN
PART I

THESIS
Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

By
Kim Jones, B.S.A.E.
1Lt USAF
Graduate Aerospace Engineering
December 1982

Approved for Public Release; Distribution Unlimited
Preface

This thesis is a continuation of the research done by Captain Lance P. Chrisinger in his M.S. Thesis at AFIT in 1980. The first part contains the theoretical development of the sequence along with the results of a survey of wing models. The survey determined the characteristics of average wing models that the procedure must be able to tolerate. The second part develops a procedure for designing and testing a NASTRAN module. Then the module used for the aeroelastic sequence, GIAS, is used as an example. The logical procedures in GIAS are then explained. Finally, a detailed description of the aeroelastic sequence is given along with a description of how to implement it.

I would like to extend my gratitude to Captain Hugh C. Briggs, who has understood and put up with me while guiding me through this thesis.

I would also like to thank Michael Bernier and Jim Trace for their help with the computer.

A very special thanks goes to my wife, Valerie, whose patience and understanding has helped greatly with my stay at AFIT.
Contents

Part I.

Preface .................................................. ii
List of Figures ........................................... iv
Abstract .................................................. v
I. Introduction ............................................ 1
II. Background ............................................. 2
   Dual-Program Method ........................................ 2
   FLEXLOADS .................................................. 3
   FLEXSTAB .................................................. 3
   Via NASTRAN ............................................... 4
III. Theory ............................................... 5
   Iterative .................................................. 5
   Noniterative .............................................. 7
   Comparison ............................................... 9
IV. Capabilities .......................................... 11
V. Procedure ............................................. 13
VI. Survey ................................................ 14
   Characteristics to be Determined ......................... 14
   Characteristics of Wings in Survey ....................... 16
   Effects of Characteristics ............................... 25
Bibliography ............................................. 29

Part II.

I. Introduction .......................................... 1
II. Designing and Testing a NASTRAN Module ............. 2
III. Example of a New NASTRAN Module, GIAS ............. 9
IV. Logical Procedures Within GIAS ........................ 18
V. Running the Flexible Wing Solution Sequence .......... 22
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Part I.</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Graphical Description of T-37 Wing Model</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>Graphical Description of T-38 Wing Model</td>
<td>19</td>
</tr>
<tr>
<td>3a</td>
<td>Graphical Description of &quot;Eglin Wing&quot; Model</td>
<td>21</td>
</tr>
<tr>
<td>3b</td>
<td>Graphical Description of &quot;Eglin Wing&quot; Model (Wing Tip)</td>
<td>22</td>
</tr>
<tr>
<td>4a</td>
<td>Graphical Description of B-1 Wing Model (Inboard Substructure)</td>
<td>23</td>
</tr>
<tr>
<td>4b</td>
<td>Graphical Description of B-1 Wing Model (Outboard Substructure)</td>
<td>24</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Figure</th>
<th>Part II.</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Link 11</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>Determining AOAP</td>
<td>24</td>
</tr>
</tbody>
</table>
The purpose of this study was to expand the capabilities of the Static Flexible Wing Aeroelastic Sequence that Captain Lance P. Chrisinger developed for NASTRAN as his Master's Thesis at AFIT. Captain Chrisinger developed a basic procedure to enable NASTRAN to analyze flexible wing airloads and stresses. That capability is expanded to enable analysis of standard wing models.

A subroutine was incorporated into NASTRAN to eliminate extensive hand-calculation of transformation matrices.

The capability to tolerate control surfaces was discovered. Also, a survey of wing models in the Air Force inventory was taken to determine the characteristics of the average wing model. This was used to determine where the aeroelastic procedure was lacking in its ability to analyze wing models.
STATIC
AEROELASTIC ANALYSIS
OF
FLEXIBLE WINGS
VIA NASTRAN
PART I

I. Introduction

An improved aeroelastic package has been developed to calculate stresses, displacements, and aerodynamic coefficients for flexible wings undergoing computer-generated airloads. The package is based on the readily available NASTRAN computer program and can tolerate moderately complex wing models with secondary structure. The need for such a package became evident as aircraft wings were designed with more structural flexibility. The changes in displacements caused by the airloads, and the changes in airloads caused by the displacements became significant enough to warrant attention. As a result, several aeroelastic analysis packages have been designed, such as FLEXLOADS and FLEXSTAB. However, all of the packages designed so far contain deficiencies that prevent them from efficiently solving complex wing structures. This package is an attempt to design an aeroelastic analysis program that does not have any serious drawbacks.
II. Background

The aeroelastic flexible wing problem is a study in fluid-structure interaction because the airloads on the structure are dependent on the displacements of the structure. Several packages have been designed to solve this interaction problem for aircraft wings.

Dual-Program Method

One type of solution technique uses an airloads generator computer program and a structural analysis package. The advantage of this method is that its parts are already in existence. However, there are some serious drawbacks to this solution technique. Much time is consumed in transforming the generated airloads into forces for the structural analysis package. Also, the displacements from the structural analysis package must be transformed into displacements for the airload generator. An additional problem with this method is the duplicate computer work required. As iterative passes are made through the airloads generator and structural analysis package the same equations are assembled and decomposed. This results in the same work being done with each pass through the system.
FLEXLOADS

Another solution technique aimed at solving the flexible wing problem involves a single program using an iterative solution. The program, FLEXLOADS, was developed by General Dynamics where it is currently being used to analyze several bombers and transports. They are also modifying FLEXLOADS to use with nonlinear aero theory in order to investigate the separation and stall flight regimes. Northrup is using FLEXLOADS in their analysis of several fighters. The advantages of FLEXLOADS are that it is a relatively small, easily manageable program and that it contains several different aero theories to choose from. A disadvantage is its size restriction of 165 degrees-of-freedom (DOF), which will limit the complexity of the models that can be analyzed. An attempt is being made at San Antonio Air Logistics Center (SAALC) to use Guyan Reduction to reduce the number of DOF of a T-37 wing model from 850 to 165 DOF and still keep its accuracy. They are currently having difficulty finding a representative set of nodes that leaves the natural frequencies and static displacements unaltered.

FLEXSTAB

Still another way to solve the flexible wing problem is to use FLEXSTAB, a computer program developed by Boeing. FLEXSTAB was originally designed as a stability analysis package but can also analyze structures. It is currently being used at Nasa-Dryden and Nasa-Wright. The primary
disadvantage of FLEXSTAB is that it gives only approximations of the stresses and displacements because it uses equivalent beam elements rather than the detailed model.

**Via NASTRAN**

Captain Lance Chrisinger, in his Master's Thesis at AFIT, had previously done all the preliminary work of comparing these various solution techniques (Ref 3). The problems a solution technique must be able to tolerate in order to effectively solve a flexible wing are: a wide variety of element types, any odd characteristics of typical wing models such as node number resequencing (SEQGP) cards, and large numbers of elements. After looking at these requirements and the characteristics of all the available solution techniques it became apparent that there is a need for yet another solution technique. The solution technique must involve a complete, self-contained computer program. All of these qualities were found in the NASTRAN computer program. The one disadvantage that this program does have is that it is rather large and unwieldy. However, with the recent inclusion of an airload generator, NASTRAN has all the requirements to effectively solve a flexible wing. In addition, Captain Chrisinger built an instruction sequence for solving the flexible wing problem for a simple generic wing box (Ref 3).
III. Theory

There are two approaches to solving flexible wings. Both are based on the same assumptions but differ in methodology. In one scheme iterative passes are made through a sequence of steps while in the other the process is assembled into a single equation to be solved (Ref 2).

Iterative

The iterative solution consists of three parts: a pre-processor to set up the operations, an iterative loop that terminates when a specified tolerance is reached, and a post-processor to recover the output.

Definitions

* - general matrix multiplier

\( U_s \) - the change in structural deformation

(\( U_s \) is the first value is the initial structural displacements representing the initial angle-of-attack)

\( U_{st} \) - The cumulative structural deformation

\( Q_s \) - The change in forces on the structural grid

\( Q_{st} \) - The cumulative forces on the structural grid

\( Q_A \) - The changes in airloads on the aero grid

\( Q_{AT} \) - The cumulative airloads on the aero grid
Procedure

set \( U_s \) = Angle of Attack
set \( U_T = - \) Angle of Attack

initial angle of attack so the initial angle of attack will not be included in summing the deformations

set \( Q_s = 0 \)
set \( Q_T = 0 \)

top of loop

\[ U_A = G \times U_s \]

\[ Q_A = A^{-1} \times U_A \]
\[ Q_s = G' \times Q_A \]

\[ U_s = K^{-1} \times Q_s \]
\[ U_T = U_T + U_s \]

G - transformation matrix from structural displacements to aero displacements

\( A^{-1} \) - matrix of aero coefficients

\( G' \) - transformation matrix from airloads on aero grid to forces on structural grid

\( K \) - stiffness matrix

cumulating structural deformation changes
\[ Q_{ST} = Q_{ST} + Q_s \]

cumulating changes in forces on structural grid

\[ Q_{AT} = Q_{AT} + Q_A \]

cumulating changes in airloads on aero grid

go to top of loop unless the norm of the change in structural displacements is less than the tolerance

output

\[ U_{ST} \]

\[ Q_{ST} \]

\[ Q_{AT} \]

aerodynamic coefficients

Noniterative

The noniterative scheme uses the same assumptions but it is ordered differently (Ref 2):

Definitions

* - general matrix multiplier

K - the stiffness matrix

\[ U_s \] - structural deformations

\[ Q_s \] - structural forces

G - the transformation from aero forces to structural forces and also from structural deformations to aero deformations

AOA - initial angle of attack

\[ U_A \] - aero deformations
\( Q_A \) - aero forces
\( CQ_A \) - coefficients of aero forces
\( CP_A \) - coefficients of aero pressures
\( DLJK \) - matrix of aero coefficients, real part of complex aero wash
\( D2JK \) - matrix of aero coefficients, imaginary part of complex aero wash
\( A_{ijr} \) - matrix of aero influence coefficients
\( SKJ \) - matrix of areas of the wing
\( Q \) - dynamic pressure

Procedure

\[
U_A = G^T \cdot U_s
\]

\[
CP_A = A^{-1}_{ijr} \cdot [DLJK + i(D2JK)] \cdot U_A
\]

\[
CQ_A = SKJ \cdot CP_A
\]

\[
Q_s = Q \cdot G \cdot CQ_A
\]

therefore

\[
Q_s = Q \cdot G \cdot SKJ \cdot A^{-1}_{ijr} \cdot [DLJK + i(D2JK)] \cdot G^T \cdot U_s
\]

but

\( D2JK = 0 \)

only concerned with static, zero frequency

and

\( F = G \cdot SKJ \cdot A^{-1}_{ijr} \cdot AOA \) rigid body \( Q_s \) due to AOA

and

\( FDEFG = G \cdot SKJ \cdot A_{ijr}^{-1} \cdot xDLJK \cdot G^T \cdot Q_s \) due to deformations
therefore

\[ Q_s = Q \ast [(F \ast AOA) + (FDEF \ast U_s)] \]

if

\[ U_s = K^{-1} \ast Q_s \]

then

\[ U_s = Q \ast K^{-1} \ast (AOA \ast F) + (FDEF \ast U_s) \]

\[ [1 - (Q \ast K^{-1} \ast FDE) \ast U_s = Q \ast K^{-1} \ast F \ast AOA \]

finally

\[ U_s = \left[1 - (Q \ast K^{-1} \ast FDE)\right]^{-1} \ast Q \ast K \ast F \ast AOA \]

The last equation is the one that is used for the noniterative scheme.

Comparison

There are two points to be considered when deciding which solution scheme to use for a particular problem. Both the accuracy and the expense (in terms of time to compute) is discussed. The accuracy of the iterative solution scheme can, at best, equal the accuracy of the noniterative solution. This is because they both make the same approximations but the iterative solution has an additional inherent error. The iterative solution takes a shorter time to assemble and decompose than the noniterative scheme but then also involves the time in going through the iteration loop. The noniterative scheme takes longer to assemble and decompose but once this is done, the equations are solved only once. Therefore, if the time to assemble and decompose the equations for a particular model in the iterative solution is significantly
shorter than that for the noniterative solution, then the iterative solution will take less time. In order for this to work, the assembly and decomposition time for the iterative solution has to be shorter than the noniterative solution by the amount of time it takes for the iterative solution to complete all the iterative passes. Recent experience indicates typically 3-6 passes are required for simple models. For the present work, the iterative scheme has been employed.
IV. Capabilities

The discussion of the capabilities of the current solution technique is broken into two parts. The first gives an outline of the applicable characteristics of NASTRAN. The second describes the capabilities of the instruction sequence.

As outlined in the background section, the primary disadvantage in using a program such as NASTRAN is its large size. Because of this the instruction sequence is more difficult to manipulate. Another disadvantage is that NASTRAN uses only one aero theory, a Doublet-Lattice Method. This theory does not generate any rotational forces; therefore, any bending elements in the model will not have forces along their rotational DOF due to the airloads. Despite these disadvantages, NASTRAN was chosen as a host program because of its several advantages. The most important of these is NASTRAN's ability to tolerate very large problems efficiently. Another major advantage is the wide variety of element types already available in NASTRAN. In addition, many options to the instruction sequence are easily implemented. These three advantages lend a great deal of flexibility to any instruction
sequence implemented with NASTRAN. This is essential when dealing with the wide range and variety of wing characteristics encountered.

This second section outlines the capabilities of the instruction sequence. The ultimate goal for the sequence is to be able to solve for the stresses, displacements, and aerodynamic coefficients of an entire flexible aircraft. Currently, a restricted type of flexible wing can be solved. Provisions have been made to calculate internally all of the transformation matrices and other derived input data. This enables the user to analyze a wing using an existing model with minimum additional input data. The instruction sequence is capable of solving deflected control surfaces as long as the deflections are small enough to stay within linear aero theory. Substructuring has not been attempted while using the instruction sequence. The only wing surface element type that can be used by the instruction sequence is the CQDMEM2 element. Provisions have yet to be made to tolerate grid point number resequencing (SEQGP) cards. Wing-fuselage aerodynamic interaction can be modeled by constructing an aero model of the fuselage near the wing root.
V. Procedure

This section is included in the event that someone with an understanding of the instruction sequence has a desire to use it. Within this section is an overview of additional input data that must be included with a standard model in NASTRAN format. The details of each additional piece of data are covered in Part II.

The instruction sequence requires that two subcases be run. The first subcase generates the initial angle of attack and the second subcase actually goes through the iterations. There are several PARAM cards that must be added. These are concerned with various dimensions of the wing model, the angle of attack, and the error deemed adequate for the iteration convergence. SPC cards are required for both subcases to help determine the AOA and to partition out the rotational DOF of the model. ASET cards are needed to narrow down the analysis set to just the Z DOF of every free node. A CAERO card and its support cards are needed to determine the flight regime. An EIGR card is needed to help set the initial angle of attack. SPLINE and SET cards are needed to build the transformation matrix from the displacements on the structural grid to the displacements and slopes on the aero grid. Finally, a table (TOSEL - Top Surface Element List) must be input in the Bulk Data Deck.
VI. Survey

In order for the instruction sequence to run efficiently with standard wing models their characteristics must be known. To this effect a survey of several wing models in the Air Force inventory was taken. The answers to a list of questions were compiled to determine the characteristics of the average wing model. Though several wings were surveyed, only four will be presented here. These depict typical wing models. The rest are not presented because of repetition.

Characteristics to be Determined

The characteristics investigated were:

1. The number of grid points and DOF. This is an estimate of the size of a typical wing model.

2. Whether the model has leading and/or trailing edge secondary structure. Since the aero and structural models should both have the same planforms structural models without the secondary structure will require modification.

3. If the model has been validated with only the Z-DOF in the analysis set. The model can be validated either by known static deformations or by known natural frequencies. This should have
been done because the current NASTRAN aero theories allow only the Z-DOF in the analysis set (Ref 1).

4. The existence of any stresses, displacements, basic aerodynamic coefficients, and distributions for flight test data and standard computer analysis. This provides a measurement standard for judging the accuracy of results. Computed versions of these items can be generated by the solution sequence to provide accuracy checks at various points in the sequence. The data from a standard computer analysis could be used to check the sequence before it enters the iterative loop. The flight test data provides a check for the final results from the solution sequence.

5. Types of area elements on the wing surface. Currently, only certain element types on the wing surface can be analyzed by the solution sequence.

6. If there are any bending elements in the model. This will affect the validation described in question 3 since the analysis set for the solution sequence is condensed to only translational DOF by Guyan reduction.

7. If there are any SEQGP cards. This would affect the internal equation numbering which in turn affects the calculation of some of the transformation matrices (Ref 1).
8. If the wing model includes wheel wells or other cutouts. This will become significant when aero theories are implemented that panel 3-D structures. The current theory uses 2-D lifting surfaces and is only affected by the curvature of mean camber surface (Ref 1).

9. Is it in NASTRAN format. Considerable time and money can be spent in reformatting data to NASTRAN format.

Characteristics of Wings in Survey

Several wing models were evaluated with respect to the criteria. These models are T-37, T-38, B-1, C-5, and KC-135. Also, a wing model termed the "Eglin Wing" was included in the survey. The four wings presented in detail are the T-37 wing, the T-38 wing, the "Eglin Wing," and the B-1 wing.

The T-37 is illustrated here because it is a typical early subsonic wing type (Fig 1). It only has two spars and a thin skin. Unlike modern fighters, the T-37 wing has a high aspect ratio and a complex wing carry-through connected to the fuselage. This carry-through will present some problems in determining the wing root boundary conditions for this type of wing. The T-37 wing model has 319 grid points with 850 DOF. The model has no leading or trailing edges and, therefore, no control surfaces. This particular wing model has been validated in the Z-DOF only. This is unusual, typically wing models contain rotational
DOF. The T-37 wing model is validated under such conditions due to the Guyan Reduction efforts being done at SAALC on the T-37 wing model for the FLEXLOADS program. There is very little flight test data because the T-37 was built when extensive tests on new aircraft were not done. There is some aerodynamic computer analysis for the T-37 wing, primarily by the airloads program called USSAERO. Several types of area elements are used on the wing surface. There are very few bending elements encountered in the T-37 wing model, they are only in the wheel well. The T-37 wing model contains SEQGP cards because the grid numbers are encoded position coordinates. Finally, the T-37 wing model is already in NASTRAN format.

The T-38 wing is a typical modern fighter-type wing having several spars, thick skin, a low aspect ratio and 388 grid points with 1564 DOF (Fig 2). It does have a leading edge structure but no trailing edge because the control surfaces are typically analyzed separately from the wing. The model has not been validated using only the Z-DOF. There are large amounts of flight test data and computer analyses to provide check data for the solution sequence. There are four types of area elements on the wing surface, some of which are plates, and there are additional bending elements within the wing. There are SEQGP cards included in the model. Wheel wells are present in the model but the wheel covers have not been modeled.
Fig 2. Graphical Description of T-38 Wing Model
Analyzing the wing is simplified by the fact that it is in NASTRAN format and has been analyzed in several static load conditions.

The Eglin Wing is presented because it is a typical damage tolerance wing model (Fig 3). These tend to be more complex because they have to closely trace the stresses throughout the wing. As a result, the Eglin Wing has 542 grid points with 2710 DOF. The model has both leading and trailing edges, the trailing edge consists partly of the control surfaces. It has not been validated with only the Z-DOF. The check data available is a minimum with only a small amount of computer analysis provided. There are several types of area elements on the wing surface in both triangular and quadrilateral shapes. The Eglin Wing does have bending elements. A large number of SEQGP cards are present which will affect the transformation matrices. The model also contains wheel well cutouts whose covers will have to be modeled if a 3-D aero theory is implemented. As with the T-38 wing model, the Eglin Wing is also in NASTRAN format and has been run in several static load cases.

Despite the effects of the wing pivot area, the B-1 wing model is taken as a typical "heavy-aircraft" type wing. These effects do not interfere with the present analysis so swing wing aircraft can also be analyzed. The outboard sections of the wing are fairly typical to other "heavy-aircraft" wings. Because the wing is large and complex, there are a great number of grid points and DOF, 5116 grid points with 24,361 DOF (Fig 4). Fortunately,
Fig 4b. Graphical Description of B-1 Wing Model (Outboard Substructure)
wings of this size can be analyzed using substructuring. The largest substructure in the B-1 wing model has 1605 grid points and 5479 DOF. This is still large but it is much more manageable than the entire wing. All parts of the wing are represented by the substructuring, including the leading and trailing edges, with the control surfaces. The B-1 wing model has not been validated using only the Z-DOF. There has been a lot of flight test data and computer analysis done on the B-1 wing. There are four types of area elements on the wing surface. The model includes bending elements, but no wheel wells or other cutouts. Since the B-1 model is not in NASTRAN format, SEQGP cards are not applicable.

Effects of Characteristics

Considering the wide range of characteristics of the wings in the survey, it is evident that compromises must be made. While the overriding concern in designing the solution sequence is to make it user-oriented, it is not possible to design the sequence for every characteristic. At times certain models will have to be altered to fit the sequence. The average size of wing models is fairly large and requires the sequence to use computer space efficiently. However, after the usage has been trimmed down as far as possible, there will still be some wing models that will be too large for the computer used. In these cases, either substructuring should be used or the complexity of the model lessened by Guyan Reduction.
If the structural wing model does not have leading and/or trailing edges, then two courses of action can be taken. One method is to alter the solution sequence to artificially transfer the loads from the leading and trailing edges to the structural wing box. However, this requires large amounts of the user's time. The second course of action would be to put leading edges on the wing model. This is the easier of the two in that, once done, it automates the load transfer from the edges to the wing box.

Since most wing models are not validated in the Z-DOF, there will be some error in using these models. The error can be measured by restricting the model to the Z-DOF, performing the validation procedures, and calculating how far off the model with only Z-DOF is from the actual structure. If the error determined is unacceptable, the model should be adjusted to give accurate values in the Z-DOF. Another course of action can be taken to reduce the error but it requires extensive rebuilding of the solution sequence. The only reason the models put through the solution sequence are reduced to Z-DOF is the aero analysis within the sequence cannot tolerate any displacements except out-of-plane translations. If desired, a different aero theory can be implemented into the solution sequence but it would require many months of effort to rebuild and debug the altered solution sequence.

Most wing models only use three or four area element
types on the wing surface. Given these element types, no problem is encountered with the solution sequence. However, if an element type different from those normally used is put on the wing surface then problems can arise. It is a simple matter to add that element type to the capabilities of the solution sequence, if sufficient knowledge of how the solution sequence works is available. Lacking that, it is advisable not to include the new element type in the wing surface.

Virtually every wing model contains bending elements. As with the validation of the models in the Z-DOF, the restriction of using bending elements in the solution sequence lies within the aero model. The bending elements will not be loaded by the aero forces in the rotational DOF. If the error incurred is unacceptable the recommended solutions are the same, either replace the bending elements or implement a different aero theory.

Most of the large wing models contain SEQGP cards in an effort to improve the bandwidth of the matrices involved. Currently there is no capability to tolerate SEQGP cards in the solution sequence so care should be taken in choosing the form of the input data for the wing model.

Most wing models contain wheel wells or other cutouts. This is not a problem with the present aero theory but it will need to be considered if a different aero theory is implemented. The most obvious solution is to include the wheel covers in the model. This will allow the wheel well
doors to be included as part of the lower wing surface.

A problem not previously mentioned is the aerodynamic interaction between the wing and other aircraft structures. If the wing is modeled alone then these effects are not taken into account. The aircraft structure that most commonly interacts with the wing is the fuselage. If the fuselage tends to be flat and vertical at the wing root, then the centerline symmetry condition of the wing alone will be a reasonable approximation. However, if the fuselage-wing root structure is rounded and smooth then there can be a great deal of crossflow. Other structures, such as engine inlets, engine exhaust, and wing mounted stores, can also affect the airflow across the wing. Wing-fuselage aerodynamic interaction can be modelled by constructing an aero model of the fuselage near the wing root.

The solution sequence can tolerate most of the characteristics of the average wing model. However, further revisions to the sequence would broaden its capabilities and applicability.
Bibliography


STATIC AEROELASTIC ANALYSIS

OF

FLEXIBLE WINGS

VIA NASTRAN

PART II

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

By

Kim Jones, B.S.A.E.

1Lt USAF

Graduate Aerospace Engineering

December 1982

Approved for Public Release; Distribution Unlimited
# Contents

## Part I.

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>II. Background</td>
<td>2</td>
</tr>
<tr>
<td>III. Theory</td>
<td>5</td>
</tr>
<tr>
<td>IV. Capabilities</td>
<td>11</td>
</tr>
<tr>
<td>V. Procedure</td>
<td>13</td>
</tr>
<tr>
<td>VI. Survey</td>
<td>14</td>
</tr>
</tbody>
</table>

## Part II.

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>I. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>II. Designing and Testing a NASTRAN Module</td>
<td>2</td>
</tr>
<tr>
<td>III. Example of a New NASTRAN Module, GIAS</td>
<td>9</td>
</tr>
<tr>
<td>IV. Logical Procedures Within GIAS</td>
<td>18</td>
</tr>
<tr>
<td>Output Datablocks</td>
<td>18</td>
</tr>
<tr>
<td>Logical Procedures</td>
<td>19</td>
</tr>
<tr>
<td>V. Running the Flexible Wing Solution Sequence</td>
<td>22</td>
</tr>
<tr>
<td>Logical Procedures</td>
<td>22</td>
</tr>
<tr>
<td>Implementation</td>
<td>26</td>
</tr>
<tr>
<td>Bibliography</td>
<td>31</td>
</tr>
<tr>
<td>Vita</td>
<td>32</td>
</tr>
<tr>
<td>Appendix A: GIAS and Environment</td>
<td>33</td>
</tr>
<tr>
<td>Appendix B: Sample Input Data and Illustration</td>
<td>79</td>
</tr>
</tbody>
</table>
Abstract

This thesis is a continuation of the research done by Captain Lance P. Chrisinger in his M.S. Thesis at AFIT in 1980. Captain Chrisinger developed a basic procedure to enable NASTRAN to analyze flexible wing airloads and stresses. That capability is expanded in this report to enable analysis of standard wing models.

A subroutine was incorporated into NASTRAN to eliminate extensive hand-calculation of transformation matrices.

The capability to tolerate control surfaces was discovered. Also, a survey of wing models in the Air Force inventory was taken to determine the characteristics of the average wing model. This was used to determine where the aeroelastic procedure was lacking in its ability to analyze wing models.
I. Introduction

This part presents a detailed account of the procedures associated with the NASTRAN instruction sequence for flexible wings. However, the presentation presumes a level of knowledge about NASTRAN and its organization. In order to fully understand the majority of this part, the reader must understand the organization of NASTRAN, to include datablocks, modules, links, and overlays. The reader should have taken, or have notes of, a course dealing with NASTRAN System Programming (Refs 3,4).

This part is organized into three sections. The first two describe a recommended procedure for designing and testing a module to be put inside NASTRAN. This will also include, as an example, the module designed to work within the NASTRAN instruction sequence to solve flexible wing problems. These two sections were included to illustrate a set of steps not given elsewhere on designing and testing a NASTRAN module. The third section in this part is a detailed account of how to run this instruction sequence, with an explanation of additions to the Bulk Data Deck.
II. Designing and Testing a NASTRAN Module

The following is a recommended procedure for designing and testing new NASTRAN modules. The procedure carries through from the initial conception of the need for a new module to the final testing of the module once it is inside NASTRAN. The procedure pulls together information from various NASTRAN publications to present a complete, logical sequence of steps to design and implement a module. These steps are designed to insure the most error free module in the least amount of time. These will minimize the debugging required while manipulating the entire NASTRAN package. To do this the module must be tested in an environment as close to NASTRAN as possible before the module is entered into NASTRAN. The recommended steps in designing and testing a NASTRAN module are:

1. Recognition of the Need. In designing a new DMAP sequence or running an extensive DMAP alter the user might recognize that none of the DMAP statements, nor any combination of them, provide him with a procedure he needs. The first thing he must do before considering adding a module is be sure that there is no other way to accomplish the procedure, even by a round-about method. The time and effort required to
design, test, and implement a new NASTRAN module is enough to make it a last resort. If it is at all reasonable to operate without the module then that course of action is recommended. In considering a DMAP to solve his problem the user might compare MSC NASTRAN to COSMIC NASTRAN. The MSC version contains many more modules and might have the proper DMAP modules where the COSMIC version did not.

2. Development of the logic process and recognition of required inputs. These two steps are lumped together because the logic processes inside the module must operate within the NASTRAN environment. Some of the inputs that would be ideal for the procedure might just not be there, or might exist but in a format that will increase the complexity of the module's logic. To determine exactly what is needed by the sequence from the module, and also to check the sequence, run the sequence with the module and its outputs replaced by hand-generated data.

3. Determine where to get the inputs.
There are several ways that inputs are available to a module. They can either be: an existing datablock, a datablock that can be built in the
DMAP before the module is executed using DMAP alters or the data that can be user supplied through PARAM, DTI, or DMI cards. Datablocks can be either matrices or tables. The way the data reaches the module will have an effect on its logic processes and on the amount of user input needed. It is desirable to require as little as possible from the user. This will reduce input errors.

4. Write the module with all of its NASTRAN-necessary parts. NASTRAN uses the Fortran IV computer language. Any limitations with this language must be taken into account when writing the module. The module will be operating in an environment with common statements, open core arrangements, and various other subroutines to which it has access. Depending on the design of the module, the use of these will have to be carefully planned out so as to minimize the module's use of computer time and space. This must be considered because the large variation of problem types encountered will cause the solution sequence to tax the capabilities of most any computer.

5. Build a file of the data the module will receive. This will be the input to a simulated environment built to debug the module as thoroughly as
possible before it is put inside NASTRAN. The content of the data should be from several sample cases to which the correct (or at least expected) answers are known. The form of the data can be up to the user but it should contain all the information that will reach the module, not just what the module will use. The procedures to develop this file are determined by how the module will get the data when it is incorporated into NASTRAN. To get the data that the user will supply to the module when it is incorporated into NASTRAN is relatively simple. Just put the data on the input file as it would appear in the Bulk Data Deck. The other data, those that are generated inside NASTRAN before the module, will require a different approach. A solution sequence nearly identical to the one that will use the module should be run. This solution sequence should not have the call to the module being developed. It is not necessary for the sequence to run until completion. All that is required is to derive the data that the module will use. This data should be punched out of a NASTRAN run onto a file. It is advisable to do this right after the last DMAP statement before the new module is called. It is easier to do this with a DMAP alter. The specific cards that
need to be inserted in the DMAP depends on the form of the data. If the data is in table format use the TABPCH module. If it is in matrix format use MATPUN.

6. Build environment around the module to simulate the NASTRAN environment. Ideally, this should be done with as few changes to the module as possible. The form of the module should be as close as possible to the form it will have when inside NASTRAN. The environment should consist of several subroutines, including the module, with a master sequence of commands whose job it will be to call an initialization subroutine and then the module. The module, in turn, will call utility subroutines. The environment should simulate all the common statements, open core, functions, and subroutines that the module will see while inside NASTRAN. The environment subroutines need not do exactly what the equivalent NASTRAN subroutines do, but they must return the same response to the module. Some of the subroutines the module calls may have to be copied from NASTRAN source code and put into the environment rather than just building similar ones. However, caution should be used in doing this because the subroutines pulled from NASTRAN will
have to have their NASTRAN environments simulated also. Their environments can be as simple as another utility subroutine or they can require an assortment of common statements, open core, functions, and subroutines of their own.

7. Run several test cases to debug the module in the environment. The test cases should use the input data described in Step 5. It is helpful here to put internal check statements within the environment. Perhaps printing out a simple statement telling which subroutine the program is in. The module must be debugged until the environment's output data is exactly what the user wants NASTRAN to see once it finishes executing the module.

8. Put the module inside NASTRAN.

The procedure to do this can be found in the NASTRAN Programmer's Manual, in most any other NASTRAN system programming text or from someone who has done it before and is knowledgeable about the NASTRAN system. The module must be put in a link that will allow it access to the subroutines and common areas the module needs.

9. Run additional tests on installed module until satisfied. These should begin with the same test cases run before, in Step 7. However, it might be a good idea to run additional cases. It could be helpful to have various matrices and tables
printed out at key locations within the DMAP. Of course, to run these cases, the trial DMAP should have been altered to include the call for the module.
III. Example of a New NASTRAN Module, GIAS

This section uses the NASTRAN module GIAS as an example of the procedure to design a module.

GIAS was designed to build various matrices used to solve the flexible wing problem. Beforehand, the matrices were manually calculated and entered into the program via DMI cards. The overriding concept in GIAS is to minimize the work the user must do. As many matrices as possible are built within GIAS. This leaves only one matrix, other than the normal model data, to be built by hand. As development on GIAS continues, this matrix will also be built inside NASTRAN.

The following is a reiteration of the steps necessary to design a module for NASTRAN, with specific application to GIAS.

1. Recognition of need. The solution sequence for solving flexible wings requires several transformation matrices and moment arm vectors. The procedure used to generate these matrices is much too detailed and complex for NASTRAN to accomplish it with DMAP. Therefore, in order to significantly reduce the amount of data to be entered by the user a new module is needed to calculate the extra data.
Development of logic process, recognition of required inputs and where to get them. Since the object of the module is to reduce the user's work most of the input data to the module already exists in NASTRAN while it is running. At this time there is only one additional table that needs to be input by hand. Those that already exist in NASTRAN are:

- **SIL** - Scalar Index List
- **BGPA** - Basic Grid Point definition table, Aerodynamics
- **GPLA** - Grid Point List, Aerodynamics
- **ACPT** - Aerodynamic Connection and Property Table
- **ECTA** - Element Connection Table, Aerodynamics

Descriptions of the existing datablocks can be found in the NASTRAN programmer's manual.

The user-supplied table is:

- **TOSEL** - Top Surface Element List

TOSEL contains a list of area element numbers on the top surface of the wing. The datablock is needed because the loads generated by the aerodynamic model in the solution sequence are only applied to the nodes of the elements listed in TOSEL. TOSEL contains one record for each type of area element on the upper surface. The records are arranged so that the element type
code numbers are in increasing order. The first entry in each record is the BCD name of that element type. The entries following that, within each record, are the element numbers of that element type on the upper surface in increasing order. TOSEL cannot be derived inside NASTRAN; currently there is no method available to isolate the upper wing surface elements in a general wing model.

4. Write the module with all of its NASTRAN-necessary parts. The entire module is listed in the back of this volume (Appendix A). There are various items that need to be included in the module to allow it to operate within NASTRAN. For GIAS the NASTRAN-necessary parts are:

**COMMON statements**

**COMMON/ A** - Obtains the variable "A" out of the blank common. "A" is the bottom of blank common.

**COMMON/ SYSTEM/ IBUF** - IBUF is the required length of GINO buffers

**COMMON/ PACKX/ TYPIN, TYPOUT, IROW, NROW, INCR** - Various parameters for the PACK subroutine

**COMMON/ ZOPEN/ Z(1)** - Designates a common area for working
DATA statements

DATA NAME/4HGIAS,4HREAD/ - Assigns a BCD value for NAME, a parameter for the MESSAGE subroutine, an error utility

DATA statements are also used to assign inputs the consecutive numbers 101-106. In addition, data statements are used to assign outputs the consecutive numbers 201-209.

Functions

CORSZ(Z,A) - Determines amount of open core available from the field length specified for the program run

Subroutines

GOPEN - Opens a datablock in order to read from it or write into it

CLOSE - Closes a datablock

READ - Reads from a datablock

PACK - Writes to a datablock

RDTRL - Reads the trailer of a datablock

WRTRL - Writes the trailer of a datablock

MESSAGE - Prints selected fatal errors as they occur
SSPLIN - Produces an interpolation matrix

A more detailed description of the subroutines can be found in the NASTRAN Programmer's Manual (Ref 2).

5. Build a computer file of the input data the module will receive. A listing of the input data to the simulated environment for a test case is included in this volume. TABPCH cards were inserted in the solution sequence near line 104; this is where GIAS was later put. Datablocks ACPT, TOSEL, ECTA, GPLA, BGPA, and SIL were output to logical file PUNCH. This file was then manipulated, along with the initialization subroutine (see subroutine INIT, Appendix A), to put these datablocks in a form where the rest of the simulated environment can use them.

6. Simulate the NASTRAN environment. All of the items discussed in Step 4 must be duplicated within the simulated environment. In the simulated environment they do not have to do exactly what the subroutines in NASTRAN do, but they must return the same response to the module being tested. The following are the simulated items in the environment for GIAS; they have the same names as the items in Step 4.
Function CORSZ(Z,A) - Returns a predetermined number to be the size of the Z array.

COMMON/SYSTEM/IBUF - Returns a predetermined arbitrary number to be the size of the buffers, the buffers used in the simulation.

Subroutines

INIT - The initialization subroutine, reads the input file into internal arrays for use by the subroutines READ and RDTRL.

GOPEN - Prints out the statement "Datablock ### opened" to announce the position of the program, but it really does nothing to the datablock.

CLOSE - Prints out the statement "Datablock ### closed" to announce the position of the program, but it really does nothing to the datablock.

READ - Fetches all or part of the datablock contained within a set of arrays prepared by subroutine INIT. It is more complex than an average read statement so it can handle any
errors it encounters, such as being asked to read past an End-of-File or an End-of-Record.

PACK - Prints out the name of the specified datablock along with its values, for inspection but does not create any files.

RDTRL - Reads the trailer of the datablock specified from the set of arrays prepared by INIT.

WRTRL - Prints out the trailer of the datablock specified.

MESSAGE - Prints out the specified error message.

SSPLIN - Produces an interpolation matrix. SSPLIN calls two other utility subroutines, INVERS and GMMATS. These must also be included in the environment. All three of these subroutines had to be copied directly from NASTRAN source code.

7. Run several test cases to debug the module. A simplified wing box was designed and run in NASTRAN using input entered by hand to replace
TRUNK
Contains Modules Such As:
PACK WRTRL (etc.)
READ RDTRL

And Common Areas Such as:
/ /
/ PACKX/
/ SYSTEM/

INVERS
GMMATS

SSPLIN
GIAS /ZOPEN/

Fig 1. Link 11
GIAS. Running several variations of this produced the required check data for the environment.

8. Put the module inside NASTRAN. Because SSPLIN is one of the utility subroutines used, GIAS was put into Link 11 of NASTRAN (Fig 2). The other subroutines needed are also available from this position.

9. Run additional test cases until satisfied. Both the simplified wing box mentioned in Step 7 and an intermediate complexity wing were run as final test cases.
IV. Logical Procedures Within GIAS

It was previously mentioned that GIAS created various transformation matrices and moment arms. This section will look at each of these datablocks and outline the logical procedures used to make them. Several of the procedures overlap within GIAS because several of the output datablocks use the same input.

Output Datablocks

The output datablocks are:

SAS - A matrix containing ones where areas are in the SKJ matrix

GTAK - Surface spline interpolation matrix for transformation of \( \Delta P' \)'s at the aero points to \( \Delta P' \)'s at the structural nodes

TTTTT - Transformation matrix from \( \Delta P' \)'s at the structural points to loads at the structural nodes

VGLS - A vector with ones in the proper positions so that when multiplied by the structural loads it sums them. VGLS also condenses PG down to nonzero terms.

VTMA - Contains the moment arms for the aero-dynamic pitching moment about the Y-axis.
VTRA - Contains the moment arms for the aero-
dynamic rolling moment about the X-axis

VTM - Contains the moment arms for the pitching
moment about the Y-axis due to the trans-
formed structural loads.

VTR - Contains the moment arms for the rolling
moment about the X-axis due to the trans-
formed structural loads.

AIDMT - Unit vector of length equal to the number
of aerodynamic boxes.

Logical Procedures

The first datablocks built in GIAS are TTTTT and VGLS.
Even though they are built at the same time the logic to
build each of them will be considered separately. TTTTT
transforms the \( \Delta \rho \) applied to the centroid of each area
element on the upper surface of the wing model to loads on
the upper surface structural nodes. The first step to do
this is to multiply the \( \Delta \rho \) applied to the element by the
area of the element. Using the geometry of the element the
load over the element is then correctly partitioned out to
each node of the element. The only information needed to
calculate TTTTT is the coordinates of the structural nodes
on the upper surface and which elements they are associated
with. Since VGLS is a list of ones in the Z DOF position
for each node on the upper surface the only information
needed is the upper surface node numbers.
The next datablocks built are VTM and VTR. The moment arms for the structural rolling and pitching moments are simply the coordinates of each upper surface node. This information was obtained when TTTTT and VGLS were built.

VTMA and VTRA are built next. The only information needed for these two is the coordinates of the corners of the aero boxes. From these the coordinates of the 1/2-span 1/4-chord point of each aero box is found. This is the point of each box where the lift is assumed to act. The X and Y values of each of these are the entries in VTMA and VTRA, respectively.

All the information needed to calculate GTAK is now available. In order to calculate GTAK the subroutine SSPLIN needs to know the coordinates of the input and output points and how many of each there are. The input points are the points on the aero boxes where the pressure is assumed to act. The output points are the upper surface structural nodes. Because GTAK has to be built all at once, rather than a column at a time like other matrices, this is the procedure in GIAS that takes up the most computer memory. In fact, depending on the problem, this procedure could be the one that governs the amount of computer memory required for the entire analysis.

The last two datablocks built by GIAS are AIDMT and SAS. To build AIDMT only the number of aero boxes is
needed. SAS has ones down two of its diagonals and zeroes elsewhere. It has as many rows as two times the number of aero boxes and as many columns as the number of aero boxes.
V. Running the Flexible Wing Solution Sequence

In order to best use a solution sequence it is advisable to first understand how it works. Therefore, this section first presents a detailed outline of how the solution sequence for flexible wings operates. Afterward, an explanation of the Bulk Data Deck is presented. An example of the input deck is given in Appendix B.

Logical Procedures

The first part of the solution sequence (from now on, called DMAP) sets up the tables that describe the structural model. These include ECT (Element Connection Table) and GPL (Grid Point List). This set of matrices describes the unconstrained structural model. The constraints of the first subcase allow only rigid body pitch about the wing root trailing edge. This will be found in the eigenvalue analysis and used to set the initial angle of attack. The frequency limits of 0.0 to 1.0 on the EIGR card ensure that this is the only eigenvector used. The DOF are then further partitioned down by Guyan Reduction to only the remaining Z DOF since the aero model can only tolerate displacements in the Z-direction. A real eigenvalue analysis is then performed within a small frequency range around zero. The purpose of the analysis is to obtain the mode shape of the rigid body.
pitch mode. The 2-displacement of the most in-board leading edge node, not including the wing root, is then normalized to one by selection of "point" on the EIGR card. All displacements are multiplied by a user calculated parameter, AOAP, in order to set the initial angle of attack (Fig 1). This procedure determines the zero-camber initial displacements of all the nodes. The indirect method of determining this is unavoidable because COSMIC NASTRAN has no module, or simple sequence of modules, that will do this. Camber is added to the wing, if desired, by using a parameter ICAMB and a matrix CAMBER. After this, various transformation matrices are built. The first one created is GTKA, which is built using the Bulk Data SPLINE cards. This will be used to transform the displacements of the structural model to values of slope on the aero model. These values will then be used by the aero analysis portion to specify the strengths of the downwash vector, DIJK or D2JK. This vector is then multiplied by the aero influence matrix for the Doublet-Lattice aero method, AJJL to determine the $\Delta C_f$ distribution on the aero model.

Following the calculation of GTKA the module GIAS is called to determine other transformation matrices and moment arms. This ends the first subcase of the DMAP.

The second subcase is distinguished from the first by a change of SPC cards. The structural model is reconstrained to allow no rigid body motion. None of the nodes on the wing root have any active DOF; all the other
AOAP = ΔX tan θ

θ = initial AOA

ΔX = X-distance from the leading edge node and the wing root trailing edge

Fig 2. Determining AOAP
nodes are limited to only translational DOF. These nodes are then further reduced to only the Z DOF by ASET cards. This procedure locks the wing root at the initial AOA as determined by the eigenvalue analysis and AOAP. Then the matrices necessary for static deflection analysis are determined for this configuration. This analysis is then performed to determine the structural loads due to the initial AOA.

The next section is the most significant one of the DMAP. The iterative loop calculates the displacements, structural loads, and aero forces due to the structural deformations. Then the structural deformation due to the new airloads are calculated. The loop begins by finding the change in structural deformations due to the airload vector. On the first iteration the load vector is due to the initial AOA. On subsequent iterations it is due to the change in structural deformations. Then the incremental deformations are transformed to slopes on the aero model using the GTAK generated from the Bulk Data SPLINE cards. These slopes are used to determine incremental $\Delta C_{p}$. The incremental $\Delta C_{p}$ are transformed to an incremental load vector which is used in the next iterative pass through the loop. This is accomplished by premultiplying the $\Delta C_{p}$ vector by GTAK and then by TTTTT. At the end of each increment the changes in deformations, structural loads, and aero forces are accumulated. The loop is finally left when the norm of incremental displacements (VNSR) is less
than the user supplied parameter $ELOG$, or a total of ten incremental passes have been made. The loop gives the final values for the structural deformations, structural forces, and $\Delta C_p$ vector.

The DMAP then recovers the complete solution for both the analysis set and the omitted DOF. A complete stress recovery for all the structural elements can be done if requested by the user. The final section of the DMAP determines $C_L$, $C_{\alpha}$, $C_{\alpha \alpha}$ due to both the final aero forces and the final structural loads. Both sets are calculated to provide a means of checking the accuracy of the force transformations between the aero and structural models.

**Implementation**

In addition to the usual model used in structural analysis there are a few more cards, and one additional datablock, needed to run the DMAP. The extra datablock is TOSEL and is entered via DTI cards. TOSEL is the table of area element numbers on the upper wing surface. The records in TOSEL are organized according to element BCD names. A more detailed explanation of TOSEL is presented in Section III of this volume. The additional cards needed are (Ref 1):

- **ASETI** - Defines the DOF in the analysis set. For the DMAP the analysis set should consist of only the Z-DOF for every non-wing-root node.
- **CAER01** - Divides the aero panels into equal boxes for the Doublet-Lattice aero theory. The param-
eters on this card are determined by the wing planform and the user's needs.

**AEFACT** - Divides the aero panels into unequal boxes, used in conjunction with the CAERO1 card.

**PAERO1** - Defines associated bodies for the CAERO1 card.

**EIGR** - Lists parameters for the real eigenvalue analysis. The frequency range of interest should be from 0.0 to 0.0 in order to include only the rigid body pitch mode. There is only one estimated and desired root in the frequency range. The eigenvector should be point normalized on the Z-displacement of the most in-board leading edge node, not including the wing root.

**AERO** - Defines the basic aero parameters. The DMAP uses the parameter Q, the dynamic pressure, and the mach number from the MKAERO card instead of the parameters on the AERO card. Therefore the values on the aero card do not matter, but they have to be positive to allow the DMAP to complete the processing of the AERO card.

**MKAERO** - Defines the mach number and reduced frequency for the desired flight condition. In the DMAP only one mach number and reduced frequency are allowed. The mach

27
number should be subsonic and representative of the desired flight condition. The reduced frequency should be a positive number very close to zero.

**SPLINE1** - Defines the parameters for generation of a surface spline to interpolate the out-of-plane displacements on the structural model to out-of-plane displacements and slopes on the aero model. The SPLINE1 card is used in exactly the same way as it is in the flutter analysis rigid format.

**SET1** - List of structural grid points to be used with the SPLINE1 card. Normally the list is only the nodes on the upper surface.

**PARAM cards**

**AOAP** - Used to set initial AOA. Real number. It multiplies the rigid body pitch mode eigenvector after the point normalization is done as specified on the EIGR card.

**ELOOP** - Tolerance of convergence for the iterative loop. It is compared to the norm of the change of deformations for each iteration. Real number.

**LMODES** - Number of modes to be used in the modal flutter formulation. Since
the only desired mode is the rigid body pitch mode LMODES is one. LMODES is normally used when several eigenvectors are considered in a flutter analysis. In these cases the different eigenvectors are arranged as columns in a forcing vector for determination of modal pressures. LMODES would then be the number of columns. LMODE is a real number.

\( Q \) - Dynamic pressure. Must correspond to the desired flight condition. Real number.

\( RC \) - Root chord length. Used in formulating aerodynamic coefficients. Real number.

\( \text{WAREA} \) - Planform area of the model. Used in calculating aerodynamic coefficients. Real number.

As an option, camber can be added to the wing. If camber is required then an additional PARAM card should set \( \text{ICAMB} = 1 \). Also, the CAMBER vector must be input via DMI cards. CAMBER is as long as the number of corners on the aero boxes. If two or more aero panels are used then the common nodes on both panels are counted twice. The values
that are in CAMBER are the Z displacement of each aero box corner due to the camber of the airfoil. In addition, any control surface deflection can be treated as if it were a change in camber.

Finally, in order to run this DMAP for aeroelastic analysis of flexible wings, the case control and executive card decks must be formulated. The executive card deck must reflect the fact that a DMAP is being run rather than a rigid format. The case control deck notes the set identification number of the EIGR card in the Bulk Data Deck. The case control lists the two subcases and the SPC identification numbers for each. In addition, output requests as needed are specified in the case control. The values of the first subcase correspond to the initial pitch configuration; those of the second subcase are a final result of the iteration loop. All the available output requests for the static loads analysis rigid format are also available for the DMAP.
Bibliography


Vita

Kim Jones was born on 28 February 1958 in Weisbaden, Germany. He graduated from high school in Clearwater, Florida in 1976 and was subsequently accepted into the Air Force Academy. A year after receiving a Bachelor of Science Degree in Aeronautical Engineering, he entered the School of Engineering, Air Force Institute of Technology, in June 1981.

Permanent Address: 837 Island Way
Clearwater, Florida 33515
### LOAP MAP - GIASCHE

<table>
<thead>
<tr>
<th>BLOCK</th>
<th>ADDRESS</th>
<th>LENGTH</th>
<th>FILE</th>
<th>DATE</th>
<th>PROCESSOR</th>
<th>PARTITION</th>
<th>HARDWARE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>/OPEN/</td>
<td>111</td>
<td>4704</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/SIZE/</td>
<td>3015</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIASCHE</td>
<td>5016</td>
<td>10301</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
<tr>
<td>/SYSTEM/</td>
<td>15317</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PROGRAM 1</td>
</tr>
<tr>
<td>/PACK/</td>
<td>13320</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIAS</td>
<td>15325</td>
<td>1725</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
<tr>
<td>/DATA/</td>
<td>17232</td>
<td>1024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SUBROUTINE 1</td>
</tr>
<tr>
<td>/TRACE/</td>
<td>20276</td>
<td>44</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INIT</td>
<td>20342</td>
<td>1322</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
<tr>
<td>MYSCC</td>
<td>22044</td>
<td>50</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
<tr>
<td>SSPL</td>
<td>22134</td>
<td>704</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
<tr>
<td>CMATS</td>
<td>23040</td>
<td>715</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
<tr>
<td>INVERSE</td>
<td>23235</td>
<td>340</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
<tr>
<td>CORZ</td>
<td>23615</td>
<td>11</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
<tr>
<td>COUP</td>
<td>23626</td>
<td>20</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
<tr>
<td>GLOB</td>
<td>23666</td>
<td>20</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
<tr>
<td>PACK</td>
<td>23686</td>
<td>114</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
<tr>
<td>POT</td>
<td>24002</td>
<td>73</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
<tr>
<td>L TFL</td>
<td>24075</td>
<td>41</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
<tr>
<td>BEST</td>
<td>24134</td>
<td>310</td>
<td>LCO</td>
<td>11/12/82</td>
<td>PTH</td>
<td>4.8</td>
<td>564</td>
<td>666X 1</td>
</tr>
</tbody>
</table>

THE REST OF THE LOADER MAP IS NONAPPLICABLE
PROGRAM CIASCHK

ENTRY POINTS
10272 CIASCHK

VARIABLES

FILE NAME MODE
0 INPUT

FILE MODES

COMMON BLOCS LENGTH
ZOPEN 2500
75177 1

STATISTICS

PROGRAM LENGTH
BUFFER LENGTH
CM LAYED COMMON LENGTH
470000 CM USED
SUBROUTINE CIA5

INTEGER ACPT,TOSEL,ECTA,CPLA,BCPA,SIL/SAS,CTAK,TTT,T,

PGC,VTMA,VTDA,VTM,VTN,AIDMT,F

INTEGER SAS,ICTAK,TTT,T,VTMA,VTDA,VTM,VTN,AIDMT,F

REAL Z

DIMENSION TRAIL(7),MCBA(7),MCBR(7),NAME(2),IZ(1)

EQUIVALENCE (IZ(1),Z(1))

COMMON /A

COMMON /SYSTEM/IBUP,

COMMON /PACK/TPITM,TPOUT,TROV,IKRO,INCR,

COMMON /OPEN/IZ(1)

STRING FOR MESSAGE UTILITY

DATA NAME,MCNIA,AMREAD/

DATA ACPT,TOSEL,ECTA,CPLA,BCPA,SIL/101,102,103,104,105,106/

DATA OUTPUT DATA BLOCKS LOCAL NAMES

DATA SAS,CTAK,TTT,T,VTMA,VTDA,VTM,VTN,AIDMT/}

&201,202,203,204,205,206,207,208,209/

INCR=1

GET END OF OPEN CORE

LCORE+CORE(E,A)

SET UP A BUFFER FOR USE WITH OPEN AND READ

BUF=LCORE-IBUP+1

IF (BUF.LE.0) GO TO 500

AN "M" BEFORE THE DATA BLOCK NAME SPECIFIES THE FIRST

ADDRESS OF THE BLOCK IN OPEN CORE

AN "L" BEFORE THE DATA BLOCK NAME SPECIFIES A PARAMETER

WHICH IS THE LENGTH OF THE DATA BLOCK IN OPEN CORE

DATA BLOCK ACPT WILL BE FIRST IN OPEN CORE

HTACPT=1

OPEN 'AERODYNAMIC CONNECTION AND PROPERTY TABLE'

CALL OPEN(ACPT,2(9999),0)

GET 4*2*MF WORDS OF ACPT

WORD 1 IF: 1 - DL*4 (DOUBLING-LATTICE METHOD)

WORD 2 - STRIP THEORY

WORD 3 - (NOT USED)

WORD 4 - # OF PANELS (MF)

WORD 5 - # OF PARFMS (NOT USED)

WORD 6 - # OF BOXES (MF)

WORD 7 - FRACTION OF BOX CHORD FROM CENTER OF PRESSURE

TO DOWNWASH CENTER (NOT USED)

WORD 8 - BOXES PER CHORD (MCR)

WORD 9 - LAST R100 ON THE PANEL (BARAY)

READ THE FIRST 2 WORDS

CALL READ(ACPT,IZ(HACPT),2,0,M)

IF=I12(HACPT+1)

READ WORDS 3 TO 4*2*MF
SUBROUTINE CIA3 74/74 CPT=1

CALL READ(ACPT,1(NACPT),3+2*RP,1,M)

60

CC Hi IS THE NUMBER OF AERO BOXES AND 1/4 CHORD, 1/2 SPAN AERO POINTS
Hi=I2(NACPT+1)

CC CLOSE ALL NODES TO ACPT, BUT KEEP WHAT'S IN OPEN CORE ALREADY
CALL CLOSE(ACPT,1)

CC FETCH TRAILER TO TOSEL (TOP SURFACE ELEMENT LIST) USER SUPPLIED
ON DTE CARDS

65

TRAILER(1)=TOSEL
CALL RDTR(1)

CC TRAILER OF TOSEL CONTAINS:

CC WORD 1 CHN DRM NAME

70

CC WORD 2 # OF ENTRIES IN TOSEL

CC WORD 3 0

CC WORD 4 0

CC WORD 5 0

CC WORD 6 0

CC Hi IS THE NUMBER OF STRUCTURAL ELEMENTS LOADED AND ALSO

CC THE NUMBER OF DEPENDENT STRUCTURAL POINTS WHEN UNIFORM PRESSURE IS

CC ASSUMED OVER STRUCTURAL BOXES

CC

CC LATER ON (AFTER BUCKS ARE IRONED OUT) WILL READ TOSEL UNTIL #HEL

CC THEN SAY HD=H

80

CC

CC IT'S ALSO THE SIZE OF TOSEL

CC HD=TRAIL(2)

CC MTOSEL=NTOS+3+2*NP

CC READS IN ALL OF TOSEL

85

CC USES 1ST BUFFER
CALL COPY(TOSEL,1(BUF1),0)

CC READ TOSEL(MTOSEL,HD,1,M)

CC CALL CLOSE(1)

90

CC GETS SIZE OF GPLA

CC CPLA (GRID POINT LIST-AERODYNAMIC)

CC TRAILER(1)=TRAIL
CALL RDTR(1)

CC WORD 7 = # OF GRID POINTS + # OF SCALAR POINT (ASSUMED=0) + # OF

CC EXTRA POINTS (ASSUMED=0) + # OF AERO POINTS

95

CC LPCPLA=TRAIL(2)

CC LPCPLA=AERODYNAMIC BECAUSE THERE ARE 4 WORDS PER POINT (WORD 1 = IDP

CC WORDS 2,3,4 = X,Y,Z)

CC

CC MCPPA = BASIC GRID POINT DEFINITION TABLE - AERODYNAMIC

100

CC LPCPLA=AERODYNAMIC

CC READ IN ALL OF GPLA TO GET IDP'S OF POINTS
CALL COPY(GPLA,1(BUF1),0)

CC MCPPA=MTOS+2
CALL READ(GPLA,1(MCPPA),LCPA,1,M)

CC CALL CLOSE(GPLA,1)

CC MCPPA=MACTP(LCPA)

CC RCOPY=BUF1-1

CC CHECK TO SEE IF ENOUGH OPEN CORE IS AVAILABLE
IF(CORE.LT.(MACTP-LCPPA)) GO TO 500

105

CC READ IN ALL OF MCPPA TO GET COORDINATES OF POINTS
CALL COPY(MCPPA,1(BUF1),0)

CC CALL READ(MCPPA,1(MCPPA),LCPPA,1,M)

CC CALL CLOSE(MCPPA,1)

CC ICROM=5700 IS THE INTERNAL IDENTIFIER OF CONME I ELEMENTS
SUBROUTINE CIAA
       74/74   OPT=1
       PRT 4.8+564
       11/03/82  20.07.46

       115  CCC THE FOLLOWING 6 STATEMENTS LOCATE THE RECORD FOR THESE
             CCC ELEMENTS IN THE ECTA TABLE
             CCC OPENS ECTA FOR READING
       120  CCC FECTA = ELEMENT CONNECTION TABLE - ARRHYTHMICT
             CCC CALL GORENT(ECTA, IZ(BUFL), 0)
             CCC FROM HERE UNTIL STATEMENT LABEL 2 NON-CQMEM2
             CCC ELEMENTS ARE SCREENED OUT, THE RECORD CONTAINING ALL
             CCC THE CQMEM2 ELEMENTS IS LOCATED AND POSITIONED FOR READING
             ICQMEM=5300
             NECTA=HDCPA+1NCPA
             1 CALL READ (ECTA, ICQMEM, 1, 0, N)
             IF (IZ(NECTA), EQ, ICQMEM) GO TO 2
       125  CCC IF 4TH ARGUMENT OF READ EQUALS ONE THEN SKIP TO END OF RECORD
             CCC WITHOUT TRANSFER OF DATA
             CALL READ(ECTA, IZ(NECTA), 0, 1, N)
             CCC IF N NOT EQUAL TO ZERO THEN WE HAVE HIT THE END OF THE RECORD
             CCC A FATAL ERROR
             IF(N.NE.0)GOTO 500
       130  CCC GO TO NEXT RECORD LOOKING FOR CQMEM2
             CCC GO TO 1
       135  CCC GET HERE WHEN CQMEM2 RECORD IS FOUND
       140  CCC S I L IS THE SCALAR INDEX LIST (SIMILAR TO AN ID ARRAY)
             2 TRAILER: S I L
             CALL NDFTRAIL(TRAFF)
       145  CCC SIZE OF S I L = NDFPA1
             NDF=TRAIL(3)
             NCF=TRAIL(2)
             IF(NCF.NE.NDFPA/6) GOTO 500
       150  CCC OPEN 1 MORE CINO BUFFER, TTITTT MUST BE OPEN FOR
             CCC WRITING WHILE ECTA IS OPEN FOR READING
       155  CCC BUFP2=BUFP1-1BUFP
             MCDFL=BUFP2-1
             CCC, BECAUSE THERE ARE 7 WORDS PER CURRENT CQMEM2 ELEMENT
             CCC ONLY KEEP 7 WORDS OF CURRENT ECTA RECORD IN CORE AT A TIME
             LECTA=7
       160  CCC CHECK FOR ENOUGH SPACE IN CINO TO PUT 2 BUFFERS
             IF(MCDFL.LT.MECTA+LECTA) GOTO 500
             CCC THE NEXT SEVERAL LINES DETERMINE THE POSITION AND
             CCC SIZE OF VARIOUS ARRAYS TO BE PLACED IN OPEN CORE
             CCC
       165  CCC ND = # OF OUTPUT POINTS FOR SUBROUTINE SPLINE (IN THIS
             CCC CASE IT ALSO HAPPENS TO BE THE # OF ELEMENTS ON THE UPPER SURFACE)
             ND=HECTA+LECTA
             LND=2*ND
             NTEX=NDF+LND
       170  CCC TEx IS NCF-R1 CONTAINS X COORDS OF GRID POINTS
             CCC TEx IS NCF-R1 CONTAINS Y COORDS OF GRID POINTS
             CCC ONE WORD FOR ALL GRID POINTS IS ALLOCATED, WHEN GOING THROUGH
             CCC TOSEL (LIST OF TOP SURFACE ELEMENTS) UPPER SURFACE NODE COORD
             CCC WILL BE FILLED IN, LEAVING THE REST FILLED WITH THE FLAG VALUE.
             CCC THEN, LATER ON, ONLY THE UPPER SURFACE NODES WILL BE PUT INTO
             CCC TEIR AND TEYR
             LTE=NCF
             NTEX=NTEX+LTEX
             LTE=NCF
             LTC=NLDF+LTEX
             LTC=NLDF
             LTE=NCF
SUBROUTINE C1AT  74,74  OPT=1  FTS 4.4+64  11/03/82  20.07.46  PAGE 4

CCC TTTT IS NhOP-BY-ONE BUT ONLY ONE COLUMN IS BUILT AND FACED.
CCC OUT AT A TIME. IN A COLUMN, PUT (1/4) AREA AT EACH UPPER SURFACE
CCC RrNMNT'S TO ROW. THIS APPORTIONS 1/4 OF TOTAL ELEMENT
CCC LOAD TO EACH COLUMN NODE. NOTE! THIS ASSUMES APPROXIMATELY
CCC SQUARE ELEMENTS.
CCC NVCLS=11TTT+11TTT
CCC LVCLS=RHOP
CCC ICOREM=NVCLS+LVCLS
CCC CHECK FOR ENOUGH SPACE IN GEMO TO PUT THE ABOVE ARRAYS
CCC IF(INCOREM,N1.1COREM) GO TO 900
CCC FROM HERE TO STATEMENT LABEL 7 ARRAYS NTEX,NTET
CCC AND,NVCLS ARE INITIALIZED
CCC VCLS IS HOP-BY-ONE VECTOR TO DOT INTO GEMO TO GET TOTAL LIFT.
CCC IT IS ALSO USED TO PARTITION PCT.
CCC IT CONTAINS A 1.0 IN EACH UPPER SURFACE NODE TO DOF
CCC NTEX AND NTET ARE INITIALIZED TO FLAG IN ORDER TO LATER ON
CCC SPOT ANY VALUES IN THEM THAT HAVE NOT BEEN USED
CCC USED ITEMS HAD BETTER BE UPPER SURFACE NODES FROM CONNECTIVITIES
CCC OF TOSEL CODENI'S
CCC DC 9 J=1,RHOP
CCC ZENTEX=J-1=FLAG
CCC ZENTET=J-1=FLAG.
CCC CONTINUE
CCC DO 7 J=1,LVCLS
CCC ZENVCLS=J-1=0.0
CCC CONTINUE
CCC CALL G0PEN(TTTT,Z(RHOP),1)
CCC SPECIFIES CHARACTERISTICS OF ARRAYS FOR PACK SUBROUTINE
CCC REAL SINGLE PRECISION INPUT FROM 2 ARRAY
CCC TYP0UT=1
CCC REAL SINGLE PRECISION OUTPUT TO DATA BLOCK
CCC TYP0UT=1
CCC INITIAL ROW POSITION
CCC IR0U=1
CCC IF NOT WILL BE REOR Prior TO EACH CALL TO PACK SINCE MULTIPLE
CCC CESP'S ARE MATRIX CONTROL BLOCKS FOR PACK ROUTINES
CCC IT WILL BE TRAILERS WHEN FINISHED, CONTENT IS THAT OF TRAILER
CCC WORD 1 = GEMO OR NAME
CCC WORD 2 = # OF COLUMNS, ZERO INITIALLY, ACCUMULATED BY PACK
CCC WORD 3 = # OF ROWS
CCC WORD 4 = MATRIX SHAPE
CCC WORD 5 = PRECISION = REAL SINGLE PRECISION
CCC WORD 6 = MAXIMUM DISTANCE BETWEEN NONZERO TERMS
CCC WORD 7 = DENSITY
CCC TRAILER FOR TTTT (TTTT SHOULD BE A HOP=1 ARRAY)
CCC MFRM(1)=TTTT
CCC MFRM(2)=0
CCC MFRM(3)=TTTT
CCC MFRM(4)=2
CCC MFRM(5)=1
CCC MFRM(6)=0
CCC MFRM(7)=0
CCC LOOP 7 CALCULATES THE FOLLOWING DATA BLOCKS
CCC TTTT,NVCLS
SUBROUTINE GEA

DO OVER ALL ELEMENTS OF TOSEL, DO LENGTH SHOULD BE SIZE OF
TOSEL (NOT # OF SPLINE OUTPUT POINTS)
DO 3 I=1,N
READS IN ONE CODMEN2 ELEMENT AT A TIME FROM ECTA
* CALL READECTA,Z(NECTA),7,0,N
MAKE SURE THE CODMEN2 ELEMENTS PRESENTED BY TOSEL
ARE IN ASCENDING CONSECUTIVE ORDER OR THE FOLLOWING MESSES UP

IF THIS CODMEN2 IS NOT ON THE UPPER SURFACE TRY THE NEXT ONE
CONTEST OF THE 7 WORDS PER CODMEN2 ELEMENT IS
WORD 1 - EID, ELEMENT NUMBER
WORD 2 - PID, PROPERTY ID
WORDS 3,4,5,6 - CORNER NODE #'S
WORD 7 - TH, MATERIAL ORIENTATION ANGLE.
FROM HERE TO LABEL 3 IS DONE ONCE FOR EACH CODMEN2 ON THE UPPER
SURFACE

IF(ZZ(NECTA),K,EZMN(*TOSEL+1-1)) GO TO 4

CONNER NODE #'S FOR EACH CODMEN2 ELEMENT
I1=1Z(NECTA+1)
I2=1Z(NECTA+3)
I3=1Z(NECTA+4)

LATER ON (RIGHT AFTER LOOP 6) THE RCPA POINTERS (ICP) ARE
CHECKED. IF THEY HAVEN'T CHANGED FROM ZERO THEN THE NODE
ISN'T IN RCPA, A BIG ERROR
ICPO=0
ICP1=0
ICP2=0
ICP3=0

LOCATES APPROPRIATE NODES IN CPLA
SEARCH CPLA - LIST OF ALL GRID POINT IDENTIFIERS FOR EACH
CONNER NODE. THE POSITION IN CPLA IS THE POINT TO THE
ID,X,Y,Z OF THE NODE IN RCPA
DO 4 J=1,LCPLA
K=1CPLA+J-1
IF (IG1.EQ.IZ(K)) IC1=1RCPA(J-1)*4
IF (IC2.EQ.IZ(K)) IC2=1RCPA(J-1)*4
IF (IG3.EQ.IZ(K)) IC3=1RCPA(J-1)*4
IF (IC4.EQ.IZ(K)) IC4=1RCPA(J-1)*4
CONTINUE

IF NODE IN ECTA IS NOT IN CPLA, THE WORLD IS UPSIDE DOWN!
IF (ICP1.EQ.0) GO TO 500
IF (ICP2.EQ.0) GO TO 500
IF (ICP3.EQ.0) GO TO 500
IF (ICP4.EQ.0) GO TO 500
CONTINUE

PUTS 1.0 INTO EACH UPPER SURFACE NODE Z DOF INTO VCLS
X(NVGLS+4*I1-1)=1.0
Z(NVGLS+4*IC1-4)=1.0
Z(NVGLS+4*IC2-4)=1.0
Z(NVGLS+4*IC3-4)=1.0
Z(NVGLS+4*IC4-4)=1.0

GET X,Y CODES OF CORNER NODES FROM RCPA
X1=1Z(1CP+1)
X2=1Z(1CP+2)
Y1=1Z(1CP+1)
Y2=1Z(1CP+2)

CONTINUE
SUBROUTINE CIALS 74/14 ORT=1

X4+Z(4CP+41)
Y4+Z(4CP+42)
CCC CENTROIDAL COORDINATES OF QUAD
CCC THE SPINE OUTPUT POINTS
290 Z(NNH+2*1-2)=(X1+X2+X3+X4)/4.
Z(NNH+1+1)=(Y1+Y2+Y3+Y4)/4.
CCC AREA OF ELEMENT
AREA= 5*(X3-X1)*(Y4-Y7)-(Y3-Y1)*(X4-X2))
CCC ZEROS OUT THE COLUMNS
DO 11 J=1,LTTTIT
11 P(WTTTTT+4-J)=0.0
CCC .25 LOAD ON MEMBRANES TO THEIR NODES IN Z DOP FOR EACH NODE
CCC IN TTTT
CCC PRESUMES ALMOST SQUARE ELEMENTS
300 Z(WTTTTT+4*IC)-=AREA,25
Z(WTTTTT+4*IC+2)=AREA,25
Z(WTTTTT+4*IC-4)=AREA,25
Z(WTTTTT+4*IC-6)=AREA,25
CCC LENGTH OF A COLUMN IN TTTT FOR PACK
CCC THE SPINE OF EACH COLUMN AT A TIME
CALL PACK(Z(WTTTTT),TTTT,MCRA)
CCC THE SPINE OF EACH COLUMN AT A TIME
310 Z(WTTTT-1*IC)=X1
Z(WTTTT-1*IC+2)=X2
Z(WTTTT-1*IC+3)=X3
Z(WTTTT-1*IC+4)=X4
Z(WTTTT-1*IC+5)=Y1
Z(WTTTT-1*IC+6)=Y2
Z(WTTTT-1*IC+7)=Y3
Z(WTTTT-1*IC+8)=Y4
3 CONTINUE
CCC THE SPINE OF EACH COLUMN AT A TIME
320 CALL CLOSE(ECTA,1)
CALL CLOSE(TTTT,1)
CALL WTRSL(MCRA)
CCC SET UP A MATRIX CONTROL BLOCK FOR PACKING VCLS
CCC TRAILER VCLS (SIZE = NDOP*1)
CCC MCRA(1)=VCLS
325 MCRA(2)=0
MCRA(3)=VCLS
MCRA(4)=2
MCRA(5)=1
MCRA(6)=0
MCRA(7)=0
330 CCC SET COLUMN LENGTH FOR PACKING VCLS
CCC PACKS THE ONE COLUMN OF VCLS
335 CALL PACK(Z(WVCLS),VCLS,MCRA)
CCC CALL WTRSL(MCRA)
CCC USES THE SPINE IN OPEN CORP USED BY TTTT, VCLS
CCC AN VTH
340 CCC PACKS THE ONE COLUMN OF VCLS
CCC CALL WTRSL(VTH,1,RBUF),1)
CALL WTRSL(VTH,2,RBUF),1)
SUBROUTINE GIAS  74/74   OPT=1

DEFINES LENGTH AND POSITION OF TETR AND TETR (MODIFIED
CUTT AND TETR), TO BE PACKED AS VTM AND VTR
CCC
345
CCC ACTUAL # OF NODES ON UPPER SURFACE IS UNKNOWN, IT MUST
CCC <= NCF. THE VARIABLE NC WILL COUNT THEM IN THE FOLLOWING LOOP,
CCC FOR NOW JUST SET ASIDE NCF WORDS, THEN AFTER REDUCTION LOOP,
CCC PACK OUT FIRST NC WORDS.
NICET$=NTETR
NTETR=NTETR+NCF
NC=0
CCC LOOP 13 CALCULATES THE VTM AND VTR MATRICES
CCC FILTERS OUT THE UPPER SURFACE VALUES OF TETR AND TETR
DO 13 J=1,NCF
IF(NTETR+J-1).EQ.FLAG) GO TO 13
Z(NTETR+MC)$=Z(NTETR+J-1)
Z(NTETR+NC)$=Z(NTETR+J-1)
NC=NC+1
13 CONTINUE
CCC IF NC<0 THEN FATAL ERROR, NO UPPER SURFACE NODES
IF(NC.GE.0) GO TO 500
CCC TRAILER OF VTM (SIZE = NC)
MCRA(1)=VM
MCRA(2)=0
MCRA(3)=NC
MCRA(4)=2
MCRA(5)=1
MCRA(6)=0
MCRA(7)=0
CCC TRAILER OF VTR (SIZE = NC)
MCRA(1)=VTR
MCRA(2)=0
MCRA(3)=MC
MCRA(4)=2
MCRA(5)=1
MCRA(6)=0
MCRA(7)=0
CCC PACK NTETR AND NTETR AS VTM AND VTR
CALL PACK(Z(NTETR),VTM,MCRA)
CALL PACK(Z(NTETR),VTR,MCRA)
CALL CLOSE(VTM,1)
CALL CLOSE(VTR,1)
CALL WATRNL(MCRA)
CALL WATRNL(MCRA)
CCC IDFS OF AERO ROSS' CORNERS START AT MIL=1
MIL=1000000
1=1
CCC MCRA EQUALS THE NUMBER OF CHORDWISE BOXES
CCC SETS POSITION AND SIZE IN OPEN CORE FOR AERO INPUT POINTS
CCC IN SUBROUTINE SPLINE
1STNR=0
1STVR=1
CCC LOOP UNTIL STATEMENT LABEL 14 IS SIMILAR TO LOOP 3 BUT FOR
CCC AERO POINTS
CCC THE PLAN: BUILD CORNERS OF SPLINE INPUT POINTS WHICH ARE THE
CCC 1ST VTR, 1/4 CORN OF THE AERO BOXES (UPPER THE DELTA CF IS
CCC PR12). GET CORN OF THIS POINT FROM THE CORNERS OF
SUBROUTINE CIAK 74/74 OPT-1  F1R 4.4524  11/03/82  20.07.46  PAGE 1

400  
405  COPPER POINTS IN RCDA. THESE ARE ONLY USED BY THE SPLINE

CCC SUPPORTING
  RMI=HTEX
  DO 14 I=1,4
  NCB=12(NACP+4+1)
  NMRAT=12(NACP+4+4P+1)
  LP=NMRAT+NFSTBX+1
  DO 20 J=1,LP
  ICONM=(J-1)/NCB+1+LSTDND+1
  IAP=I
  IAP3=0
  IAP4=0
  IF (IAP.EQ.0) GO TO 500
  IF (IAP3.EQ.0) GO TO 500
  IF (IAP4.EQ.0) GO TO 500
  16 CONTINUE

CCC THESE ARE THE COORDINATES OF AERO POINTS AT THE
CCC ONE QUARTER CHORD POINT OF THE ROX
CCC L IS BOX #, WITH ONE INPUT POINT PER BOX IT IS ALSO THE
CCC INDEPENDENT POINT #

440  Z(NH+2*(L-1)-(1)*3+(X1+X3)-X3-X1)/4)2.
  Z(NH+2*(L-1)-(1)*3+Y3+4*Y4)/4.
  L=L+1

CCC NEXT ROX
20 CONTINUE

445  LSTDND=I4A-NC
  NFSTBX=NMRAT+4
14 CONTINUE

CCC ALL THE ABOVE ARRAYS EXCEPT THE ND AND N1 ARRAYS ARE NO LONGER
CCC NEEDED, THEREFORE, TO CONSERVE OPER CORT, THE ND AND N1 ARRAYS
CCC ARE MOVED NOW TO THE BOTTOM AND THE SPACE REQUIRED BY SPLIN
CCC IS PUT RIGHT ABOVE THEM. THIS WIPE OUT ALL THE PREVIOUS
CCC ARRAYS

CCC DO 200 I=1,LND
200  ?(1)=?(1+NN-1)
SUBROUTINE CIAE 24/74 OPT=1

NMD=1
LNI=2*NLI
DO 290 I=1,LNI

290 Z((NMD+LNI-1)-Z(NNI+1-1))
      NNI=NMD+LNI
CC
CCC TAILOR GTAK (SIZE = # OF AERO BOXES TIMES NUMBER OF
CCC STRUCTURAL BOXES)
CCC PRESUMES CONSTANT PRESSURE ASSUMPTION ON STRUCTURAL BOXES
CCC
MCRA(1)=GTAK
MCRA(2)=0
MCRA(3)=HD
MCRA(4)=2
MCRA(5)=1
MCRA(6)=0
MCRA(7)=0
MCAY=NR1+NR2
CCC FLAG FOR TRANSPOSITION: KT=0 DO NOT TRANSPOSE OUTPUT
KT=0
CCC SYMMETRY FLAG FOR X-DIRECTION: EX=0 GENERAL
EX=0
CCC SYMMETRY FLAG FOR Y-DIRECTION: EY=0 GENERAL
EY=0
CCC FLAG FOR SLOPES: KD=0 INTERPOLATION ONLY
KD=0
CCC ATTACHMENT FLEXIBILITY: REAL > 0.0
DF=0.0
CCC 
CCC NXR=NR2
CCC NCF=10
CCC NCF=NR2-1
CCC SPECIFIES THE AMOUNT OF OPEN CORE REQUIRED TO RUN SEPLIN
NCF=(NR2**2)+1
CCC THIS CHECKS OPEN CORE TO SEE IF ENOUGH IS AVAILABLE
CCC IF(NCF.LT.(NSCR+NCN)) GO TO 500
CCC CALL SEPLIN(N1,Z(N1),NR1,NS,EX,EY,KD,EZ,Z(GTAK),NSCR,
CCC,ISIG)
CCC RESPECTS AMOUNT OF CORE LEFT ASIDE FOR BUFFERS (# OF BUFFERS
CCC REQUIRED CHANGES FROM 1 TO 2)
CCC
CCC CALL SEPLIN(N1,Z(N1),NR1,NS,EX,EY,KD,EZ,Z(GTAK),NSCR,
CCC,ISIG)
CCC PACK NI COLUMNS OF GTAK, EACH COLUMN IS NOT LONGER
CCC DO IA J=1,N1
CCC CALL PACK(Z(GTAK+(J-1)*HD),GTAK,MCPA)
CCC CONTINUE
CCC
CCC PERSIST A FAPN TEXT AND TEF (THIS TIME FOR AERO POINTS) AT
CCC POSITIONS NCAY AND NCAY+1
CCC LNFX=NCAY
LTEX=NCAY+1
LTYF=X1
LTYF=NCAY+1
CCC IF(NCF.LT.(MTY4+LTET)) GO TO 500
CCC LOOP 17 CALCULATES THE VTMA AND VTRAN MATRICES
SUBROUTINE GIAS 74/74 OPT=1

515  CCCC  COORDINATES OF 1/4 POINTS OF AFRO MODEL
    70(NTET+1)=2(NH+2*1-1-1)
    Z(NTET+1)=2(NH+2*1-1)
17  CONTINUE
     BROW=N1

520  CCCC  TRAILER VTMA (SIZE = N1*N1)
    MCBA(1)=VTMA
    MCBA(2)=0
    MCBA(3)=N1
    MCBA(4)=2
    MCBA(5)=1
    MCBA(6)=0
    MCBA(7)=0

525  CCCC  TRAILER VTBA (SIZE = N1*N1)
    MCBR(1)=VTBA
    MCBR(2)=0
    MCBR(3)=N1
    MCBR(4)=2
    MCBR(5)=1
    MCBR(6)=0
    MCBR(7)=0

530  CCCC  PACKS BOTH VTMA AND VTBA AS N1 BY 1 ARRAYS
       NOTICE BOTH PACKS USE SAME BROW VALUE
    CALL PACK(ZNTET),VTMA,MCBA
    CALL PACK(ZNTET),VTBA,MCBB

535  CCCC  TRAILER AIDMT (VECTOR OF ONES N1 LONG)
    CCCC  AIDMT IS USED IN CALCULATING THE C.O.I. FOR AFRO MODEL
    MCRA(1)=AIDMT
    MCRA(2)=0
    MCRA(3)=N1
    MCRA(4)=2
    MCRA(5)=1
    MCRA(6)=0
    MCRA(7)=0
    KROW=N1
    CALL PACK(ZKRAIDMT),AIDMT,MCRA
    CALL CLOSE(AIDMT,1)
    CALL WRITFL(MCRA)
    CALL WRITFL(MCBB)
    WAITMT=1

540  CCCC  TRAILER AIDMT (VECTOR OF ZEROS)
    DO 18 1=1,N1
    X(ZKRAIDMT+1-1)=1.0
     18  CONTINUE
    CALL COPEN(AIDMT,Z(BUF1),1)

545  CCCC  TRAILER AIDMT (VECTOR OF ONES N1 LONG)
    CCCC  AIDMT IS USED IN CALCULATING THE C.O.I. FOR AFRO MODEL
    MCRA(1)=AIDMT
    MCRA(2)=0
    MCRA(3)=N1
    MCRA(4)=2
    MCRA(5)=1
    MCRA(6)=0
    MCRA(7)=0
    KROW=N1
    CALL PACK(ZKRAIDMT),AIDMT,MCRA
    CALL CLOSE(AIDMT,1)
    CALL WRITFL(MCRA)
    CALL COPEN(SAS,Z(BUF1),1)

550  CCCC  COMPLEX
    TYPE=1

555  CCCC  FILE AT BEGINNING OF OPEN CORE
    NCAT=1
    KROW=2*N1

560  CCCC  FILE ONE (TYPE COMPLEX) CAN BE REAP ON CORE AS TWO

565  CCCC  COMPLEX
    N1=1*KROW

570  .
<table>
<thead>
<tr>
<th>Variables</th>
<th>Size</th>
<th>Type</th>
<th>Relocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1451 IAS</td>
<td>INTEGER</td>
<td>1A3</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1455 ICORER</td>
<td>INTEGER</td>
<td>1BDF</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1453 ICF</td>
<td>INTEGER</td>
<td>1BF</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1457 ICF2</td>
<td>INTEGER</td>
<td>1B7</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1611 IGI</td>
<td>INTEGER</td>
<td>1612 IGP</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1413 IGR</td>
<td>INTEGER</td>
<td>1614 IGA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1470 ISINC</td>
<td>INTEGER</td>
<td>2 ICR</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1617 J</td>
<td>INTEGER</td>
<td>1621 IZ</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1665 KD</td>
<td>INTEGER</td>
<td>1683 LK</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1662 KT</td>
<td>INTEGER</td>
<td>1663 KX</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1664 KY</td>
<td>INTEGER</td>
<td>1637 L</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1562 LNPMA</td>
<td>INTEGER</td>
<td>1562 LCP</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1572 LPLA</td>
<td>INTEGER</td>
<td>1561 LCPLA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1574 LMP</td>
<td>INTEGER</td>
<td>1560 LNP1</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1664 LP</td>
<td>INTEGER</td>
<td>1660 LSTMD</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1576 LTFX</td>
<td>INTEGER</td>
<td>1600 LTFY</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1602 LTTTIT</td>
<td>INTEGER</td>
<td>1604 LVCLS</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1554 MCBT</td>
<td>INTEGER</td>
<td>1705 MCBA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1714 MCBB</td>
<td>INTEGER</td>
<td>ARRAY</td>
<td></td>
</tr>
<tr>
<td>1674 M</td>
<td>INTEGER</td>
<td>1553 NAGPT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1671 MAIDMT</td>
<td>INTEGER</td>
<td>1702 NAME</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1644 MBARAT</td>
<td>INTEGER</td>
<td>1564 MBGPA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1635 MB</td>
<td>INTEGER</td>
<td>1663 MB</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1565 MDCRE</td>
<td>INTEGER</td>
<td>1557 MD</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1570 MDOF</td>
<td>INTEGER</td>
<td>1561 MCTA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1641 MPSTOR</td>
<td>INTEGER</td>
<td>1571 MGP</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1563 NPCPA</td>
<td>INTEGER</td>
<td>1661 NOTAR</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1546 HI</td>
<td>INTEGER</td>
<td>1573 HDR</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1642 HI</td>
<td>INTEGER</td>
<td>1555 HP</td>
<td>INTEGER</td>
</tr>
<tr>
<td>3 MRROW</td>
<td>INTEGER</td>
<td>PACKX</td>
<td></td>
</tr>
<tr>
<td>1667 MRSC</td>
<td>INTEGER</td>
<td>1675 MSHOR</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1575 MTEX</td>
<td>INTEGER</td>
<td>1633 MTEX</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1577 MRTE</td>
<td>INTEGER</td>
<td>1634 MTRER</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1473 NTL</td>
<td>INTEGER</td>
<td>1560 NTS5L</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1601 NTTTTT</td>
<td>INTEGER</td>
<td>1603 NVCLS</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1521 SAS</td>
<td>INTEGER</td>
<td>1520 SIL</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1514 TOSEL</td>
<td>INTEGER</td>
<td>1676 TRAIL</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1523 TTTTT</td>
<td>INTEGER</td>
<td>0 TYPER</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1 TYPFOUT</td>
<td>INTEGER</td>
<td>PACKX</td>
<td></td>
</tr>
<tr>
<td>1527 TVN</td>
<td>INTEGER</td>
<td>1525 VINHA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1530 TVN</td>
<td>INTEGER</td>
<td>1526 VTRA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1627 X1</td>
<td>REAL</td>
<td>1624 X2</td>
<td>REAL</td>
</tr>
<tr>
<td>1626 X3</td>
<td>REAL</td>
<td>1630 X4</td>
<td>REAL</td>
</tr>
<tr>
<td>1623 Y1</td>
<td>REAL</td>
<td>1625 Y2</td>
<td>REAL</td>
</tr>
<tr>
<td>1627 Y3</td>
<td>REAL</td>
<td>1631 Y4</td>
<td>REAL</td>
</tr>
</tbody>
</table>

**EXTERNALS**

<table>
<thead>
<tr>
<th>Type</th>
<th>Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORSF</td>
<td>2</td>
</tr>
<tr>
<td>CORDER</td>
<td>3</td>
</tr>
<tr>
<td>PACK</td>
<td>5</td>
</tr>
<tr>
<td>READ</td>
<td>17</td>
</tr>
<tr>
<td>WRUPDI</td>
<td>1</td>
</tr>
</tbody>
</table>

**Variables**

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Type</th>
<th>Relocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1451 IAS</td>
<td>INTEGER</td>
<td>1A3</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1455 ICORER</td>
<td>INTEGER</td>
<td>1BDF</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1453 ICF</td>
<td>INTEGER</td>
<td>1BF</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1457 ICF2</td>
<td>INTEGER</td>
<td>1B7</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1611 IGI</td>
<td>INTEGER</td>
<td>1612 IGP</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1413 IGR</td>
<td>INTEGER</td>
<td>1614 IGA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1470 ISINC</td>
<td>INTEGER</td>
<td>2 ICR</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1617 J</td>
<td>INTEGER</td>
<td>1621 IZ</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1665 KD</td>
<td>INTEGER</td>
<td>1683 LK</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1662 KT</td>
<td>INTEGER</td>
<td>1663 KX</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1664 KY</td>
<td>INTEGER</td>
<td>1637 L</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1562 LNPMA</td>
<td>INTEGER</td>
<td>1562 LCP</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1572 LPLA</td>
<td>INTEGER</td>
<td>1561 LCPLA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1574 LMP</td>
<td>INTEGER</td>
<td>1560 LNP1</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1664 LP</td>
<td>INTEGER</td>
<td>1660 LSTMD</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1576 LTFX</td>
<td>INTEGER</td>
<td>1600 LTFY</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1602 LTTTIT</td>
<td>INTEGER</td>
<td>1604 LVCLS</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1554 MCBT</td>
<td>INTEGER</td>
<td>1705 MCBA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1714 MCBB</td>
<td>INTEGER</td>
<td>ARRAY</td>
<td></td>
</tr>
<tr>
<td>1674 M</td>
<td>INTEGER</td>
<td>1553 NAGPT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1671 MAIDMT</td>
<td>INTEGER</td>
<td>1702 NAME</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1644 MBARAT</td>
<td>INTEGER</td>
<td>1564 MBGPA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1635 MB</td>
<td>INTEGER</td>
<td>1663 MB</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1565 MDCRE</td>
<td>INTEGER</td>
<td>1557 MD</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1570 MDOF</td>
<td>INTEGER</td>
<td>1561 MCTA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1641 MPSTOR</td>
<td>INTEGER</td>
<td>1571 MGP</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1563 NPCPA</td>
<td>INTEGER</td>
<td>1661 NOTAR</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1546 HI</td>
<td>INTEGER</td>
<td>1573 HDR</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1642 HI</td>
<td>INTEGER</td>
<td>1555 HP</td>
<td>INTEGER</td>
</tr>
<tr>
<td>3 MRROW</td>
<td>INTEGER</td>
<td>PACKX</td>
<td></td>
</tr>
<tr>
<td>1667 MRSC</td>
<td>INTEGER</td>
<td>1675 MSHOR</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1575 MTEX</td>
<td>INTEGER</td>
<td>1633 MTEX</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1577 MRTE</td>
<td>INTEGER</td>
<td>1634 MTRER</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1473 NTL</td>
<td>INTEGER</td>
<td>1560 NTS5L</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1601 NTTTTT</td>
<td>INTEGER</td>
<td>1603 NVCLS</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1521 SAS</td>
<td>INTEGER</td>
<td>1520 SIL</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1514 TOSEL</td>
<td>INTEGER</td>
<td>1676 TRAIL</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1523 TTTTT</td>
<td>INTEGER</td>
<td>0 TYPER</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1 TYPFOUT</td>
<td>INTEGER</td>
<td>PACKX</td>
<td></td>
</tr>
<tr>
<td>1527 TVN</td>
<td>INTEGER</td>
<td>1525 VINHA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1530 TVN</td>
<td>INTEGER</td>
<td>1526 VTRA</td>
<td>INTEGER</td>
</tr>
<tr>
<td>1627 X1</td>
<td>REAL</td>
<td>1624 X2</td>
<td>REAL</td>
</tr>
<tr>
<td>1626 X3</td>
<td>REAL</td>
<td>1630 X4</td>
<td>REAL</td>
</tr>
<tr>
<td>1623 Y1</td>
<td>REAL</td>
<td>1625 Y2</td>
<td>REAL</td>
</tr>
<tr>
<td>1627 Y3</td>
<td>REAL</td>
<td>1631 Y4</td>
<td>REAL</td>
</tr>
</tbody>
</table>

**EXTERNALS Type Arcs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORSF</td>
<td>2</td>
</tr>
<tr>
<td>CORDER</td>
<td>3</td>
</tr>
<tr>
<td>PACK</td>
<td>5</td>
</tr>
<tr>
<td>READ</td>
<td>17</td>
</tr>
<tr>
<td>WRUPDI</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE GPS</td>
<td>74/74</td>
</tr>
<tr>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>STATEMENT LABELS</td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>1</td>
</tr>
<tr>
<td>216</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>0</td>
<td>200</td>
</tr>
<tr>
<td>LOOPS LABEL INDEX FROM-TO LENGTH PROPERTIES</td>
<td></td>
</tr>
<tr>
<td>167 9</td>
<td>J</td>
</tr>
<tr>
<td>177 7</td>
<td>J</td>
</tr>
<tr>
<td>236 3</td>
<td>I</td>
</tr>
<tr>
<td>243 6</td>
<td>J</td>
</tr>
<tr>
<td>353 13</td>
<td>J</td>
</tr>
<tr>
<td>473 13</td>
<td>J</td>
</tr>
<tr>
<td>5-2 14</td>
<td>J</td>
</tr>
<tr>
<td>513 29</td>
<td>J</td>
</tr>
<tr>
<td>516 15</td>
<td>KK</td>
</tr>
<tr>
<td>675 200</td>
<td>I</td>
</tr>
<tr>
<td>711 210</td>
<td>I</td>
</tr>
<tr>
<td>760 16</td>
<td>J</td>
</tr>
<tr>
<td>1017 17</td>
<td>I</td>
</tr>
<tr>
<td>1062 18</td>
<td>I</td>
</tr>
<tr>
<td>1122 19</td>
<td>I</td>
</tr>
<tr>
<td>1130 20</td>
<td>I</td>
</tr>
<tr>
<td>1150 21</td>
<td>I</td>
</tr>
<tr>
<td>COMMON BLOCKS LENGTH</td>
<td></td>
</tr>
<tr>
<td>/ /</td>
<td>1</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>PACEX</td>
<td>5</td>
</tr>
<tr>
<td>ZOPEN</td>
<td>1</td>
</tr>
<tr>
<td>STATISTICS PROGRAM LENGTH</td>
<td>1729</td>
</tr>
<tr>
<td>CM LAPPED COMMON LENGTH</td>
<td>78</td>
</tr>
<tr>
<td>CM BLANK COMMON LENGTH</td>
<td>18</td>
</tr>
<tr>
<td>PROGRAM CM USED</td>
<td>20900</td>
</tr>
</tbody>
</table>
SUBROUTINE INIT

COMMON/SYSTEM/IBUF
COMMON/ZDATA/ZACPT(49),ZGCFA(166),ZTOSEL(4),ZCPLA(4),ZCIL(18)
1,ZCETA(64,4)
COMMON/DTRAIL/TACPT(4),TNECA(6),TUSER(6),TCLPLA(6),TSEL(6),
1,TECCTA(4)
COMMON/REC/TACPT,TTOSEL,TECCT,TUSER,TECCT,ZTOSEL,ZCPLA,ZCIL,ZCIL,
1,ZCPLA,ECIL,TECCTA,ZACPT(49)
REAL ZACPT,ZGCFA
EQUIVALENCE (ZACPT(49),ZACPT(49))
IBUF=8

READ ACPT AND TRAILER
READ(5,493) (TACPT(I),I=1,6)
FORMAT(24X,618)
READ(5,500)
FORMAT(14X)
READ(5,591) (ZACPT(I),I=1,2)
FORMAT(13X,13X,13)
READ(5,592) (ZACPT(I),I=1,2)
FORMAT(23X,11,11,15X,11,15X,11)
IF(IZACPT(2).EQ.0) GOTO 501
IF(IZACPT(2).EQ.3) GOTO 502
READ(5,970) (ZACPT(I),I=8,10)
FORMAT(23X,11,1E16.9)
GOTO 510
READ(5,501) (ZACPT(I),I=7,10)
FORMAT(6X,13X,11)
GOTO 510
READ(5,505) (ZACPT(I),I=7,9)
FORMAT(23X,11,15X,11,1E16.9)
CONTINUE
IF(IZACPT(2).EQ.3) GOTO 511
READ(5,971) (ZACPT(I),I=1,14)
FORMAT(6X,13X,11)
GOTO 512
READ(5,970) (ZACPT(I),I=1,12,14)
CONTINUE
IF(IZACPT(2).EQ.7) GOTO 520
IF(IZACPT(2).EQ.3) GOTO 513
READ(5,972) (ZACPT(I),I=1,16)
FORMAT(2X,1E16.9,2X,15X,11)
READ(5,970) (ZACPT(I),I=1,20,22)
GOTO 530
DO 514 J=1,2
READ(5,971) (ZACPT(I),I=8,10)
GOTO 530
READ(5,972) (ZACPT(I),I=1,14)
GOTO 530
READ(5,971) (ZACPT(I),I=1,15,18)
GOTO 530
SUBROUTINE INIT 74/74 OPT=1

READ(5,505)(IACPT(I),I=19,21),ZACPT(22)
530 CONTINUE
IF (ZACPT(2).EQ.2) NR=5
IF (ZACPT(2).EQ.3) COTO 514
DO 10 J=1,NR
10 READ(5,971)(ZACPT(22+4*(J-1)+4*I),I=1,6)
READ(5,973)(ZACPT(1),I=39,40)
973 FORMAT(8X,2E16.9)
COTO 526
515 READ(5,516)(ZACPT(23),I=ZACPT(1),I=14,26)
516 FORMAT(8X,E16.9,3(1X,N1))
READ(5,970)(ZACPT(27),I=ZACPT(1),I=28,30)
DO 517 J=1,6
517 READ(5,971)(ZACPT(30+4*(J-1)+4*I),I=1,4)
READ(5,518)(ZACPT(1),I=47,49)
518 FORMAT(9X)
526 READ(5,500)
CCC
CCC READ TSOEL AND TRAILER
CCC
80 3 FORMAT(19H BEGINNING OF TSOEL)
READ(5,930)(TSOEL(I),I=1,6)
READ(5,500)
READ(5,940)(ZTOSEL(I),I=1,4)
FORMAT(24X,418)
840 CCC
CCC READ ECTA AND TRAILER
CCC
READ(5,930)(TECTA(I),I=1,6)
READ(5,500)
90 DO 43 J=1,6
43 READ(5,940)(TECTA(I,J),J=1,6)
DO 45 J=1,6
45 READ(5,950)(TECTA(I,J),J=1,6)
FORMAT(24X,618)
IF (J.NE.1) COTO 50
910 DO 95 I=1,6
95 READ(5,955)(TECTA(M)*H(I)+6,1),I=1,8)
951 FORMAT(24X,618)
952 READ(5,955)(TECTA(I),I=55,59)
953 FORMAT(24X,518)
GOTO 15
90 IF (J.NE.2) COTO 51
954 DO 96 I=1,4
96 READ(5,955)(TECTA(M)*H(I)+6,2),I=1,8)
955 FORMAT(24X,618)
GOTO 15
90 IF (J.NE.2) COTO 51
956 DO 96 I=1,7
96 READ(5,955)(TECTA(M)*H(I)+6,3),I=1,8)
956 CONTINUE
SUBROUTINE INIT 74/74 OPT-1

DO 957 M=1,4
   READ(5,952)(ZICPA(R*(M-1)+146,4),I=1,R)
   FORMAT(5,955)ZICPA(39,4)
   CONTINUE

READ CPLA AND TRAILER
READ(5,930)ZECPLA(1),I=1,6
READ(5,900)
FORMAT(RX,618)
NR=0
IF(IZACPIT(2),NE.1)NR=4
DO 16 J=1,NR
   READ(5,952)(ZCPLA(R*(J-1)+64),I=1,R)
   CONTINUE
   IF(IZACPIT(2),EQ.2)GOTO 540
   IF(IZACPIT(2),EQ.3)GOTO 541
   READ(5,962)ZCPLA(1),I=31,36
   GOTO 350
   READ(5,942)ZCPLA(1),I=39,41
   FORMAT(RX,318)
   FORMAT(RX,18)
   CONTINUE
   FORMAT(RX,618)
READ RCPA AND TRAILER
READ(5,930)ZRCPLA(1),I=1,6
READ(5,900)
READ(5,931)ZRCPLA(1),ZRCPLA(2)
READ(5,933)ZRCPLA(1),I=3,6
963 FORMAT(7IX,E3.1,E16.9,13X,F3.1,13X,F3.1)
DO 17 J=1,5
   READ(5,964)ZRCPLA(J-1),I=1,4
   CONTINUE
   IF(J=1,11)
   RETURN
964 FORMAT(7IX,E3.1,E16.9,13X,F3.1)
DO 18 J=1,11
   READ(5,965)ZRCPLA(J-1),I=1,4
   RETURN
965 FORMAT(RX,E16.9,13X,F3.1)
DO 19 J=1,3
   READ(5,966)ZRCPLA(J-1),I=1,3
   RETURN
966 FORMAT(RX,E16.9,13X,F3.1)
DO 20 J=1,3
   READ(5,967)ZRCPLA(J-1),I=1,14
   RETURN
967 FORMAT(RX,54,18)
IF(IZACPIT(2),EQ.2)NR=14
IF(IZACPIT(2),EQ.3)NR=18
DO 21 J=1,NR
   READ(5,965)ZRCPLA(J-1),I=1,4
   RETURN
21 FORMAT(RX,54,18)
IF(IZACPIT(2),EQ.2)GOTO 540
IF(IZACPIT(2),EQ.3)GOTO 561
READ(5,97)ZRCPLA(167),ZRCPLA(164)
GOTO 350
READ(5,947)ZRCPLA(167),ZRCPLA(164)
SUBROUTINE INIT 74/74 OPT=1  FMT 4, R+44 11/03/72 20:07:46  PAGE 4

COTO 570
540 READ(5,947) ZRCPA(155), ZRCPA(156)
570 CONTINUE
175 467 FORMAT(2X18.9)
1  READ(5,500)
  READ(5,500)
  IF(LEACP(1, EQ.1) READ(5,500)

180 CCC
185 CCC READ STL AND TRAILFR
CCC READ(5,930) (ZSIL(I), I=1,6)
185 READ(5,500)
1  READ(5,961) (ZSIL(I), I=1,6)
  READ(5,961) (ZSIL(I), I=1,6)
  READ(5,961) (ZSIL(I), I=1,4)
960 FORMAT(82, 418)
964 FORMAT(82, 418)
190 RETURN
END

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
I INIT

VARIABLES 5N  TYPE  RELLOCATION
1516 I  INTEGER  ARRAY  ZDATA  1517 J  INTEGER  SYSTEM
1527 I  INTEGER  ARRAY  DREAL  1528 K  INTEGER  SYSTEM
1  INTEGER  ARRAY  DREAL  6  INTEGER  ARRAY  DREAL
36 TECTA INTEGER  ARRAY  ZDATA  TPLA  INTEGER  ARRAY  DREAL
30 TECI INTEGER  ARRAY  DREAL  TOSEL INTEGER  ARRAY  DREAL
0 ZACP REAL  ARRAY  ZDATA  ZACP REAL  ARRAY  ZDATA
424 ZECTA INTEGER  ARRAY  ZDATA  ZPLA INTEGER  ARRAY  ZDATA
402 ZSIL INTEGER  ARRAY  ZDATA  ZSIL INTEGER  ARRAY  ZDATA

FILE NAMES
TAPE  NO
STATEMENT LABELS  FMT  NO REFS  FMT  NO REFS
1057 3  FMT  NO REFS  0 10  340 15
10  17  0 17  0 17
9  20  0 20  0 20
147 51  247 51  315 52  315 52
637 500 FMT  24 501  21 502  21 502
703 503 FMT  714 505 FMT  26 510  26 510
33 511  35 512  45 513  45 513
0 514  127 514  1024 514 FMT  1024 514 FMT
0 517  1051 518 FMT  67 520  67 520
156 526  73 530  407 540  407 540
404 541  1242 541 FMT  1242 541 FMT
411 540  553 540  553 540  553 540
<table>
<thead>
<tr>
<th>STMT</th>
<th>LOCATION</th>
<th>LABELS</th>
<th>FTR</th>
<th>OPR</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>555</td>
<td>930</td>
<td>631</td>
<td>FMT</td>
<td>11/03/82</td>
<td>5</td>
</tr>
<tr>
<td>570</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>661</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>932</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>953</td>
<td>1101</td>
<td>FMT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>955</td>
<td>1132</td>
<td>FMT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>957</td>
<td>1157</td>
<td>FMT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>1227</td>
<td>FMT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1311</td>
<td>963</td>
<td>1324</td>
<td>FMT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>966</td>
<td></td>
<td>1340</td>
<td>FMT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1351</td>
<td>969</td>
<td>1352</td>
<td>FMT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>970</td>
<td></td>
<td>1372</td>
<td>FMT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1514</td>
<td>972</td>
<td>1375</td>
<td>FMT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**LOOP LABEL INDEX LENGTH PROPERTIES**

<table>
<thead>
<tr>
<th>LOOP</th>
<th>LABEL</th>
<th>INDEX</th>
<th>LOCATION</th>
<th>FORM-TO</th>
<th>LENGTH</th>
<th>INSTAC</th>
<th>PROPERTIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>S14</td>
<td>J</td>
<td>54</td>
<td>55</td>
<td>21B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td></td>
<td>I</td>
<td>55</td>
<td>55</td>
<td>21B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>10</td>
<td>J</td>
<td>63</td>
<td>64</td>
<td>12B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>305</td>
<td>5</td>
<td>J</td>
<td>64</td>
<td>64</td>
<td>12B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>134</td>
<td>517</td>
<td>J</td>
<td>72</td>
<td>73</td>
<td>12B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>137</td>
<td></td>
<td>I</td>
<td>73</td>
<td>73</td>
<td>12B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>173</td>
<td>43</td>
<td>J</td>
<td>90</td>
<td>92</td>
<td>12B</td>
<td>NOT INNER</td>
<td></td>
</tr>
<tr>
<td>206</td>
<td>15</td>
<td>J</td>
<td>93</td>
<td>118</td>
<td>135B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>217</td>
<td>951</td>
<td>M</td>
<td>97</td>
<td>98</td>
<td>20B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>222</td>
<td></td>
<td>I</td>
<td>98</td>
<td>98</td>
<td>20B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>244</td>
<td>954</td>
<td>M</td>
<td>104</td>
<td>105</td>
<td>21B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>247</td>
<td></td>
<td>I</td>
<td>105</td>
<td>105</td>
<td>21B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>272</td>
<td>956</td>
<td>M</td>
<td>110</td>
<td>111</td>
<td>21B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>275</td>
<td></td>
<td>I</td>
<td>111</td>
<td>111</td>
<td>21B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>316</td>
<td>457</td>
<td>M</td>
<td>115</td>
<td>116</td>
<td>12B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>321</td>
<td></td>
<td>J</td>
<td>116</td>
<td>116</td>
<td>12B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>355</td>
<td>16</td>
<td>J</td>
<td>129</td>
<td>129</td>
<td>12B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>360</td>
<td></td>
<td>I</td>
<td>129</td>
<td>129</td>
<td>12B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>427</td>
<td>17</td>
<td>J</td>
<td>149</td>
<td>150</td>
<td>20B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>425</td>
<td></td>
<td>I</td>
<td>150</td>
<td>150</td>
<td>20B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>443</td>
<td>18</td>
<td>J</td>
<td>152</td>
<td>153</td>
<td>20B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>446</td>
<td></td>
<td>I</td>
<td>153</td>
<td>153</td>
<td>20B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>471</td>
<td>19</td>
<td>J</td>
<td>159</td>
<td>160</td>
<td>20B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>474</td>
<td></td>
<td>I</td>
<td>160</td>
<td>160</td>
<td>20B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>522</td>
<td>20</td>
<td>J</td>
<td>165</td>
<td>166</td>
<td>12B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>525</td>
<td></td>
<td>I</td>
<td>166</td>
<td>166</td>
<td>12B</td>
<td>EXT REFS NOT INNER</td>
<td></td>
</tr>
<tr>
<td>600</td>
<td></td>
<td>I</td>
<td>166</td>
<td>166</td>
<td>7R</td>
<td>EXT REFS</td>
<td></td>
</tr>
<tr>
<td>613</td>
<td></td>
<td>I</td>
<td>167</td>
<td>167</td>
<td>7R</td>
<td>EXT REFS</td>
<td></td>
</tr>
</tbody>
</table>

**COMMON BLOCKS LENGTH**

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>ZDATA</th>
<th>FREIIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>317</td>
<td>36</td>
</tr>
</tbody>
</table>

**STATISTICS**

<table>
<thead>
<tr>
<th>PROGRAM LENGTH</th>
<th>COMMON LENGTH</th>
<th>CM LENGTH</th>
<th>CM USED</th>
</tr>
</thead>
<tbody>
<tr>
<td>15228</td>
<td>850</td>
<td>10710</td>
<td>568</td>
</tr>
</tbody>
</table>
SUBROUTINE MESSAGE  74/74  OPT=1  FTH 4.8564  11/03/52  20.07.44  PAGE 1

1     CCC
   CCC
   CCC
   CCC
   CCC
   CCC

SUBROUTINE MESSAGE(NON,PARN,NAME)
DIMENSION NAME(2)
INTEGER PARN
IF (PARN.EQ.21) GO TO 7
IF (PARN.EQ.10) GO TO 4
WRITE (6,400)
400 FORMAT ('1',5X,'FATAL ERROR XXPLIN')
GO TO 1
4 WRITE(6,402)
7 FORMAT ('1',5X,'FATAL ERROR CIAS')
GOTO 0
2 WRITE (6,401)
401 FORMAT ('1',5X,'FATAL ERROR CHKATS')
3 RETURN
20 END

SYMBOLIC REFERENCES MAP (K=1)
ENTRY POINTS
3 MESSAGE

VARIABLES  SK  TYPE  RELLOCATION
0 NAME  INTEGER  ARRAY  F.P.
0 PARN  INTEGER

FILE NAMES  MODE  TAPE  FMT

STATEMENT LABELS
16 2
24 401  FMT

STATISTICS
PROGRAM LENGTH  008  40
&280000 ON USED
SUBROUTINE SSPLIN(N1,X1,PD,EX,ET,EP,EP2,MCORE,ISNC)
LOGICAL L,LY,LO,X,IXT
DIMENSION C((1),X1(1),Y1(1),NAME(2))
REAL N
DATA NAME/H55SPL46,N5IN /
LONE = .TRUE.
LX = .TRUE.
LY = .TRUE.
IXT = .FALSE.
IXD = .FALSE.
IF(ET.0.0.0.0.XT.0.0.0.0.0.LONE) LONE = .FALSE.
IF(ET.0.0.0.0.0.XT.0.0.0.0.LY) LY = .FALSE.
IF(ET.0.0.0.0.0.XT.0.0.0.0.0.LY) LY = .FALSE.
N = N1
IF(LONE) N=N+1
IF(LX) N=N+1
IF(LY) N=N+1
EX = FLOAT(X1)
ET = FLOAT(ET)
IF(ET.EQ.0.0.0.0.XT.0.0.0.0.0.1) IXT = .TRUE.
IF(ET.EQ.0.0.0.0.0.XT.0.0.0.0.1) IXD = .TRUE.
MB = MD*(1+KD)
CORE NEEDED
A = C INVERS
NEEDED = N*N1 + 3*N
IF(IXT) NEEDED = NEEDED + N*N + N
C
A OR B
IF(NH1.IF) NEEDED = NEEDED + N*N + MAX(N*N,N*N)
IF(MF1.EQ.MC) CALL WESAGE(-8,0,NAME)
IS = N*N1 = 3*N - 1
IC = 1
IF ET = 1 COMPUTE B THEN A THEN C IN A SPACE
IF ET = 0 COMPUTE C THEN A THEN B IN A SPACE
C
WT = 2*N1
IF(.NOT.XT) GO TO 65
GO TO 9A
C
COMPUTE TO A MATRIX
C
A = 1
300
C
K = 1
C
IF K = 1
C
GO TO 30
C
END
C
COMPUTE TO A MATRIX
SUBROUTINE SPLIN

50 C(I) = 0.0
     I = 1
     IF
     DO 60 I = 1, N, 2
     F = F(I)
     JJ = 1/2
     DO 70 J = 1, N, 2
     E = E(J)
     JJ = JJ / 1
     SUM = 0.0
     VM = (X(I) - X(J)) ** 2
     VP = (X(I) + X(J)) ** 2
     VM = (X(I+1) - X(J+1)) ** 2
     VP = (X(I+1) + X(J+1)) ** 2
     T1 = VM * VM
     T2 = VP * VP
     T3 = X(I) - X(J)
     T4 = X(I+1) - X(J+1)
     IF(T1.NE.0.0) SUM = SUM + (T2 * ALOC(T2) * EX)
     IF(T3.NE.0.0) SUM = SUM + (T1 * ALOC(T1) * ET)
     IF(T4.NE.0.0) SUM = SUM + (T4 * ALOC(T4) * EX*ET)
     IF(J.EQ.1) GO TO 10
     C(Y) = SUM

C SYMFTY TERM

55 XY = Y + (N-1) * (JJ-1)
     G(XY) = SUM
     GO TO 20

10 C(Y) = SUM + D2
     FJ = E

20 CONTINUE
     IMP = 0
     IF(.NOT.LONE) GO TO 30

30 IMP = IMP + 1
     G(IMP) = 1.0
     G(IMP+1) = 1.0
     IF(.NOT.LX) GO TO 40

40 IMP = IMP + 1
     G(IMP) = Y(I)
     IF(.NOT.LY) GO TO 50

50 IMP = IMP + 1
     G(IMP) = X(J)
     IF(J+1 .GE. I) GO TO 10

100 IF = 1
     II = 1

CONTINUE

C 140 Parem, FOR A-1 C OR A+1 B

110 IC = 1
     IMP = IC
     IF = IC
     IC = IC + 1
     IF = IC
     IC = 1
GO TO 70

MATRIX COLUMN STORED

IC = IK
IF = IC
DO 40 J = 1, N
DO 40 I = 1, M
IF = IC
C(FC) = 0.0
IF(FL, FC) C(FC) = 1.0
CONTINUE
CONTINUE
IF(IF) GO TO 170
IF = IC
IA = IC
GO TO 1

MATRIX COLUMN STORED

NB = NA
IF = IB + 1
GO TO 170
IB = IA
IB = IB + 1
DO 160 J = 1, MR, 2
DO 170 I = 1, MT, 2
IF = IB + 1
ALT1 = 0.0
ALT2 = 0.0
ALT3 = 0.0
ALT4 = 0.0
XH = XH(J) - XH(I)
XP = XH(I) + XH(J)
YP = YH(I) + YH(J)
T1 = YHMY + XHMY
T2 = XPMY + T1MY
T3 = XPMY + T2MY
T4 = XPMY + T2MP
IF(TI, MF, 0.0) ALT1 = ALOC(TI)
IF(TJ, MF, 0.0) ALT2 = ALOC(TJ)
IF(TK, MF, 0.0) ALT3 = ALOC(TK)
IF(TL, MF, 0.0) ALT4 = ALOC(TL)
C(FC) = T1*ALT1 + T2*ALT2 + T3*ALT3 + T4*ALT4 + XP + YP
IF (.NOT. IFD) GO TO 120
IF = IB + N
C(FC) = 2.0*XP*YP*(1.0 + ALT1) + XP*(1.0 + ALT2)*YP + YP*(1.0 + ALT3)*XP + XP*(1.0 + ALT4)*YP
CONTINUE
C = 0
IF (.NOT. IFD) GO TO 130
IF = IF + 1
IFD = IFD + 1.0
IF(IF) / (IF + IFD) = 0.0
IFD = IF + IFD
GO TO 130
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A
SUBROUTINE SSPLIN 74/74 OPT-1                                                     FTN 4.8+5AA 11/03/82 20.07.84  PAGE 6

IRE = IHP + 1
C(IRE+1) = X(Y(J))
IF(IKH) C(IRE+IHP+1) = 1.0
175 140 IF(.NOT.LY) GO TO 150
: IHP = IHP + 1
C(IRE+IHP) = X(Y(J+1))
IF(IKH) C(IRE+IHP+1) = 0.0
150 IRA = IR + IHP + N

180 CONTINUE
IF(.NOT.LY) GO TO 180
IA = IR
NC = NA
GO TO 1

100 CONTINUE
C MWANTS ROW STORED SO INVERT ROWS AND COLUMNS AND INVERT
C MULTIPLICATION ORDER
C
190 C CALL CMATS CT R
CALL CMATS(C(MPI),NA,N,O,C(F),MI,N1,G(IG))
GO TO 1000

180 CONTINUE
C CALL CMATS CT C
C
1000 RETURN
END

SYMBOLIC REFERENCE MAP (F-1)

ENTRY POINTS
1 SSPLIN

<table>
<thead>
<tr>
<th>VARIABLES</th>
<th>TN TYPE</th>
<th>RELLOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>610 ALT1</td>
<td>REAL</td>
<td>631 ALTA</td>
</tr>
<tr>
<td>612 ALT2</td>
<td>REAL</td>
<td>640 ALTA</td>
</tr>
<tr>
<td>670 BTT</td>
<td>REAL</td>
<td>0 DZ</td>
</tr>
<tr>
<td>572 FA</td>
<td>REAL</td>
<td>573 ET</td>
</tr>
<tr>
<td>602 IA</td>
<td>INTEGER</td>
<td>605 I</td>
</tr>
<tr>
<td>625 IC</td>
<td>INTEGER</td>
<td>626 I</td>
</tr>
<tr>
<td>603 IJ</td>
<td>INTEGER</td>
<td>604 I</td>
</tr>
<tr>
<td>561 1JK</td>
<td>LOGICAL</td>
<td>566 1JK</td>
</tr>
<tr>
<td>622 IMR</td>
<td>INTEGER</td>
<td>627 1M</td>
</tr>
<tr>
<td>574 INC</td>
<td>INTEGER</td>
<td>607 J</td>
</tr>
<tr>
<td>606 IJ</td>
<td>INTEGER</td>
<td>601 K</td>
</tr>
<tr>
<td>607 IF</td>
<td>INTEGER</td>
<td>621 FZ</td>
</tr>
<tr>
<td>604 IF</td>
<td>INTEGER</td>
<td>0 FX</td>
</tr>
<tr>
<td>665 LOG</td>
<td>LOGICAL</td>
<td>666 LOG</td>
</tr>
<tr>
<td>694 LY</td>
<td>LOGICAL</td>
<td>695 LY</td>
</tr>
<tr>
<td>623 MV</td>
<td>INTEGER</td>
<td>624 MV</td>
</tr>
<tr>
<td>574 1NRA</td>
<td>INTEGER</td>
<td>654 1NRA</td>
</tr>
<tr>
<td>574 1N</td>
<td>INTEGER</td>
<td>654 1N</td>
</tr>
<tr>
<td>625 1K</td>
<td>INTEGER</td>
<td>626 1K</td>
</tr>
<tr>
<td>654 LOG</td>
<td>LOGICAL</td>
<td>655 LOG</td>
</tr>
<tr>
<td>695 LY</td>
<td>LOGICAL</td>
<td>696 LY</td>
</tr>
<tr>
<td>624 MV</td>
<td>INTEGER</td>
<td>625 1MV</td>
</tr>
<tr>
<td>654 1NRA</td>
<td>INTEGER</td>
<td>655 1NRA</td>
</tr>
<tr>
<td>626 1K</td>
<td>INTEGER</td>
<td>626 1K</td>
</tr>
</tbody>
</table>
### SUBLISTING SPLITIN

<table>
<thead>
<tr>
<th>VARIABLES</th>
<th>SN</th>
<th>TYPE</th>
<th>RELOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>1</td>
<td>INTEGER F.P.</td>
<td>627</td>
</tr>
<tr>
<td>600</td>
<td>1</td>
<td>INTEGER</td>
<td>610</td>
</tr>
<tr>
<td>113</td>
<td>1</td>
<td>REAL</td>
<td>616</td>
</tr>
<tr>
<td>217</td>
<td>1</td>
<td>REAL</td>
<td>620</td>
</tr>
<tr>
<td>211</td>
<td>1</td>
<td>REAL</td>
<td>612</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>REAL</td>
<td>0</td>
</tr>
</tbody>
</table>

### EXTERNALS

<table>
<thead>
<tr>
<th>PROCEDURE</th>
<th>TYPE</th>
<th>ARCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALOC</td>
<td>REAL</td>
<td>1 LIBRARY</td>
</tr>
<tr>
<td>INVERS</td>
<td></td>
<td>8 MESSAC</td>
</tr>
</tbody>
</table>

### INLINE FUNCTIONS

<table>
<thead>
<tr>
<th>PROCEDURE</th>
<th>TYPE</th>
<th>ARCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOA</td>
<td>REAL</td>
<td>1 INTRIN</td>
</tr>
</tbody>
</table>

### STATEMENT LABELS

<table>
<thead>
<tr>
<th>STATEMENT</th>
<th>LABEL</th>
<th>FROM</th>
<th>TO</th>
</tr>
</thead>
<tbody>
<tr>
<td>104</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>216</td>
<td>20</td>
<td>227</td>
<td>30</td>
</tr>
<tr>
<td>245</td>
<td>50</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>275</td>
<td>70</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>320</td>
<td>95</td>
<td>324</td>
<td>100</td>
</tr>
<tr>
<td>425</td>
<td>120</td>
<td>437</td>
<td>130</td>
</tr>
<tr>
<td>457</td>
<td>150</td>
<td>0</td>
<td>160</td>
</tr>
<tr>
<td>514</td>
<td>1000</td>
<td>0</td>
<td>1000</td>
</tr>
</tbody>
</table>

### PHYSICAL MAP

<table>
<thead>
<tr>
<th>LABEL</th>
<th>INDEX</th>
<th>FROM</th>
<th>TO</th>
<th>LENGTH</th>
<th>PROPERTIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>114</td>
<td>5</td>
<td>57</td>
<td>58</td>
<td>2B</td>
<td>INSTACK</td>
</tr>
<tr>
<td>121</td>
<td>60</td>
<td>61</td>
<td>106</td>
<td>132B</td>
<td>EXT REFS, NOT INNER</td>
</tr>
<tr>
<td>125</td>
<td>20</td>
<td>64</td>
<td>90</td>
<td>74B</td>
<td>EXT REFS</td>
</tr>
<tr>
<td>276</td>
<td>40</td>
<td>121</td>
<td>127</td>
<td>16B</td>
<td>NOT INNER</td>
</tr>
<tr>
<td>303</td>
<td>40</td>
<td>122</td>
<td>126</td>
<td>5B</td>
<td>INSTACK</td>
</tr>
<tr>
<td>312</td>
<td>160</td>
<td>141</td>
<td>180</td>
<td>133B</td>
<td>EXT REFS, NOT INNER</td>
</tr>
<tr>
<td>333</td>
<td>120</td>
<td>142</td>
<td>165</td>
<td>75B</td>
<td>EXT REFS</td>
</tr>
</tbody>
</table>

### STATISTICS

<table>
<thead>
<tr>
<th>PROGRAM LENGTH</th>
<th>STORED</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>70454</td>
<td>452</td>
<td></td>
</tr>
</tbody>
</table>
SUBROUTINE CMATS (A, IROWS, ICOLA, MTA, B, IROWS, ICOLB, MTR, C)

C******
CMATS - GENERAL MATRIX MULTIPLY
AND
TRANSPOSE
SINGLE PRECISION VERSION
C
PERFORMS
A + B = C MTA=0 MTR=0
A + B TRANSPOSE = C 0 1
A TRANSPOSE + B = C 1 0
A TRANSPOSE + B TRANSPOSE = C 0 1
C******
A = IS A MATRIX (ROWS) ROWS BY (COLA) COLUMNS
B = IS A MATRIX (ROWS) ROWS BY (COLB) COLUMNS
A, B AND C ARE STORED BY ROWS (EXAMPLE)
C
MATRX STORED
A 1 2 A 1
3 4
C 5 6
C
C******

IF MTA .LT. 0, C IS NOT ZEROED OUT. HENCE THE ROUTINE, IN THIS
CASE, COMPUTES A * B + D = C WHERE THE MATRIX D HAS BEEN
STORED ROW-WISE AT C BY THE LINC PROGRAM. IF MTA = -1, A
IS TRANSPOSED. IF MTA = -2, A IS NOT TRANSPOSED. MTR IS
DEFINED AS ABOVE AND IS INDEPENDENT OF MTA.

INTEGER IROWS, ICOLA, IROWS, ICOLB

DIMENSION A(1),B(1),C(1),IPARM(2)

IROWS = IROWS
ICOLA = ICOLA
IROWS = IROWS
ICOLB = ICOLB
MTA = IAPS(MTA)
IF (MTA.EQ.(-3)) MTA = 0
IF (MTA .EQ. 0) AND (MTB .EQ. 0) IF (ICOLA = IROWS) RO, 5, 40
IF (MTA .EQ. 0) AND (MTB .EQ. 1) IF (ICOLA = IROWS) RO, 5, 40
IF (MTA .EQ. 0) AND (MTB .EQ. 1) IF (ICOLA = ICOLB) RO, 5, 40
IF (MTA .EQ. 1) AND (MTB .EQ. 1) IF (ICOLA = ICOLB) RO, 5, 40
SUBROUTINE CMATS  74/74  OPT=1

5 IF (MTA .LE. 1) GO TO 10
   ILIM = ROWA
   RLI = COLA
   INC = COLA
   INCX = 1
   GO TO 20
10 ILIM = COLA
65 KLI = ROWA
   INC = 1
   INCX = COLA
   IF (MTA.EQ.1) GO TO 30
10 JLI = COLB
   INCJ = 1
   INCXB = COLB
   GO TO 40
30 JLI = ROWB
   INCJ = COLB
   INCXB = 1
   IF (MTA .LT. 0) GO TO 47
   ILIM = JLIM = KLIM
   DO 45 1 = 1, ILIM
45 C(I) = 0.0
47 1 = 0
50 1 = 1 + 1
   IFIX = 1*INC - COLA
   J = 0
85 J = J + 1
   IJ = 1 + 1
   IA = IFIX
   JBL = J + INCJ - COLB
   F = 0
90 70 X = K + 1
   IA = IA + INCX
   JBL = JB + INCX
   C(IJ) = C(IJ) + A(IA) * B(JB)
   IF (K .LT. KLIM) GO TO 70
   IF (J .LT. JBL) GO TO 60
   IF (1 .LT. ILIM) GO TO 50
   RETURN
80 IFARM(1) = MTA
   IFARM(2) = MTA
100 CALL MESSAGE (-30, 21, IFARM(1))
   RETURN
END

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 CMATS
SUBROUTINE INVERS
74/74 OPT-1

**SUBROUTINE INVERS(MDIM,A,N,B,N,DETERM,ISING,INDEX)**

**** INVERSE OR LINEAR EQUATIONS SOLVER ****************************

**********************************************************************

**MDIM IS THE ACTUAL SIZE OF A IN CALLING PROGRAM.**
EC. AC(MDIM,MDIM)
**A IS SQUARE MATRIX TO BE INVERTED.**
**N IS SIZE OF UPPER LEFT PORTION BEING INVERTED.**
**M IS THE NUMBER OF COLUMNS OF CONSTANTS**
**DETERM RETURNS THE VALUE OF DETERMINANT IF NON-SINGULAR**
**ISING RETURNS 0 IF MATRIX A(N,N) IS SINGULAR**
**INVERSE RETURNS IN A SOLUTION VECTORS RETURN IN B**
**INDEX IS WORKING STORAGE (N,3)**

**********************************************************************

**DIMENSION A(MDIM,1), B(MDIM,1), INDEX(N,3)**
**EQUIVALENCE (IROW,ICOL), (ICOL,JCOL), (AMAX, T, SWAP)**

**INITIALIZE**

DETERM = 1.0E0
DO 10 J = 1, N
10 INDEX(J,3) = 0
DO 130 J = 1, N

**SEARCH FOR PIVOT**

AMAX = 0.0E0
DO 40 J = 1, N
IF(INDEX(J,3) .EQ. 1) GO TO 40
DO 30 K = 1, N
30 CONTINUE

IF(INDEX(K,3) = 1) 20, 30, 190
20 IF(ABS( A(J,K) ) .LE. AMAX) GO TO 30
IROW = J
ICOL = K
AMAX = ABS( A(J,K) )
30 CONTINUE

INDEX(ICOL,3) = INDEX(ICOL,3) + 1
INDEX(1,1) = IROW
INDEX(1,2) = ICOL

**INTERCHANGE ROWS TO PUT PIVOT ELEMENT ON DIAGONAL**

IF (IROW .EQ. ICOL) GO TO 70
DETERM = -DETERM
DO 50 L = 1, N
SWAP = A(IROW,L)
A(IROW,L) = A(ICOL,L)
50 CONTINUE

20 70 190
SUBROUTINE INVERS 74/74  OPT-1

50 A(ICOLUN,M) = SWAP
   IF(N .LE. 0) GO TO 70
   DO 60 L=1,M
     SWAP = B(IROW,L)
     B(IROW,L) = A(ICOLUN,L)
   60 B(ICOLUN,L) = SWAP

C C C C C C

65 DIVIDE PIVOT ROW BY PIVOT ELEMENT
C C C C C
70 PIVOT = A(ICOLUN,ICOLUN)
    DETERM = DETERM * PIVOT
    A(ICOLUN,ICOLUN) = 1.0D0
    DO 80 L=1,M
       80 A(ICOLUN,L) = A(ICOLUN,L) / PIVOT
    IF(N .LE. 0) GO TO 100
    DO 90 L=1,M
       90 B(ICOLUN,L) = B(ICOLUN,L) / PIVOT

C C C C C

75 REDUCE NOW PIVOT ROWS
C C C C C
100 DO 130 L=1,M
    IF(L .EQ. ICOLUN) GO TO 130
       T = A(L,ICOLUN)
       A(L,ICOLUN) = 0.0D0
       DO 110 L=1,M
          110 A(L,L) = A(L,L) - A(ICOLUN,L) * T
       IF(N .LE. 0) GO TO 130
       DO 120 L=1,M
          120 B(L,L) = B(L,L) - B(ICOLUN,L) * T
     130 CONTINUE

C C C C C

90 INTERCHANGE COLUMNS
C C C C C
130 DO 150 I=1,M
     L = M + 1 - I
     IF(INDEX(L,1) .EQ. INDEX(L,2)) GO TO 150
     JROW = INDEX(L,1)
     JCOLUMN = INDEX(L,2)
     DO 140 K=1,M
        SWAP = A(K,JROW)
        A(K,JROW) = A(K,JCOLUMN)
        A(K,JCOLUMN) = SWAP
     140 CONTINUE
     150 CONTINUE
     DO 170 K=1,M
        IF(INDEX(K,3) .EQ. 1) GO TO 160
        ISING = -2
        160 CONTINUE
     170 CONTINUE
     ISING = +1
     180 RETURN
     190 ISING = -2
     RETURN
     END
## Symbolic Reference Map (R=1)

### Entry Points

| INVERS | 3 |

### Variables

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>ARRAY</th>
<th>F.P.</th>
<th>ANAX</th>
<th>REAL</th>
<th>REAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>260</td>
<td>T</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>K</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>L</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>263</td>
<td>PIVOT</td>
<td>REAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>267</td>
<td>T</td>
<td>REAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>260</td>
<td>N</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>B</td>
<td>REAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>A</td>
<td>REAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>D</td>
<td>REAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>267</td>
<td>A</td>
<td>REAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>I</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>I</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>I</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>J</td>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Inline Functions

| TYPE | ARCS | REAL | 1 INTRIN |

### Statement Labels

| 0   | 30  | 0  | 20  | INACTIVE | 46 | 30 |
| 115 | 70  | 0  | 80  |          | 0  | 40 |
| 146 | 100 | 0  | 110 |          | 0  | 120|
| 205 | 130 | 0  | 140 |          | 234| 150|
| 263 | 160 | 0  | 170 |          | 235| 180|
| 252 | 190 | 0  |     |          |    |    |

### Loops

<table>
<thead>
<tr>
<th>LABEL</th>
<th>INDEX</th>
<th>FROM-TO</th>
<th>LENGTH</th>
<th>PROPERTIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>J</td>
<td>30 31</td>
<td>28</td>
<td>INSTACK</td>
</tr>
<tr>
<td>130</td>
<td>J</td>
<td>32 87</td>
<td>165B</td>
<td>EXITS</td>
</tr>
<tr>
<td>130</td>
<td>J</td>
<td>32 87</td>
<td>165B</td>
<td>NOT INNER</td>
</tr>
<tr>
<td>130</td>
<td>J</td>
<td>32 87</td>
<td>165B</td>
<td>NOT INNER</td>
</tr>
<tr>
<td>130</td>
<td>J</td>
<td>32 87</td>
<td>165B</td>
<td>NOT INNER</td>
</tr>
<tr>
<td>130</td>
<td>J</td>
<td>32 87</td>
<td>165B</td>
<td>NOT INNER</td>
</tr>
<tr>
<td>130</td>
<td>J</td>
<td>32 87</td>
<td>165B</td>
<td>NOT INNER</td>
</tr>
<tr>
<td>130</td>
<td>J</td>
<td>32 87</td>
<td>165B</td>
<td>NOT INNER</td>
</tr>
</tbody>
</table>

### Statistics

<table>
<thead>
<tr>
<th>PROGRAM LENGTH</th>
<th>3408</th>
<th>224</th>
</tr>
</thead>
<tbody>
<tr>
<td>3200000 CM USED</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FUNCTION CORSZ 74/74 OPT-1

1
CCC
CCC
CCC
CCC

5
CCC
INTEGER FUNCTION CORSZ(Z,A)
COMMON/ZSIZE/ZSIZE
CCC DEFINES AMOUNT OF OPEN CORE SUPPLIED
ZSIZE/ZSIZE
RETURN
END

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 CORSZ

VARIABLES IN TYPE RELOCATION
0 A REAL *UNUSED F.P. 10 CORSZ INTEGER *UNUSED F.P.
0 SIZE REAL ZSIZE

COMMON BLOCKS LENGTH
2 SIZE 1

STATISTICS
PROGRAM LENGTH 118 9
CH LABELS COMMON LENGTH 18 1
52000B CH USED
SUBROUTINE COPEN

<table>
<thead>
<tr>
<th>ENTRY POINTS</th>
<th>3 COPEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARIABLES</td>
<td>DB TYPE</td>
</tr>
<tr>
<td>BUFFER</td>
<td>REAL</td>
</tr>
<tr>
<td>NUP</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DB</td>
<td>INTEGER</td>
</tr>
<tr>
<td>FILE NAMES</td>
<td>MODE</td>
</tr>
<tr>
<td>STATEMENT LABELS</td>
<td>14 910</td>
</tr>
<tr>
<td>STATISTICS</td>
<td>PROGRAM LENGTH</td>
</tr>
<tr>
<td></td>
<td>52000R CM USED</td>
</tr>
</tbody>
</table>

SYMBOLIC REFERENCE MAP (R=1)

<table>
<thead>
<tr>
<th>INTEGER</th>
<th>DB</th>
<th>FORMAT(1X,&quot;OPENED DATABLOCK:&quot;,I5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETURN</td>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>
SUBROUTINE CLOSE

1      CCC
      CCC
      CCC
      CCC
      CCC

5      CCC:
      SUBROUTINE CLOSE(DR,NUM)
      INTEGER DR
      WRITE(6,920)DR
      920    FORMAT(IX, 'CLOSED DATABASE', 1X)
      RETURN
      END

SYMBOLIC REFERENCE MAP (M=1)

ENTRY POINTS
3     CLOSE

VARIABLES  SN  TYPE  RELOCATION  F.P.  0  NUM  INTEGER  #UNUSED  F.P.
0      DB   INTEGER

FILE NAMES  MODE
TAPF6   PMT

STATEMENT LABELS
14  920  PMT

STATISTICS
PROGRAM LENGTH
520000  CH  USED
208  16
SUBROUTINE PACK (\texttt{IN}, DB, TRAILER)
COMMON/PACK/TYPIN, TYPOUT, IR\texttt{OW}, NROW, INCR
INTEGER TRAILER(7), DB
DIMENSION ZH(1)
WRITE(6,100) DB
100 FORMAT(1X, "DATABASE=", I5)
MWDPLN=10
J=0
LIMIT=NROW/NWDPLN
IF(LIMIT.LE.0)GOTO 200
DO 300 1=1, LIMIT
\indent WRITE(6,110)(ZH(J+K), K=1, NWDPLN)
\indent \indent J=J+NWDPLN
300 CONTINUE
200 LIMIT=NROW-LIMIT*NWDPLN
IF(LIMIT.LE.0)GOTO 400
\indent \indent WRITE(6,110)(ZH(J+1), J=1, LIMIT)
400 RETURN
110 FORMAT(10(IX, E12.5))
25 \texttt{END}

\textbf{SYMBOLIC REFERENCE MAP (R=1)}

\textbf{ENTRY POINTS}
3 \texttt{PACK}

\textbf{VARIABLES}
\begin{tabular}{|l|c|c|c|}
\hline
\textbf{SN} & \textbf{TYPE} & \textbf{RELOCATION} & \textbf{PACK} \\
\hline
0 & DB & INTEGER & F.P. \\
1 & INCR & INTEGER & PACKX \\
2 & IROW & INTEGER & PACKX \\
3 & TYPIN & INTEGER & PACKX \\
4 & ZH & REAL & ARRAY \\
5 & TN & REAL & ARRAY \\
\hline
\end{tabular}

\textbf{FILE NAMES}
\texttt{TAPFA}

\textbf{STATEMENT LABELS}
\begin{tabular}{|l|l|l|l|l|}
\hline
\textbf{SM} & \textbf{SN} & \textbf{FMT} & \textbf{SN} & \textbf{FMT} \\
\hline
67 & 100 & FMT & 110 & 110 & FMT \\
\hline
\end{tabular}

\textbf{LOCUS LAPP INDEX FROM TO LENGTH PROPERTIES}
\begin{tabular}{|l|l|l|l|l|}
\hline
\textbf{SM} & \textbf{SN} & \textbf{FMT} & \textbf{SN} & \textbf{FMT} \\
\hline
1 & 1 & 16 19 & 70B & EXT REFS NOT INNER \\
2 & 1 & 17 17 & 10B & EXT REFS \\
3 & 1 & 22 22 & 10B & EXT REFS \\
\hline
\end{tabular}
### Subroutine RTRAIL

```fortran
SUBROUTINE RTRAIL( TRAIL )
COMMON/STRAIL/TACP(6), TBCPA(6), TTSSEL(6), TCPLA(6), TSIL(6), TECTA(6)
INTEGER TRAIL(7), TACP, TBCPA, TTSSEL, TCPLA, TSIL, TECTA
COTO(100, 200, 300, 400, 500, 600) TRAIL(1)-100
10      DO 10 I = 1, 6
10      TRAIL(I+1) = TACP(I)
      RETURN
20      DO 20 I + = 1, 6
20      TRAIL(I+1) = TTSSEL(I)
      RETURN
30      DO 30 I = 1, 6
30      TRAIL(I+1) = TECTA(I)
      RETURN
40      DO 40 I = 1, 6
40      TRAIL(I+1) = TCPLA(I)
      RETURN
50      DO 50 I = 1, 6
50      TRAIL(I+1) = TBCPA(I)
      RETURN
60      DO 60 I = 1, 6
60      TRAIL(I+1) = TSIL(I)
      RETURN
   END
```

### Symbolic Reference Map (E=1)

<table>
<thead>
<tr>
<th>ENTRY POINTS</th>
<th>ROUTE</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>VARIABLES</th>
<th>SN</th>
<th>TYPE</th>
<th>RELLOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>I</td>
<td>INTEGER</td>
<td>O TACP INTEGER ARRAY DTRAIL</td>
</tr>
<tr>
<td>6</td>
<td>N</td>
<td>TBCPA</td>
<td>30 TECTA INTEGER ARRAY DTRAIL</td>
</tr>
<tr>
<td>22</td>
<td>N</td>
<td>TCPLA</td>
<td>0 TRAIL INTEGER ARRAY F.P.</td>
</tr>
<tr>
<td>30</td>
<td>N</td>
<td>TSIL</td>
<td>14 TTSSEL INTEGER ARRAY DTRAIL</td>
</tr>
</tbody>
</table>

### Statement Labels

<table>
<thead>
<tr>
<th>LABEL</th>
<th>SN</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>01</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

### Load Labels

<table>
<thead>
<tr>
<th>LABEL</th>
<th>INDEX</th>
<th>LENGTH</th>
<th>PROPERTIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>73</td>
<td>1</td>
<td>2</td>
<td>INSTACK</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>2</td>
<td>INSTACK</td>
</tr>
<tr>
<td>41</td>
<td>1</td>
<td>2</td>
<td>INSTACK</td>
</tr>
<tr>
<td>75</td>
<td>1</td>
<td>2</td>
<td>INSTACK</td>
</tr>
<tr>
<td>77</td>
<td>1</td>
<td>2</td>
<td>INSTACK</td>
</tr>
</tbody>
</table>
PROGRAM WRTEL

ENTRY POINTS
3 WRTEL

VARIABLES
36 I INTEGER

FILE NAMS
MODF
TAPP

STATEMENT LABELS
74 100 FMT

LABEL INDEX FROM-TO LENGTH PROPERTIES
11 20F I 10 11 10B EXIT REFS

STATISTICS
PROGRAM LENGTH
41B 33

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 WRTEL

VARIABLES
36 I INTEGER

FILE NAMS
MODF
TAPP

STATEMENT LABELS
74 100 FMT

LABEL INDEX FROM-TO LENGTH PROPERTIES
11 20F I 10 11 10B EXIT REFS

STATISTICS
PROGRAM LENGTH
41B 33
**SUBROUTINE READ**

```fortran
74/74  OPT=1

IF(JECTA.LE.4)RETURN
JECTA=4
JECTA=64
RETURN

CCC
CCC READ FROM GPLA
CCC
65 DO 240 I=1,NW
   IF(ICPLA+I-1.GT.41)GOTO 360
240 ZK(I)=ICPLA(ICPLA+I-1)
   I=ICPLA+I
   ZK(I)=ZK(I-1)
   ICPLA=ICPLA+NW
   IF(FLAG.NE.1)RETURN
   ICPLA=42
   RETURN
360 M=I-1
   ICPLA=42
   RETURN

CCC
CCC READ FROM BCPLA
CCC
80 DO 250 I=1,NW
   IF(IBCPLA+I-1.GT.164)GOTO 380
250 ZK(I)=IBCPLA(IBCPLA+I-1)
   I=IBCPLA+IBCPLA+NW
   M=I
   IF(FLAG.NE.1)RETURN
   IBCPLA=165
   RETURN
380 M=I-1
   IBCPLA=165
   RETURN

CCC
CCC READ FROM SIL
CCC
90 DO 260 I=1,NW
   IF(ISIL=I-1.GT.18)GOTO 390
260 ZK(I)=ISIL(ISIL+I-1)
   ISIL=ISIL+ISIL+NW
   M=I
   IF(FLAG.NE.1)RETURN
   ISIL=19
   RETURN
390 M=I-1
   ISIL=19
   RETURN
END
```

**SYMBOLIC REFERENCE MAP**

(R=1)
<table>
<thead>
<tr>
<th>ENTRY POINTS</th>
<th>3 READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARIABLES SN TYPE</td>
<td>RELLOCATION</td>
</tr>
<tr>
<td>0 DB INTEGER</td>
<td>F.P.</td>
</tr>
<tr>
<td>211 INTEGER</td>
<td>212 SACP INTEGER</td>
</tr>
<tr>
<td>217 INTEGER</td>
<td>214 JECTA INTEGER</td>
</tr>
<tr>
<td>216 INTEGER</td>
<td>220 JSIL INTEGER</td>
</tr>
<tr>
<td>213 INTEGER</td>
<td>215 JECTA INTEGER</td>
</tr>
<tr>
<td>0 M INTEGER</td>
<td>F.P.</td>
</tr>
<tr>
<td>0 ZACP REAL ARRAY ZDATA</td>
<td>61 ZBCPA REAL ARRAY ZDATA</td>
</tr>
<tr>
<td>424 ZECTA INTEGER ARRAY ZDATA</td>
<td>221 ZCPL INTEGER ARRAY ZDATA</td>
</tr>
<tr>
<td>0 ZM REAL ARRAY F.P.</td>
<td>222 ZMI INTEGER ARRAY</td>
</tr>
<tr>
<td>222 ZHR REAL ARRAY</td>
<td>402 ZSIL INTEGER ARRAY</td>
</tr>
<tr>
<td>325 ZTOSEL INTEGER ARRAY ZDATA</td>
<td></td>
</tr>
<tr>
<td>STATEMENT LABELS</td>
<td></td>
</tr>
<tr>
<td>21 10</td>
<td>44 20</td>
</tr>
<tr>
<td>117 40</td>
<td>143 50</td>
</tr>
<tr>
<td>0 200</td>
<td>0 210</td>
</tr>
<tr>
<td>0 240</td>
<td>0 250</td>
</tr>
<tr>
<td>40 300</td>
<td>64 320</td>
</tr>
<tr>
<td>137 360</td>
<td>162 380</td>
</tr>
<tr>
<td>LOOPS LABEL INDEX FROM-TO LENGTH PROPERTIES</td>
<td></td>
</tr>
<tr>
<td>22 200</td>
<td>1</td>
</tr>
<tr>
<td>45 210</td>
<td>1</td>
</tr>
<tr>
<td>71 220</td>
<td>1</td>
</tr>
<tr>
<td>120 240</td>
<td>1</td>
</tr>
<tr>
<td>144 250</td>
<td>1</td>
</tr>
<tr>
<td>167 260</td>
<td>1</td>
</tr>
<tr>
<td>COMMON BLOCKS LENGTH</td>
<td></td>
</tr>
<tr>
<td>ZDATA 332</td>
<td></td>
</tr>
<tr>
<td>STATISTICS</td>
<td></td>
</tr>
<tr>
<td>PROGRAM LENGTH</td>
<td>3108 200</td>
</tr>
<tr>
<td>CM Labeled Common Length</td>
<td>10248 532</td>
</tr>
<tr>
<td>52000P CM Used</td>
<td></td>
</tr>
<tr>
<td>DT1</td>
<td>ACT</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>*AC</td>
<td>OENDREC</td>
</tr>
<tr>
<td>DT1</td>
<td>ACT</td>
</tr>
<tr>
<td>*AC</td>
<td>4</td>
</tr>
<tr>
<td>*AC</td>
<td>5</td>
</tr>
<tr>
<td>*AC</td>
<td>6</td>
</tr>
<tr>
<td>*AC</td>
<td>7</td>
</tr>
<tr>
<td>*AC</td>
<td>8</td>
</tr>
<tr>
<td>*AC</td>
<td>9</td>
</tr>
<tr>
<td>*AC</td>
<td>10</td>
</tr>
<tr>
<td>*AC</td>
<td>11</td>
</tr>
<tr>
<td>*AC</td>
<td>12</td>
</tr>
<tr>
<td>*AC</td>
<td>13</td>
</tr>
<tr>
<td>*AC</td>
<td>14</td>
</tr>
<tr>
<td>*AC</td>
<td>15</td>
</tr>
<tr>
<td>DT1</td>
<td>TOSEL</td>
</tr>
<tr>
<td>DT1</td>
<td>TOSEL</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>FCTA</td>
</tr>
<tr>
<td>DT1</td>
<td>GPLA</td>
</tr>
<tr>
<td>DT1</td>
<td>GPLA</td>
</tr>
<tr>
<td>DT1</td>
<td>GPLA</td>
</tr>
<tr>
<td>DT1</td>
<td>GPLA</td>
</tr>
<tr>
<td>DT1</td>
<td>GPLA</td>
</tr>
<tr>
<td>DT1</td>
<td>GPLA</td>
</tr>
<tr>
<td>DT1</td>
<td>GPLA</td>
</tr>
</tbody>
</table>
ID SAMPLE
APP DMAP
BEGIN %
$ $ SETS UP THE TABLES THAT DESCRIBE THE STRUCTURAL MODEL $ $
GP1 GEOM1,GEOM2,/GP,EQEXIN,CPDT,CSTH,BCPDT,SIL/V,M,LUSET/ V,M, HOCPDT %
SAVE LUSET,HOCPDT %
COND ERROR1,HOCPDT %
GP2 GEOM2,EQEXIN/ECT %
GP3 GEOM3,EQEXIN,GEOM2,CPPT/V,M,HOCRTN %
TAI ECT,EPT,HOCPDT,SIL,CPPT,CSTH/EST,CE1,GPCT,/,V,M,LUSET/ V,M, HOSIMP/C,M,1/V,M,NOGENL/V,M,CEMEL %
SAVE NOGENL,HOSIMP,CEMEL %
COND ERROR1,HOSIMP %
PARAM //C,M,ADD/V,M,NOGCCX/C,M,1/C,M,0 %
PARAM //C,M,ADD/V,M,NOGCC /C,M,1/C,M,0 %
SAVE NOGCCX,NOGCC %
COND JMFCEC3,NOGCC %
I-A CPCT,EDICT,REL/M,EGC3,CPST %
LABEL I-MFCEC3 %
COND ERROR1,NOGCC %
EWE CPCT,MDICT,REL/M,NOGC /C,M,-1/C,Y,WATMASS=1.0 %
COND LCPWa,CRDPM %
CPW BCPDT,CSTH,EQEXIN,NOGCC/OCFMC/V,T,CRDPM--1/C,Y,WATMASS %
GCP OCFMC,etc.,/V,M,CARDHO %
LABEL LCPW %
EQUIV KGC3,EGC3/NOGENL %
COND LNL1,NOGENL %
SHA3 CE1,EGC3/V,M,LUSET/V,M,NOGENL/C,M,-1 %
ADD KGC3,KGC3/EGC3 %
LABEL LNL1 %
PARAM //C,M,MPY/V,M,NSKIP/C,M,0/C,M,0 %
PARAM //C,M,SUB/V,M,DESINT/C,M,0/C,M,1 %
PICKE RRRR,DIJE,OJJE/DESINT %
PARAM //C,M,MPY/V,M,NOCUNIT/C,M,1/C,Y,ICAMS--1 %
PURGE CARNB/NOCARNB %
$ $ THE CONSTRAINTS OF THE FIRST SUBCASE ARE APPLIED $ $
SAVE MPF1,MPF2,SINGLE,OMIT,REACT,NSKIP,REPEAT,NOSET,NOA/NOA,SUBD %
GCF CPST CPST,SET,USET,SIL/M,CPST/V,M,NOCGST %
X Y AUGMT %
HINT LNL1,HOGCET %
CP 0CFMC,etc.,/V,M,CARDHO %
LABEL LNL1 %
HINT MPF1,MPF1/EGC3,HUGR,MPF1 %
COND LNL2,MPF1 %
CPSF       CPL, CPST, USET, SIL/OCPS/T/V, N, NOCPST $  
SAVE      NOCPST $  
COND      LBL14, NOCPST $  
OFF       OCPST,...// $  
LABEL     LBL14 $  
EQUIV     EGG, KNN/MCPFJ $  
COND      LBL15, MCPFJ $  
MCE1       USET, KG/CM $  
MCE2       USET, CM, KG,.../KNN,... $  
LABEL     LBL15 $  
EQUIV     KNN, RFF/SINGLE $  
COND      LBL16, SINGLE $  
SCE        USET, KNN,.../RFF, EPS, RSS... $  
LABEL     LBL16 $  
EQUIV     RFF, AAA/OMIT $  
COND      LBL17, OMIT $  
NMP       USET, RFF,.../GO, AAA, EEO, LOGO,... $  
LABEL     LBL17 $  
EQUIV     AAA, KFL/REACT $  
COND      LBL19, REACT $  
RMHC1      USET, AAA,.../KFL, KLR, KKR... $  
LABEL     LBL19 $  
RMHC2      KFL/LAA $  
COND      LBL19, REACT $  
RMHC3      KFL, KLR, KKR/DD $  
LABEL     LBL19 $  
COPT      POS, CP/DESEMT $  
PERS       ULV1, LOOVI/IWEQ $  
PERS       PST, CRI/IWEQ $  
$  
$ TOP OF THE ITERATIVE LOOP  
JMPF      LOGICTOP $  
LABEL     LOGICTOP $  
COND      FINISH, ELOOP $  
SFC2      USET, CM, KS, CGF, CO, EN, PG/QT, PO, PS, PL $  
SFC3      LIL, FFL, PL, JGG, FOF, PO/PG/RDH, ROV/GOV/V, N, OMIT/  
V, Y, IRKS-IV/V, N, RSIPFlo, V, N, EPSI $  
SAVE      EPSI $  
COND      LBG10, IRK $  
MATCPR    CPST, USET, SIL, NULIV/C, R, L $  
MATCPR    CPST, USET, SIL, ROLIV/F, E, O $  
LABEL     LBL10 $  
ADD      PHIDH, ULV1/ULV $  
ADD      PS, PSI/PSST $  
ADD      ROOV1, UOOV1/UOOVT $  
COND      LBL30, REACT $  
ADD      QP, OR/QUFT $  
LABEL     LBL30 $  
JMPAH     /C, A, ADD/V, V, N, NULIC, C, N, N, P $  
ADD      A31, EAT, AC, PRO, PGQ, PSR, R, EAQ, USEDF, AFRO/FRH, QHRL,  
CHLIV/V, N, UOE/GOE, L/K, FOUV1/V, Y, FURNRE $  
SAVE      Y, PUN $  
JMP       LBL14, LBL30, JMP  
MATCPR    CPST, USET, SIL, NULIV/C, R, L $  
MATCPR    CPST, USET, SIL, ROLIV/F, E, O $  
ADD      QP, OR/QUFT $  
ADD      ROOV1, UOOV1/UOOVT $
CPDR CASTCC,UGL,KELM,KUICT,ECT,EXE1N,CPECT,PGC,GC/0NRCY1,0GPPF1/
C,R,STATIC $ S
OFF ONRX1,0CPFR1,...// $ S
SDR CASECC,CETH,MPT,5IT,EXE1N,SIL,CPTT,DT,EPCF1,GC,UGV,EST,
XYCB,PGC/OPCI,0QCI,0QCV1,0ES1,0EF1,PGCV1/C,R,STATIC/V,N,
MOSORT2=1 $ S
SAVE MOSORT2 $ S
COND LBL21,MOSORT2 $ S
SDR OUGV1,0QCI,0QCI,0EF1,0ES1,/0UGV2,0PGC2,0QCV2,0EF2,0ES2, $ S
OFF OUGV2,0PGC2,0QCV2,0EF2,0ES2,//V,N,CARDNO $ S
SAVE CARDNO $ S
XYTRAN XYCB,0PGC2,0QCV2,0ES2,0EF2/XYPLTT/C/R,TRAN/C,N,PSET/V,N,
PFILE/V,N,CARDNO $ S
SAVE PFILE,CARDNO $ S
XYLOT XYLOT// $ S
JUMP DLLOT $ S
LABEL LBL24 $ S
OFF OUGV1,0QCI,0QCI,0EF1,0ES1,//V,N,CARDNO $ S
SAVE CARDNO $ S
COND P2,JUMPLOT $ S
LABEL DLLOT $ S
PLTLOT,GPS1S,ELS1S,CASECC,BCPDT,EXE1N,SIL,PGCV1,,CPECT,DES1/
PLTLOT2/V,N,NSIL/V,N,USET/V,N,JUMPLOT/V,N,PLTLC/V,N,PFILE $ S
SAVE PFILE $ S
PATHSC PLTLOT2// $ S
LABEL P2 $ S
JUMP FINIS $ S
LABEL $ S
PLTFRM /C,N,-2/C,N,FLLTR $ S
LABEL ERROR2 $ S
PLTFRM /C,N,-1/C,N,FLLTR $ S
LABEL $ S
PLTFRM /C,N,-4/C,N,FLLTR $ S
LABEL ERROR $ S
PLTFRM /C,N,-3/C,N,STATIC $ S
LABEL FINIS $ S
END $ S
DIAG 14
COND $ S
$ S
TITLE=SAMPLE
METHOD=10
ECHO=PATH
DISPLACEMENT = ALL
$ SUBCASE TO SET INITIAL ANGLE OF ATTACK
SUBCASE 1
SPEC = 1
$ SUBCASE 2 TO "CALCULATE EQUATIONS, ETC."
SUBCASE 2
SPEC = 2
INPUT BULK DATA DECK ECHO

1 2 3 4 5 6 7 8 9 10

$ TOLERANCE OF ITERATION LOOP
PARAM ELOOP .01

$ ROOT CHORD
PARAM RC 40.0

$ WING PLANFORM AREA, AERO MODEL
PARAM WAREA 2100.0

$ DYNAMIC PRESSURE
PARAM Q 59.34009

$ ANGLE OF ATTACK PARAMETER
PARAM ADAP 1.134925 0.0

$ PARAMETER TO INCLUDE CAMBER MATRIX IN CALCULATIONS
PARAM ICAMB 1

$ PARAMETER TO PICK WHICH EIGENVECTOR TO USE
PARAM LMODES 1

$ CONSTRAINS FOR BOTH SUBCASES
SPC1 1 2 1 THRU 6
SPC1 1 3 5
SPC1 1 45 1 THRU 10
SPC1 1 123 1 THRU 6
SPC1 2 45 1 THRU 10

$ DOF FOR ANALYSIS SET
ASET1 3 7 THRU 18

$ DNI CAMBER 0 2 1 1 17 1
DNI CAMBER 1 15 1.3123 17 1.3123

$ AERODYNAMIC PARAMETERS
AERO 223.4 40.0 7.650-2 1 +AERO1
+AERO1 1.0-15

$ DESCRIBES DIMENSIONS OF AERO MODEL
CAERO1 1000 1 2 610 1 +AERO1
+AERO1 0.0 0.0 2.25 60. 11.25 45. 2.25 32.5
CAERO1 1050 1 1 1 1 +AERO2
+AERO2 11.25 45. 2.25 16.25 15. 60. 2.25 15.
CAERO1 1075 1 1 1 1 +AERO3
+AERO3 27.5 45. 2.25 16.25 30. 60. 2.25 15.
ATTACH 10 11.6666671.0
PAEO1 1

$ COMIN1 1 1 1 3 9 7
COMIN2 2 7 3 5 11
COMIN2 3 3 7 9 15
### INPUT BULK DATA DECK ECHO

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>COXME1</td>
<td>4</td>
<td>4</td>
<td>9</td>
<td>11</td>
<td>17</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COXME2</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>10</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COXME3</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>12</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COXME4</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>10</td>
<td>16</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COXME5</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>18</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR3D</td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR50</td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR54</td>
<td>11</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR80</td>
<td>12</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR104</td>
<td>13</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR140</td>
<td>14</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR180</td>
<td>15</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR216</td>
<td>16</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR240</td>
<td>17</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR254</td>
<td>18</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>19</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>20</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>21</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>22</td>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>23</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>24</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>25</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>26</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>27</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>28</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>29</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>30</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>31</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSHEAR</td>
<td>32</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### EIGENVALUE CARD TO HELP DETERMINE INITIAL ANGLE OF ATTACK

<table>
<thead>
<tr>
<th>EIGCR</th>
<th>10</th>
<th>INV</th>
<th>0.0</th>
<th>1.0</th>
<th>1</th>
<th>1</th>
<th>*EIGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>EIGCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>POINT</th>
<th>7</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| GRID  | 1  | 0.0000 | 0.0000 | 2.2500 |
| GRID  | 2  | 0.0000 | 0.0000 | -2.2500 |
| GRID  | 3  | 20.0000 | 0.0000 | 3.0000 |
| GRID  | 4  | 20.0000 | 0.0000 | -3.0000 |
| GRID  | 5  | 40.0000 | 0.0000 | 2.2500 |
| GRID  | 6  | 40.0000 | 0.0000 | -2.2500 |
| GRID  | 7  | 7.5000 | 30.0000 | 1.6125 |
| GRID  | 8  | 7.5000 | 30.0000 | -1.6125 |
| GRID  | 9  | 25.0000 | 30.0000 | 2.1509 |
| GRID  | 10 | 25.0000 | 30.0000 | -2.1509 |
| GRID  | 11 | 42.5000 | 30.0000 | 1.6125 |
| GRID  | 12 | 42.5000 | 30.0000 | -1.6125 |
| GRID  | 13 | 15.0000 | *0.0000 | 1.1250 |
| GRID  | 14 | 15.0000 | 60.0000 | -1.1250 |
| GRID  | 15 | 30.0000 | 60.0000 | 1.5000 |
| GRID  | 16 | 30.0000 | 60.0000 | -1.5000 |
| GRID  | 17 | 45.0000 | 60.0000 | 1.1250 |
| GRID  | 18 | 45.0000 | 60.0000 | -1.1250 |

MAT1: 1 10.446 0.3 0.1
INPUT BULK DATA DECK ECHO

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>PQ04042</td>
<td>1</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQ04042</td>
<td>2</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQ04042</td>
<td>3</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQ04042</td>
<td>4</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQ04042</td>
<td>5</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQ04042</td>
<td>6</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQ04042</td>
<td>7</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQ04042</td>
<td>8</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQ04042</td>
<td>9</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROD 1</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROD 2</td>
<td>2.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROD 3</td>
<td>3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROD 4</td>
<td>4.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROD 5</td>
<td>5.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROD 6</td>
<td>6.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROD 7</td>
<td>7.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROD 8</td>
<td>8.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROD 9</td>
<td>9.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSHFAR 1</td>
<td>1</td>
<td>0.06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSHFAR 2</td>
<td>1</td>
<td>0.06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSHFAR 3</td>
<td>1</td>
<td>0.06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSHFAR 6</td>
<td>1</td>
<td>0.06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSHFAR 7</td>
<td>1</td>
<td>0.06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSHFAR 8</td>
<td>1</td>
<td>0.06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSHFAR 9</td>
<td>1</td>
<td>0.06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSHFAR 10</td>
<td>1</td>
<td>0.06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSHFAR 11</td>
<td>1</td>
<td>0.06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DEFINES THE TRANSFORMATION MATRIX FROM STRUCTURAL DEFORMATIONS TO AERO DISPLACEMENTS

SETI 1 1 1 5 7 9 11 13 +51
+51 15 17
SPLINE1 2000 1000 1000 1003 1 0.0
SPLINE1 1000 1050 1050 1050 1 0.0
SPLINE1 1075 1075 1075 1 0.0
DT1 TOSFL 0 4 +DT11
+DT11 ENDREC

TOP SURFACE ELEMENT LIST
DT1 TOSFL 1 1 2 3 4 ENDREC

ENDATA
The purpose of this study was to expand the capabilities of the Static Flexible Wing Aeroelastic Sequence that Captain Lance P. Chrisinger developed for NASIRAN as his Master's Thesis at AFIT. Captain Chrisinger developed a basic procedure to enable NASIRAN to analyze flexible wing airloads and stresses. That capability is expanded to enable analysis of standard wing models.